Oracle9*i*

XML API リファレンス - XDK および Oracle XML DB

リリース 2（9.2）

2002 年 7 月

部品番号 : J06308-01

ORACLE®

Oracle9*i* XML API リファレンス - XDK および Oracle XML DB, リリース 2（9.2）

部品番号 : J06308-01

原本名：Oracle9*i* XML API Reference - XDK and Oracle XML DB, Release 2 (9.2)

原本部品番号：A96616-01

原本著者：Roza Leyderman

原本協力者：Joan Gregoire、Shelley Higgins、Diana Lorentz、Jack Melnick、Cathy Shea、Nipun Agarwal、Chandramouli Balasubramaniam、Sivasankaran Chandrasekar、Dan Chiba、Mark Drake、Fei Ge、Stanley Guan、Bill Han、Wenyn He、Sam Idicula、Anish Karmarkar、K. Karun、Bhushan Khaladkar、Muralidhar Krishnaprasad、Bruce Lowenthal、Ian Macky、Anjana Manian、Anand Manikutty、Meghna Mehta、Nick Montoya、Steve Muench、Subramanian Muralidhar、Ravi Murthy、Anguel Novoselsky、Visar Nimani、Tomas Saulys、Mark Scardina、Eric Sedlar、Tarvinder Singh、Jinyu Wang、Jim Warner、Simon Wong、Neal Wyse、Tim Yu、Kongyi Zhou

目次

2 ドキュメント・オブジェクト・モデル（DOM）

3 Java での XML 処理（JAXP）

4 Java での XSLT 処理

5 Java での圧縮

6 XML Schema の処理

7 XML Class Generator for Java

8 XML SQL Utility for Java

9 XSQL Pages パブリッシング・フレームワーク for Java

10 Oracle XML JavaBeans

11 Simple Object Access Protocol（SOAP）for Java

第 V 部　XML データベース・サポート : Oracle XML DB for Java

21　Java API for XMLType

22　Oracle XML DB の JavaBeans API

23　Java/JNDI 用のリソース API

第 VI 部　XML データベース・サポート : Oracle XML DB for PL/SQL

24　XMLType API for PL/SQL

第 VIII 部　XML データベース・サポート：データベース URIType

35　データベース URI 型

索引

表目次

はじめに

このマニュアルでは、Oracle XML Developer's Kit（XDK）および Oracle XML DB Application Program Interface（API）について説明します。主に、これらの API に対応するファンクション、メソッドおよびプロシージャの構文を示します。ここでは、次の項目について説明します。

対象読者

このマニュアルは、Oracle9*i* データベースで XML アプリケーションを構築する開発者を対象としています。

このマニュアルの構成

このマニュアルは、次の章で構成されています。

第 I 部「XDK for Java パッケージ」

第 I 部では、XDK 用の Java API について説明します。

第 1 章「XML Parser for Java」

第 2 章「ドキュメント・オブジェクト・モデル（DOM）」

第 3 章「Java での XML 処理（JAXP）」

第 4 章「Java での XSLT 処理」

第 5 章「Java での圧縮」

第 6 章「XML Schema の処理」

第 7 章「XML Class Generator for Java」

第 8 章「XML SQL Utility for Java」

第 9 章「XSQL Pages パブリッシング・フレームワーク for Java」

第 10 章「Oracle XML JavaBeans」

第 11 章「Simple Object Access Protocol（SOAP）for Java」

第 12 章「TransX Utility for Java」

第 II 部「XDK for C パッケージ」

第 II 部では、XDK 用の C API について説明します。

第 III 部「XDK for C++ パッケージ」

第 III 部では、XDK 用の C++ API について説明します。

第 IV 部「XDK for PL/SQL」

第 IV 部では、XDK 用の PL/SQL API について説明します。

第 V 部「XML データベース・サポート：Oracle XML DB for Java」

第 V 部では、Oracle XML DB 用の Java API について説明します。

第 VI 部「XML データベース・サポート : Oracle XML DB for PL/SQL」

第 VI 部では、Oracle XML DB 用の PL/SQL API について説明します。

第 24 章「XMLType API for PL/SQL」

第 25 章「PL/SQL DOM API for XMLType」

第 26 章「PL/SQL Parser API for XMLType」

第 27 章「XMLType での PL/SQL XSLT 処理」

第 28 章「PL/SQL での DBMS_XMLSCHEMA およびカタログ・ビュー」

第 29 章「PL/SQL でのリソース管理およびアクセス制御」

第 30 章「PL/SQL での DBMS_XMLGEN を使用した問合せの生成」

第 31 章「Oracle XML DB Resource View API for PL/SQL」

第 32 章「Oracle XML DB バージョニング API for PL/SQL」

第 33 章「ConText インデックスの管理 : PL/SQL の DBMS_XDBT」

第 VII 部「XML データベース・サポート : SQLX」

第 VII 部では、Oracle XML DB 用の SQL API について説明します。

第 34 章「SQLX 関数および演算子」

第 VIII 部「XML データベース・サポート : データベース URIType」

第 VIII 部では、Oracle XML DB 用の URIType API について説明します。

第 35 章「データベース URI 型」

関連文書

詳細は、次の Oracle マニュアルを参照してください。

- 『Oracle9*i* データベース新機能』
- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* ケース・スタディ -XML アプリケーション』
- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* データベース・エラー・メッセージ』
- 『Oracle Text リファレンス』
- 『Oracle Text アプリケーション開発者ガイド』
- 『Oracle9*i* データベース概要』
- 『Oracle9*i* アプリケーション開発者ガイド - 基礎編』
- 『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』
- 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

このマニュアルの多くの例で、Oracle のインストール時にデフォルトとしてインストールされるシード・データベースのサンプル・スキーマを使用しています。スキーマの作成方法および使用方法の詳細は、『Oracle9*i* サンプル・スキーマ』を参照してください。

リリース・ノート、インストール・マニュアル、ホワイト・ペーパーまたはその他の関連文書は、OTN-J（Oracle Technology Network Japan）に接続すれば、無償でダウンロードできます。OTN-J を使用するには、オンラインでの登録が必要です。次の URL で登録できます。

`http://otn.oracle.co.jp/membership/`

すでに OTN-J のユーザー名およびパスワードを取得済であれば、次の OTN-J Web サイトの文書セクションに直接接続できます。

`http://otn.oracle.co.jp/document/`

さらに、次の書籍および URL を参照してください。

- 『The Oracle XML Handbook』XML Core Development Team 著、Oracle Press 出版、2000 年
- 『Building Oracle XML Applications』Steve Muench 著、O'Reilly & Associates 出版、2000 年
- 『XML Bible』Elliotte Rusty Harold 著、Hungry Minds, Inc 出版、2001 年
- 『XML Unleashed』Morrison その他著、SAMS 出版、1999 年

- 『Building XML Applications』St.Laurent、Cerami 著、McGraw-Hill Professional Publishing 出版、1999 年
- 『Building Web Sites with XML』Michael Floyd 著、Prentice Hall PTR 出版、1999 年
- 『Building Corporate Portals with XML』Finkelstein、Aiken、Zachman 著、McGraw-Hill Professional Publishing 出版、1999 年
- 『XML in a Nutshell』Elliotte Rusty Harold、W. Scott Means 著、O'Reilly & Associates 出版、2002 年
- 『XML in a Nutshell : A Desktop Quick Reference』Elliotte Rusty Harold、W. Scott Means 著、O'Reilly & Associates 出版、2001 年
- 『Learning XML』Erik T. Ray 著、O'Reilly & Associates 出版、2001 年
- 『XSLT』Doug Tidwell 著、O'Reilly & Associates 出版、2001 年
- http://www.xml.com/pub/rg/46
- http://www.xml.org/xmlorg_resources/index.shtml
- http://www.fawcette.com/xmlmag/
- http://www.webmethods.com/
- http://www.xmlwriter.com/
- http://webdevelopersjournal.com/articles/why_xml.html
- http://www.w3schools.com/xml/
- http://www.xml101.com/examples/

表記規則

この項では、このマニュアルの本文およびコード例で使用される表記規則について説明します。この項の内容は次のとおりです。

本文中の表記規則

本文では、特別な用語をより迅速に識別できるように、様々な表記規則を使用します。次の表に、それらの表記規則を説明し、その使用例を示します。

表記規則	意味	例
太字	太字は、本文中で定義されている用語または用語集に記載されている用語（あるいはその両方）を示します。	この句を指定すると、**索引構成表**が作成されます。
固定幅フォントの大文字	固定幅フォントの大文字は、システムが提供する要素を示します。このような要素には、パラメータ、権限、データ型、Recovery Manager キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージおよびメソッドが含まれます。また、システムが提供する列名、データベース・オブジェクト、データベース構造、ユーザー名およびロールも含まれます。	NUMBER 列に対してのみに、この句を指定できます。 BACKUP コマンドを使用して、データベースのバックアップを取ることができます。 USER_TABLES データ・ディクショナリ・ビュー内の TABLE_NAME 列を問い合せます。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびユーザーが提供する要素のサンプルを示します。このような要素には、コンピュータ名およびデータベース名、ネット・サービス名および接続識別子が含まれます。また、ユーザーが提供するデータベース・オブジェクトとデータベース構造と列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニットおよびパラメータ値も含まれます。 **注意：**大文字と小文字を組み合せて使用するプログラム要素もあります。これらの要素は、記載されているとおり入力してください。	sqlplus と入力して、SQL*Plus をオープンします。 パスワードは、orapwd ファイルで指定します。 /disk1/oracle/dbs ディレクトリ内のデータ・ファイルおよび制御ファイルのバックアップを取ります。 hr.departments 表には、department_id、department_name および location_id 列があります。 QUERY_REWRITE_ENABLED 初期化パラメータを true に設定します。 oe ユーザーとして接続します。 JRepUtil クラスが次のメソッドを実装します。
固定幅フォントの小文字のイタリック	固定幅フォントの小文字のイタリックは、プレースホルダまたは変数を示します。	*parallel_clause* を指定できます。 U*old_release*.SQL を実行します。ここで、*old_release* とはアップグレード前にインストールしたリリースを示します。

コード例中の表記規則

コード例は、SQL、PL/SQL、SQL*Plus または他のコマンドライン文を示します。コード例は、固定幅フォントで示され、次の例に示すとおり通常のテキストと区別されます。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表に、コード例で使用される表記規則を説明し、その使用例を示します。

表記規則	意味	例
[ ]	大カッコは、任意に選択する 1 つ以上の項目を囲みます。大カッコは入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコは 2 つ以上の項目を囲み、そのうちの 1 つの項目は必須です。中カッコは入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内の 2 つ以上のオプションの選択項目を表します。オプションのうちの 1 つを入力します。縦線は入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
. . .	水平省略記号は、次のいずれかを示します。	
	■ 例に直接関連しないコードの一部が省略されている。	CREATE TABLE ... AS *subquery*;
	■ コードの一部を繰り返すことができる。	SELECT *col1*, *col2*, ... , *coln* FROM employees;
. ..	垂直の省略記号は、例に直接関連しない複数の行が省略されていることを示します。	
その他の句読点	大カッコ、中カッコ、縦線および省略記号以外の句読点は、表示されているとおり入力する必要があります。	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
イタリック体	イタリック文は、プレースホルダまたは特定の値を指定する必要のある変数を示します。	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*
大文字	大文字は、システムが提供する要素を示します。これらの用語は、ユーザー定義の用語と区別するために大文字で示されます。用語が大カッコ内にないかぎり、表示されているとおりの順序および綴りで入力します。ただし、これらの用語は大 / 小文字が区別されないため、小文字でも入力できます。	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
小文字	小文字は、ユーザーが提供するプログラム要素を示します。たとえば、表名、列名、ファイル名などです。 **注意 :** 大文字と小文字を組み合せて使用するプログラム要素もあります。これらの要素は、記載されているとおり入力してください。	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

XML 対応テクノロジの新機能

ここでは、次の新機能について説明します。

- Oracle9i リリース 2（9.2）で導入された XML Developer's Kit の新機能
- Oracle9i リリース 2（9.2）で導入された XML データベースの新機能

Oracle9*i* リリース 2（9.2）で導入された XML Developer's Kit の新機能

- **XML Schema Processor for Java**

  World Wide Web Consortium（W3C）の XML Schema 勧告をサポートします。個々のスキーマの制約の検証に、外部 DocumentBuilder が不要になります。

- **XSL スタイルシート**

  スレッド・セーフな XSLStylesheet オブジェクトをサポートします。

- **XSQL Servlet**

  <xsql:include-owa> のパフォーマンスを向上させる新しいオプションです。

  <xsql:set-page-param> は、xpath="Expr" 属性をサポートします。

  CLOB 列および VARCHAR2 列からの XML の挿入が簡単になります。

  ポストされた XML を挿入するための新しい <xsql:include-posted-xml> アクション・ハンドラが追加されています。

  Apache FOP リリース 0.19 をサポートします。

  Cookie として設定された値の即時読込みをサポートします。

  単一の SQL 文による複数のパラメータ値の設定をサポートします。

- **Class Generator for Java**

  DTD Class Generator にデータのバインド機能が追加されています。

  XML インスタンス・ドキュメントを入力値として指定し、生成されたクラスにインスタンス・データをロードできます。

- **XDK for Java**

  XML SQL Utility（XSU）は、SAX 2.0、および SQL 問合せの XML Schema の生成をサポートします。

  DOM レベルの圧縮をサポートします。

  Oracle SOAP API が追加されています。

  Java XML Parser での SAX2.0 のサポートが拡張されています。

  XDK for Java によって JAXP 1.1 がサポートされています。

  Oracle TransX Utility は、データおよびテキストのロードに有効です。

  XML Schema Processor for Java は、LAX モードと STRICT モードの両方をサポートします。

  Java XML Parser での XML の圧縮がサポートされています。

- **XDK for C**

  Linux 版でリリースされています。

- **XDK for C++**

  Linux 版でリリースされています。

- **XDK for JavaBeans**

  新しい XMLDiff Bean が追加されています。

  SourceViewer Bean に内部 DTD サポートが追加されています。

- **OTN**

  新しい XDK のライブ・デモを次の URL で実行できます。

  `http://otn.oracle.com/tech/xml/xdk_sample/xdkdemo_faq.html`

  `http://otn.oracle.com/tech/xml/xdk_sample/xdkdemo_xsql.html`

  最新の XDK のテクニカル・ペーパー『Building Server-Side XML Schema Validation』を、次の URL で参照できます。

  `http://otn.oracle.com/tech/xml/xdk_sample/xdksample_093001i.html`

Oracle9*i* リリース 2（9.2）で導入された XML データベースの新機能

- **XMLType 表**

  今回のリリースでは、`XMLType` データ型を使用して `XMLType` の表を作成できます。

- **XMLType コンストラクタ**

  今回のリリースでは、いくつかの `XMLType` コンストラクタが追加されています。

- **W3C の XML Schema サポート**

  XML Schema に基づく `XMLType` オブジェクトを作成し、そのオブジェクトを継続して検証させます。

  XML Schema に基づく `XMLType` 表を作成します。

  `DBMS_XMLSCHEMA` パッケージを使用して注釈付き XML Schema を登録し、記憶域および型定義を共有します。

  受信 XML 文書を事前解析し、その文書をデフォルトの記憶表に自動的に転送します。

  Oracle XML DB に XML 文書またはインスタンスが追加された場合、その XML 文書またはインスタンスを、W3C の XML Schema に対して自動的に検証します。

XMLType に XMLIsSchemaValid() メソッドを使用して、XML 文書およびインスタンスを、XML Schema に対して明示的に検証します。

extractValue 演算子を使用して、データ型を認識する方法で XML 文書の一部を抽出します。

- **SQLX ファンクションおよび Oracle の拡張機能**

  既存のリレーショナル表およびオブジェクト・リレーショナル表から XML を生成します。これは、XML 関連仕様（SQL/XML）の ISO ANSI 草案（ISO/IEC 9075 Part 14 および ANSI）に基づいています。この草案では、データベース言語 SQL を XML と組み合せて使用する方法が定義されています。

- **抽出、条件チェックおよび更新に対する W3C の XPath のサポート**

  extract()、existsNode() および extractValue() は、名前空間に基づく操作をサポートします。

  extract()、existsNode() および extractValue() は、軸演算子を含む完全な XPath ファンクション・セットをサポートします。

  updateXML() は、XPath をロケータとして使用することによって、XMLType の一部の DOM のかわりに使用できます。

- **ToObject メソッド**

  ToObject メソッドを使用すると、コール元によって XMLType オブジェクトを PL/SQL オブジェクト型に変換できます。

- **XMLType ビュー**

  今回のリリースでは、XMLType に基づくビューがサポートされています。これらのビューを使用すると、データベース内のすべてのデータを XML として表示できます。XMLType ビューは、XML Schema に基づくビューまたは XML Schema に基づかないビューにもできます。

- **W3C の eXtensible Stylesheet Language Transformation（XSLT）サポート**

  XMLTransform() を使用すると、キャッシュされた XML 文書またはディスク上の XML 文書をデータベース上で変換できます。

- **XMLType に対する JDBC のサポート**

  Oracle XML DB では、データベース・クライアントによって XMLType をバインドおよび定義できます。JDBC サポートには、豊富な機能を持つ XMLType クラスが含まれています。XMLType クラスを使用すると、（Thick JDBC に対する）固有の XML 機能のサポートが可能です。

- **C ベースの PL/SQL DOM、Parser および XSLT API**

  今回のリリースには、データベース・コードに統合されている固有の PL/SQL DOM、Praser および XSLT API が含まれています。これらの PL/SQL API は、Oracle9*i* リリース 1（9.0.1）以上で XDK for PL/SQL の一部として付属している Java ベースの PL/SQL API と互換性があります。

- **Oracle XML DB: リポジトリ**

  データベースおよびその内容を、リソース（通常はファイルおよびフォルダ）を含むファイル・システムとして表示します。

  パス名を使用して、SQL および Java API を介してリソースにアクセスします。

  FTP、HTTP および WebDAV 用の固有の組込みプロトコル・サーバーを介して、リソースにアクセスします。

  Oracle XML DB リソースに対してアクセス制御リスト（ACL）・ベースのセキュリティを行います。

- **Oracle XML DB Resource API（PL/SQL）: DBMS_XDB**

  `DBMS_XDB` パッケージでは、Oracle XML DB リソースにアクセスし、操作するメソッドが提供されます。

- **Oracle XML DB Resource API（JNDI）**

  リソースの場所を検索し、コレクションを管理します。Oracle JVM プラットフォームのデータベース・サーバー内でのみ動作します。

- **Oracle XML DB Resource View API（SQL）**

  `RESOURCE-VIEW` は、パブリック `XMLType` ビューです。`RESOURCE-VIEW` を使用すると、データベース・インスタンス内のすべてのリソースに対して、パス名に基づく問合せを実行できます。

- **Oracle XML DB バージョニング : DBMS_XDB_VERSION**

  `DBMS_XDB_VERSION` パッケージでは、Oracle XML DB リソースをバージョニングするメソッドが提供されます。

- **Oracle XML DB の ACL セキュリティ**

  ACL ベースのセキュリティを実装するメソッドは、`DBMS_XDB` パッケージの一部として開発されています。これらのメソッドを使用すると、すべての `XMLType` オブジェクトに対する高パフォーマンスのアクセス制御リストを作成できます。

- **Oracle XML DB プロトコル・サーバー**

  プロトコル・サーバーによって、FTP、HTTP および WebDAV を介してすべてのフォルダリングされた `XMLType` 行にアクセスできます。

- **XDBUriType**

  URIType のサブタイプである XDBUriType は、Oracle XML DB 内のパス名を表します。

- **Oracle Enterprise Manager**

  Oracle Enterprise Manager では、グラフィカル・インタフェースを使用して Oracle XML DB を管理および構成できます。

- **Oracle Text の拡張機能**

  Oracle Text を使用して、Oracle9*i* データベース内で XMLType 型の列を固有にインデックス付けできます。

  CONTAINS() は、XPath 問合せの ora:contains および existsNode() 関数の一部として使用する新しい関数です。

  CTXXPATH は、XPath 検索を高速化するために existsNode() とともに使用する新しい索引タイプです。

- **Oracle Advanced Queuing（AQ）のサポート**

  iDAP が追加され、インターネット経由で AQ を簡単に使用できます。AQ XML Servlet は、HTTP および SOAP を使用して Oracle9*i* AQ にアクセスできます。

  iDAP は、SOAP リクエストの本体で使用される XML メッセージ構成を定義します。

  XMLType は、Oracle オブジェクト型の属性として埋め込むのではなく、AQ のペイロード型として直接使用できます。

第 I 部

XDK for Java パッケージ

第 I 部に含まれる章は、次のとおりです。

1

XML Parser for Java

XML を解析すると、XML 文書が処理され、様々な API を介してそれらの文書に含まれる情報にアクセスできるようになるため、XML をサポートするように既存のアプリケーションを簡単に拡張できます。この章の内容は次のとおりです。

- DefaultXMLDocumentHandler クラス
- DocumentBuilder クラス
- DOMParser クラス
- NodeFactory クラス
- oraxml クラス
- SAXAttrList クラス
- SAXParser クラス
- XMLParseException クラス
- XMLParser クラス
- XMLToken クラス
- XMLTokenizer クラス
- NSName クラス
- XMLError クラス
- XMLException クラス

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

DefaultXMLDocumentHandler クラス

DefaultXMLDocumentHandler の説明

このクラスは、XMLDocumentHandler インタフェースのデフォルトの動作を実装します。アプリケーション開発者は、このインタフェースの一部のみを実装する必要がある場合、このクラスを拡張できます。

DefaultXMLDocumentHandler の構文

```
public class DefaultXMLDocumentHandler implements
oracle.xml.parser.v2.XMLDocumentHandler
```

oracle.xml.parser.v2.DefaultXMLDocumentHandler

DefaultXMLDocumentHandler の実装済インタフェース

XMLDocumentHandler

DefaultXMLDocumentHandler のメソッド

表 1-1 DefaultXMLDocumentHandler のメソッドの概要

メソッド	説明
DefaultXMLDocumentHandler()（1-3 ページ）	デフォルトのドキュメント・ハンドラを作成します。
cDATASection()（1-3 ページ）	CDATA セクションの通知を受け取ります。
comment()（1-4 ページ）	コメントの通知を受け取ります。
endDoctype()（1-4 ページ）	DTD の終わりの通知を受け取ります。
endElement()（1-4 ページ）	要素の終わりの通知を受け取ります。
endPrefixMapping()（1-5 ページ）	Uniform Resource Identifier（URI）の接頭辞マッピングの有効範囲を終了します。
getHandler()（1-5 ページ）	次のパイプライン・ノード・ハンドラを取得します。
setDoctype()（1-6 ページ）	DTD の通知を受け取ります。DTD を設定します。
setError()（1-6 ページ）	XMLError ハンドラの通知を受け取ります。
setHandler()（1-6 ページ）	次のパイプライン・ノード・ハンドラの通知を受け取ります。
setTextDecl()（1-7 ページ）	テキスト XML 宣言の通知を受け取ります。

表 1-1 DefaultXMLDocumentHandler のメソッドの概要（続き）

メソッド	説明
setXMLDecl()（1-7 ページ）	XML 宣言の通知を受け取ります。
setXMLSchema()（1-8 ページ）	XMLSchema オブジェクトの通知を受け取ります。
skippedEntity()（1-8 ページ）	スキップされたエンティティの通知を受け取ります。
startElement()（1-9 ページ）	要素の始まりの通知を受け取ります。
startPrefixMapping()（1-10 ページ）	URI 名前空間の接頭辞マッピングの有効範囲を開始します。

DefaultXMLDocumentHandler()

説明

デフォルトのドキュメント・ハンドラを作成します。

構文

```
public  DefaultXMLDocumentHandler();
```

cDATASection()

説明

CDATA セクションの通知を受け取ります。パーサーは、CDATA セクションが検索されるたびに、このメソッドを 1 回コールします。`org.xml.sax.SAXException` が発生します。この例外は、`SAXException` またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void cDATASection( char[] ch,
                          int start,
                          int length);
```

パラメータ	説明
ch	CDATA セクションの文字列を保持する配列。
start	文字配列内の開始位置。
length	文字配列のうちの使用する文字数。

comment()

説明

コメントの通知を受け取ります。パーサーは、コメントが検索されるたびに、このメソッドを 1 回コールします。コメントは、主なドキュメント要素の前後で発生することに注意してください。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void comment( String data);
```

パラメータ	説明
data	コメント・データ（指定されていない場合は null）。

endDoctype()

説明

DTD の終わりの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void endDoctype();
```

endElement()

説明

要素の終わりの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。次の表に、オプションを示します。

構文	説明
public void endElement(NSName elem);	要素を使用して、その要素の終わりの通知を受け取ります。

構文	説明
public void endElement(String namespaceURI, String localName, String qName);	名前空間、ローカル名および修飾名を使用して、要素の終わりの通知を受け取ります。

パラメータ	説明
elem	要素名を表す NSName オブジェクト。
namespaceURI	名前空間 URI（要素に名前空間 URI が設定されていない場合または名前空間処理が実行されていない場合は、空の文字列）。
localName	接頭辞なしのローカル名（名前空間処理が実行されていない場合は空の文字列）。
qname	XML 1.0 の接頭辞付き修飾名（修飾名が使用不可能な場合は空の文字列）。

endPrefixMapping()

説明

URI の接頭辞マッピングの有効範囲を終了します。`org.xml.sax.SAXException` が発生します。この例外は、`SAXException` またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void endPrefixMapping( String prefix);
```

getHandler()

説明

次のパイプライン・ノード・ハンドラ・ノードを戻します。

構文

```
public XMLDocumentHandler getHandler();
```

setDoctype()

説明

DTD の通知を受け取ることができるように、DTD を設定します。
org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setDoctype( DTD dtd);
```

パラメータ	説明
dtd	DTD。

setError()

説明

XMLError ハンドラの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setError( XMLError he);
```

パラメータ	説明
err	XMLError オブジェクト。

setHandler()

説明

次のパイプライン・ノード・ハンドラの通知を受け取ります。
org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setHandler( XMLDocumentHandler h);
```

パラメータ	説明
h	XMLDocumentHandler ノード。

setTextDecl()

説明

テキスト XML 宣言の通知を受け取ります。パーサーは、テキスト XMLDecl ごとに、このメソッドを 1 回コールします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setTextDecl( String version,
                         String encoding);
```

パラメータ	説明
version	バージョン番号。
encoding	エンコーディング名（指定されていない場合は null）。

setXMLDecl()

説明

XML 宣言の通知を受け取ります。パーサーは、XMLDecl ごとに、このメソッドを 1 回コールします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setXMLDecl( String version,
                        String standalone,
                        String encoding);
```

パラメータ	説明
version	バージョン番号。
standalone	スタンドアロン値（指定されていない場合は null）。
encoding	エンコーディング名（指定されていない場合は null）。

setXMLSchema()

説明

XMLSchema オブジェクトの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setXMLSchema( Object s);
```

パラメータ	説明
s	XMLSchema オブジェクト。

skippedEntity()

説明

スキップされたエンティティの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void skippedEntity( String name);
```

パラメータ	説明
name	スキップされたエンティティの名前。パラメータ・エンティティの場合は、「%」で始まり、外部 DTD サブセットの場合は、「[dtd]」という文字列になります。

startElement()

説明

要素の始まりの通知を受け取ります。`org.xml.sax.SAXException` が発生します。この例外は、`SAXException` またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。次の表に、オプションを示します。

構文	説明
public void startElement(NSName elem, SAXAttrList attrlist);	要素を使用して、その要素の始まりの通知を受け取ります。
public void startElement(String namespaceURI, String localName, String qName, org.xml.sax.Attributes atts);	名前空間、ローカル名および修飾名を使用して、要素の始まりの通知を受け取ります。

パラメータ	説明
elem	要素名を表す NSName オブジェクト。
attlist	その要素の属性リスト。
namespaceURI	名前空間 URI（要素に名前空間 URI が設定されていない場合または名前空間処理が実行されていない場合は、空の文字列）。
localName	接頭辞なしのローカル名（名前空間処理が実行されていない場合は空の文字列）。
qname	接頭辞付き修飾名（修飾名が使用できない場合は空の文字列）。
atts	要素に追加された属性（属性が存在しない場合は空の Attributes オブジェクト）。

startPrefixMapping()

説明

URI 名前空間の接頭辞マッピングの有効範囲を開始します。
`org.xml.sax.SAXException` が発生します。この例外は、`SAXException` またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void startPrefixMapping( String prefix,
                                String uri);
```

パラメータ	説明
prefix	宣言対象の名前空間の接頭辞。
uri	接頭辞がマップされる名前空間。

DocumentBuilder クラス

DocumentBuilder の構文

```
public class DocumentBuilder
```

oracle.xml.parser.v2.DocumentBuilder

DocumentBuilder の説明

このクラスは、XMLDocumentHandler（使用できません）および ContentHandler を実装して、SAX 2.0 イベントから DOM ツリーを構築します。XMLDocumentHandler イベントでは、下位互換性がサポートされています。

DocumentBuilder のメソッド

表 1-2 DocumentBuilder のメソッドの概要

メソッド	説明
DocumentBuilder()（1-13 ページ）	XMLDocumentHandler として使用可能なドキュメント・ビルダーを作成します。
attributeDecl()（1-13 ページ）	属性の型宣言をレポートします。
cDATASection()（1-14 ページ）	要素内の CDATA セクション・データの通知を受け取ります。
characters()（1-14 ページ）	要素内の文字データの通知を受け取ります。
comment()（1-15 ページ）	コメントの通知を受け取ります。
elementDecl()（1-15 ページ）	要素型宣言をレポートします。
endCDATA()（1-16 ページ）	CDATA セクションの終わりをレポートします。
endDoctype()（1-16 ページ）	DTD の終わりの通知を受け取ります。
endDocument()（1-16 ページ）	ドキュメントの終わりの通知を受け取ります。
endDTD()（1-17 ページ）	DTD 宣言の終わりをレポートします。
endElement()（1-17 ページ）	要素の終わりの通知を受け取ります。
endEntity()（1-18 ページ）	エンティティの終わりをレポートします。
externalEntityDecl()（1-18 ページ）	解析対象外部エンティティ宣言をレポートします。

表 1-2 DocumentBuilder のメソッドの概要（続き）

メソッド	説明
getCurrentNode()（1-19 ページ）	作成対象の現行ノードを戻します。
getDocument()（1-19 ページ）	作成対象のドキュメントを取得します。
ignorableWhitespace()（1-19 ページ）	要素の内容にある無視可能な空白の通知を受け取ります。
internalEntityDecl()（1-20 ページ）	内部エンティティ宣言をレポートします。
processingInstruction()（1-20 ページ）	処理命令の通知を受け取ります。
retainCDATASection()（1-21 ページ）	CDATA セクションを保存するフラグを設定します。
setDebugMode()（1-21 ページ）	ドキュメントのデバッグ情報を有効にするフラグを設定します。
setDoctype()（1-21 ページ）	DTD の通知を受け取ります。DTD を設定します。
setDocumentLocator()（1-22 ページ）	ドキュメント・イベント用の Locator オブジェクトを受け取ります。デフォルトでは、何も実行されません。アプリケーション開発者は、ロケータを格納して他のドキュメント・イベントで使用する必要がある場合、このメソッドをサブクラスでオーバーライドできます。
setNodeFactory()（1-22 ページ）	カスタム DOM ツリーの作成に使用するオプションの NodeFactory を設定します。
setTextDecl()（1-23 ページ）	テキスト XML 宣言の通知を受け取ります。パーサーは、テキスト XML 宣言ごとに、このメソッドを 1 回コールします。
setXMLDecl()（1-23 ページ）	XML 宣言の通知を受け取ります。パーサーは、XML 宣言ごとに、このメソッドを 1 回コールします。
startCDATA()（1-24 ページ）	CDATA セクションの始まりをレポートします。
startDocument()（1-24 ページ）	ドキュメントの始まりの通知を受け取ります。
startDTD()（1-24 ページ）	DTD 宣言（存在する場合）の始まりをレポートします。
startElement()（1-25 ページ）	要素の始まりの通知を受け取ります。
startEntity()（1-26 ページ）	内部および外部の XML エンティティの始まりをレポートします。すべての startEntity イベントおよび endEntity イベントは、適切にネストされる必要があります。

DocumentBuilder()

説明

デフォルトのコンストラクタです。XMLDocumentHandler として使用可能なドキュメント・ビルダーを作成します。

構文

```
public  DocumentBuilder();
```

attributeDecl()

説明

属性の型宣言をレポートします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void attributeDecl( String eName,
                           String aName,
                           String type,
                           String valueDefault,
                           String value);
```

パラメータ	説明
eName	対応する要素の名前。
aName	属性名。
type	属性の型を表す文字列。
valueDefault	属性のデフォルトを表す文字列（「#IMPLIED」、「#REQUIRED」または「#FIXED」）。これらの文字列のいずれも適用されない場合は null。
value	属性のデフォルト値を表す文字列（文字列がない場合は null）。

cDATASection()

説明

要素内の CDATA セクション・データの通知を受け取ります。
org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void cDATASection( char[] ch,
                          int start,
                          int length);
```

パラメータ	説明
ch	CDATA セクションの文字列を保持する配列。
start	配列内の開始位置。
length	配列のうちの使用する文字数。

characters()

説明

要素内の文字データの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void characters( char[] ch,
                        int start,
                        int length)
```

パラメータ	説明
ch	文字データを保持する配列。
start	配列内の開始位置。
length	配列のうちの使用する文字数。

comment()

説明

コメントの通知を受け取ります。ドキュメント内の任意の場所に存在する XML コメントをレポートします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。次の表に、オプションを示します。

構文	説明
public void comment(char[] ch, int start, int length);	コメント文字配列のパラメータを使用して、コメントの通知を受け取ります。
public void comment(String data);	コメント・データを使用して、コメントの通知を受け取ります。

パラメータ	説明
ch	コメント内の文字を保持する配列。
start	配列内の開始位置。
length	配列のうちの使用する文字数。
data	コメント・データ（指定されていない場合は null）。

elementDecl()

説明

要素型宣言をレポートします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void elementDecl( String name,
                         String model);
```

パラメータ	説明
name	要素型名。
model	正規化された文字列としてのコンテンツ・モデル。

endCDATA()

説明

CDATA セクションの終わりをレポートします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void endCDATA();
```

endDoctype()

説明

DTD の終わりの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void endDoctype();
```

endDocument()

説明

ドキュメントの終わりの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void endDocument();
```

endDTD()

説明

DTD 宣言の終わりをレポートします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void endDTD();
```

endElement()

説明

要素の終わりの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。次の表に、オプションを示します。

構文	説明
public void endElement(NSName elem);	要素を使用して、その要素の終わりの通知を受け取ります。
public void endElement(String namespaceURI, String localName, String qName);	名前空間、ローカル名および修飾名を使用して、要素の終わりの通知を受け取ります。

パラメータ	説明
elem	要素名を表す NSName オブジェクト。
namespaceURI	名前空間 URI（要素に名前空間 URI が設定されていない場合または名前空間処理が実行されていない場合は、空の文字列）。
localName	接頭辞なしのローカル名（名前空間処理が実行されていない場合は空の文字列）。
qname	XML 1.0 の接頭辞付き修飾名（修飾名が使用不可能な場合は空の文字列）。

endEntity()

説明

エンティティの終わりをレポートします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void endEntity( String name);
```

パラメータ	説明
name	終了中のエンティティの名前。

externalEntityDecl()

説明

解析対象外部エンティティ宣言をレポートします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void externalEntityDecl( String name,
                                 String publicId,
                                 String systemId);
```

パラメータ	説明
name	エンティティ名。パラメータ・エンティティの場合は、「%」で始まります。
publicId	エンティティ宣言の宣言された公開識別子（宣言されていない場合は null）。
systemId	エンティティの宣言されたシステム識別子。

getCurrentNode()

説明

作成対象の現行 XMLNode を戻します。

構文

```
public XMLNode getCurrentNode();
```

getDocument()

説明

作成対象のドキュメントを戻します。

構文

```
public XMLDocument getDocument();
```

ignorableWhitespace()

説明

要素の内容にある無視可能な空白の通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void ignorableWhitespace( char[] ch,
                                 int start,
                                 int length);
```

パラメータ	説明
ch	空白文字。
start	文字配列内の開始位置。
length	文字配列のうちの使用する文字数。

internalEntityDecl()

説明

内部エンティティ宣言をレポートします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void internalEntityDecl( String name,
                                String value);
```

パラメータ	説明
name	エンティティ名。パラメータ・エンティティの場合は、「%」で始まります。
value	エンティティの置換テキスト。

processingInstruction()

説明

処理命令の通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void processingInstruction( String target,
                                   String data);
```

パラメータ	説明
target	処理命令のターゲット。
data	処理命令のデータ（指定されていない場合は null）。

retainCDATASection()

説明

CDATA セクションを保存するフラグを設定します。

構文

```
public void retainCDATASection( boolean flag);
```

パラメータ	説明
flag	CDATA セクションを保存するかどうかを指定します。保存する場合は `true`、保存しない場合は `false` を指定します。

setDebugMode()

説明

ドキュメントのデバッグ情報を有効にするフラグを設定します。

構文

```
public void setDebugMode( boolean flag);
```

パラメータ	説明
flag	デバッグ情報を保存するかどうかを指定します。保存する場合は `true`、保存しない場合は `false` を指定します。

setDoctype()

説明

DTD の通知を受け取ることができるように、DTD を設定します。`org.xml.sax.SAXException` が発生します。この例外は、`SAXException` またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setDoctype( DTD dtd);
```

パラメータ	説明
did	ドキュメントの DTD

setDocumentLocator()

説明

ドキュメント・イベント用の Locator オブジェクトの通知を受け取ることができるように、ロケータを設定します。デフォルトでは、何も実行されません。アプリケーション開発者は、そのロケータを格納して他のドキュメント・イベントで使用する必要がある場合、このメソッドをサブクラスでオーバーライドできます。

構文

```
public void setDocumentLocator( org.xml.sax.Locator locator);
```

パラメータ	説明
locator	すべての SAX ドキュメント・イベント用のロケータ。

setNodeFactory()

説明

カスタム DOM ツリーの作成に使用するオプションの NodeFactory を設定します。

構文

```
public void setNodeFactory( NodeFactory f);
```

パラメータ	説明
f	NodeFactory。

setTextDecl()

説明

テキスト XML 宣言の通知を受け取ります。パーサーは、テキスト XML 宣言ごとに、このメソッドを 1 回コールします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setTextDecl( String version,
                         String encoding);
```

パラメータ	説明
version	バージョン番号（指定されていない場合は null）。
encoding	エンコーディング名（指定されていない場合は null）。

setXMLDecl()

説明

XML 宣言の通知を受け取ります。パーサーは、XML 宣言ごとに、このメソッドを 1 回コールします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setXMLDecl( String version,
                        String standalone,
                        String encoding);
```

パラメータ	説明
version	バージョン番号。
standalone	スタンドアロン値（指定されていない場合は null）。
encoding	エンコーディング名（指定されていない場合は null）。

startCDATA()

説明

CDATA セクションの始まりをレポートします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void startCDATA();
```

startDocument()

説明

ドキュメントの始まりの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void startDocument();
```

startDTD()

説明

DTD 宣言（存在する場合）の始まりをレポートします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void startDTD( String name,
                      String publicId,
                      String systemId);
```

パラメータ	説明
name	ドキュメント・タイプ名。
publicId	外部 DTD サブセットの宣言された公開識別子（宣言されていない場合は null）。

パラメータ	説明
systemId	外部 DTD サブセットの宣言されたシステム識別子（宣言されていない場合は null）。

startElement()

説明

要素の始まりの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。次の表に、オプションを示します。

構文	説明
public void startElement(NSName elem, SAXAttrList attrlist);	要素を使用して、その要素の始まりの通知を受け取ります。
public void startElement(String namespaceURI, String localName, String qName, org.xml.sax.Attributes atts);	名前空間、ローカル名および修飾名を使用して、要素の始まりの通知を受け取ります。

パラメータ	説明
elem	要素名を表す NSName オブジェクト。
attlist	その要素の属性リスト。
naemspaceURI	名前空間 URI（要素に名前空間 URI が設定されていない場合または名前空間処理が実行されていない場合は、空の文字列）。
localName	接頭辞なしのローカル名（名前空間処理が実行されていない場合は空の文字列）。
qName	接頭辞付き修飾名（修飾名が使用できない場合は空の文字列）。
atts	要素に追加された属性（属性が存在しない場合は空の Attributes オブジェクト）。

startEntity()

説明

内部および外部の XML エンティティの始まりをレポートします。すべての startEntity イベントおよび endEntity イベントは、適切にネストされる必要があります。
org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void startEntity( String name);
```

パラメータ	説明
name	エンティティ名。パラメータ・エンティティの場合は、「%」で始まり、外部 DTD サブセットの場合は、「[dtd]」という文字列になります。

DOMParser クラス

DOMParser の構文

```
public class DOMParser
```

oracle.xml.parser.v2.DOMParser

DOMParser の説明

このクラスは、World Wide Web Consortium（W3C）の勧告に準拠した eXtensible Markup Language（XML）1.0 のパーサーを実装して、XML の文書の解析および DOM ツリーの構築を行います。

DOMParser のフィールド

表 1-3 DOMParser のフィールド

フィールド	構文	説明
DEBUG_MODE	public static final java.lang.String DEBUG_MODE	デバッグ・モード（Boolean.TRUE または Boolean.FALSE）。
ERROR_ENCODING	public static final java.lang.String ERROR_ENCODING	エラー・ストリームを介したエラー・レポートのエンコーディング（ERROR_STREAM が設定されている場合のみ）。
ERROR_STREAM	public static final java.lang.String ERROR_STREAM	エラーをレポートするためのエラー・ストリーム。オブジェクトは、OutputStream または PrintWriter です。エラー・ハンドラが設定されている場合、この属性は無視されます。
NODE_FACTORY	public static final java.lang.String NODE_FACTORY	カスタム・ノードを作成するための NodeFactory。
SHOW_WARNINGS	public static final java.lang.String SHOW_WARNINGS	警告を無視するブール値（Boolean.TRUE または Boolean.FALSE）。

DOMParser のメソッド

表 1-4 DOMParser のメソッドの概要

メソッド	説明
DOMParser()（1-28 ページ）	新しいパーサー・オブジェクトを作成します。
getAttribute()（1-28 ページ）	実際の実装の特定の属性を戻します。
getDoctype()（1-29 ページ）	DTD を取得します。
getDocument()（1-29 ページ）	ドキュメントを取得します。
parseDTD()（1-29 ページ）	指定された入力ソースから XML 外部 DTD を解析します。
reset()（1-30 ページ）	パーサーの状態をリセットします。
retainCDATASection()（1-31 ページ）	CDATA セクションを保存するかどうかを判別します。
setAttribute()（1-31 ページ）	実際の実装に特定の属性を設定します。
setDebugMode()（1-31 ページ）	ドキュメントのデバッグ情報を有効にするフラグを設定します。
setErrorStream()（1-32 ページ）	エラーおよび警告用の出力ストリームを設定します。
setNodeFactory()（1-33 ページ）	ノード・ファクトリを設定します。
showWarnings()（1-33 ページ）	警告を出力するかどうかを判別します。

DOMParser()

説明

新しいパーサー・オブジェクトを作成します。

構文

```
public  DOMParser();
```

getAttribute()

説明

実際の実装の特定の属性を戻します。実際の実装がその属性を認識しない場合は、`IllegalArgumentException` が発生します。

構文

```
public java.lang.Object getAttribute( String name);
```

パラメータ	説明
name	属性名。

getDoctype()

説明

DTD を戻します。

構文

```
public DTD getDoctype();
```

getDocument()

説明

解析対象のドキュメントを戻します。

構文

```
public XMLDocument getDocument();
```

parseDTD()

説明

指定された入力ストリームから XML 外部 DTD を解析します。次の例外が発生します。

- `XMLParseException`（構文エラーまたは他のエラーが発生した場合）
- `SAXException`（`SAXException` またはそのサブクラスのインスタンスで、別の例外が隠されている可能性あり）
- `IOException`（I/O エラーが発生した場合）

次の表に、オプションを示します。

構文	説明
public final void parseDTD(org.xml.sax.InputSource in, String rootName);	入力ソースから XML 外部 DTD を解析します。
public final void parseDTD(java.io.InputStream in, String rootName);	入力ストリームから XML 外部 DTD を解析します。外部エンティティおよび DTD を解決するためのベース Uniform Resource Locator（URL）を設定する必要があります。
public final void parseDTD(java.io.Reader r, String rootName);	リーダーから XML 外部 DTD を解析します。外部エンティティおよび DTD を解決するためのベース URL を設定する必要があります。
public final void parseDTD(String in, String rootName);	文字列から XML 外部 DTD を解析します。
public final void parseDTD(java.net.URL url, String rootName);	指定された URL が指す XML 外部 DTD を解析し、対応する XML 文書の階層を作成します。

パラメータ	説明
in	解析する入力。
rootName	ルート要素として使用する要素。
r	解析する XML データを含むリーダー。
url	解析する XML 文書を指す URL。

reset()

説明

パーサーの状態をリセットします。

構文

```
public void reset();
```

retainCDATASection()

説明

CDATA セクションを保存するかどうかを判別します。

構文

```
public void retainCDATASection( boolean flag);
```

パラメータ	説明
flag	CDATA セクションを保存する場合は `true`（デフォルト）、 CDATA セクションを Text ノードに変換する場合は `false`。

setAttribute()

説明

実際の実装に特定の属性を設定します。実際の実装がその属性を認識しない場合は、`IllegalArgumentException` が発生します。

構文

```
public void setAttribute( String name,
                          Object value);
```

パラメータ	説明
name	属性名。
value	属性値。

setDebugMode()

説明

ドキュメントのデバッグ情報を有効にするフラグを設定します。

構文

```
public void setDebugMode( boolean flag);
```

パラメータ	説明
flag	デバッグ情報を格納するかどうかを判別します。

setErrorStream()

説明

エラーおよび警告を出力するための出力ストリームを設定します。エラー用の出力ストリームが指定されていない場合、パーサーは、標準エラー出力ストリーム System.err を使用して、エラーおよび警告を出力します。また、指定されたエンコーディングがサポートされていない場合は、例外が発生します。次の表に、オプションを示します。

構文	説明
public final void setErrorStream(java.io.OutputStream out);	エラーおよび警告を出力するための出力ストリームを設定します。
public final void setErrorStream(java.io.OutputStream out, String enc);	エラーおよび警告を出力するための出力ストリームを設定します。指定されたエンコーディングがサポートされていない場合は、IOExceptionが発生します。
public final void setErrorStream(java.io.PrintWriter out);	エラーおよび警告を出力するためのプリント・ライターを設定します。エラー・ストリームの設定中にI/Oエラーが発生した場合は、IOExceptionが発生します。

パラメータ	説明
out	エラーおよび警告の出力先。
enc	使用するエンコーディング。

setNodeFactory()

説明

ノード・ファクトリを設定します。アプリケーションは、このメソッドを介して NodeFactory を拡張して登録できます。パーサーは、ユーザーが提供する NodeFactory を使用して、DOM ツリーのノードを作成します。無効なファクトリを設定した場合は、`XMLParseException` が発生します。

構文

```
public void setNodeFactory( NodeFactory factory);
```

パラメータ	説明
factory	設定する NodeFactory。

showWarnings()

説明

警告を出力するかどうかを判別します。

構文

```
public void showWarnings( boolean flag);
```

パラメータ	説明
flag	警告を表示する必要があるかどうかを指定します。

NodeFactory クラス

NodeFactory の説明

このクラスは、解析中に構築される DOM ツリーの様々なノードを作成するメソッドを指定します。アプリケーションは、これらのメソッドをオーバーライドして、解析中に DOM ツリーに追加する独自のカスタム・クラスを作成します。アプリケーションは、XMLParser の setNodeFactory() メソッドを使用して、独自のノード・ファクトリを登録する必要があります。これらのメソッドが null ポインタを戻した場合、ノードは DOM ツリーに追加されません。

NodeFactory の構文

```
public class NodeFactory extends java.lang.Object implements java.io.Serializable

java.lang.Object
  |
  +--oracle.xml.parser.v2.NodeFactory
```

NodeFactory の実装済インタフェース

```
java.io.Serializable
```

NodeFactory のメソッド

表 1-5 NodeFactory のメソッドの概要

メソッド	説明
NodeFactory()（1-35 ページ）	クラス・コンストラクタです。
createAttribute()（1-35 ページ）	指定したタグおよびテキストを含む属性ノードを作成して戻します。
createCDATASection()（1-36 ページ）	指定したテキストを含む CDATA ノードを作成して戻します。
createComment()（1-36 ページ）	指定したテキストを含むコメント・ノードを作成して戻します。
createDocument()（1-37 ページ）	ドキュメント・ノードを作成して戻します。このメソッドが null ポインタを戻すことはありません。
createDocumentFragment()（1-37 ページ）	指定したタグを含むドキュメント・フラグメント・ノードを作成して戻します。
createElement()（1-37 ページ）	指定したタグを含む要素ノードを作成して戻します。

表 1-5 NodeFactory のメソッドの概要（続き）

メソッド	説明
createElementNS()（1-38 ページ）	指定したローカル名、接頭辞および名前空間 URI を含む要素ノードを作成して戻します。
createEntityReference()（1-38 ページ）	指定したタグを含む実体参照ノードを作成して戻します。
createProcessingInstruction()（1-39 ページ）	指定したタグおよびテキストを含む処理命令ノードを作成して戻します。
createTextNode()（1-39 ページ）	指定したテキストを含む Text ノードを作成して戻します。

NodeFactory()

説明

クラス・コンストラクタです。

構文

```
public  NodeFactory();
```

createAttribute()

説明

属性ノードを作成して戻します。次の表に、オプションを示します。

構文	説明
public XMLAttr createAttribute(String tag, String text);	指定したタグおよびテキストを含む属性ノードを作成して戻します。
public XMLAttr createAttribute(String localName, String prefix, String namespaceURI, String value);	指定したローカル名、接頭辞、名前空間 URI および値を含む属性ノードを作成して戻します。

パラメータ	説明
tag	ノード名。
text	ノードに対応するテキスト。
localName	ノード名。
prefix	ノードの接頭辞。
naemspaceURI	ノードの名前空間。
value	ノードに対応する値。

createCDATASection()

説明

指定したテキストを含む CDATA ノードを作成して戻します。

構文

```
public XMLCDATA createCDATASection( String text);
```

パラメータ	説明
text	ノードに対応するテキスト。

createComment()

説明

指定したテキストを含むコメント・ノードを作成して戻します。

構文

```
public XMLComment createComment( String text);
```

パラメータ	説明
text	ノードに対応するテキスト。

createDocument()

説明
ドキュメント・ノードを作成して戻します。このメソッドが null ポインタを戻すことはありません。

構文
```
public XMLDocument createDocument();
```

createDocumentFragment()

説明
指定したタグを含むドキュメント・フラグメント・ノードを作成します。

構文
```
public XMLDocumentFragment createDocumentFragment();
```

戻り値
作成されたドキュメント・フラグメント・ノード。

createElement()

説明
指定したタグを含む要素ノードを作成して戻します。

構文
```
public XMLElement createElement( String tag);
```

パラメータ	説明
tag	要素名。

createElementNS()

説明

指定したローカル名、接頭辞および名前空間 URI を含む要素ノードを作成して戻します。

構文

```
public XMLElement createElementNS( String localName,
                                   String prefix,
                                   String namespaceURI);
```

パラメータ	説明
localName	要素名。
prefix	要素の接頭辞。
namespaceURI	要素の名前空間。

createEntityReference()

説明

指定したタグを含む実体参照ノードを作成して戻します。

構文

```
public XMLEntityReference createEntityReference( String tag);
```

パラメータ	説明
tag	ノード名。

createProcessingInstruction()

説明

指定したタグおよびテキストを含む処理命令ノードを作成して戻します。

構文

```
public XMLPI createProcessingInstruction( String tag,
                                          String text);
```

パラメータ	説明
tag	ノード名。
text	ノードに対応するテキスト。

createTextNode()

説明

指定したテキストを含む Text ノードを作成して戻します。

構文

```
public XMLText createTextNode( String text);
```

パラメータ	説明
text	ノードに対応するテキスト。

oraxml クラス

oraxml の説明

このクラスは、XML ファイルを検証するためのコマンドライン・インタフェースを提供します。

表 1-6 oraxml コマンドライン・インタフェース

コマンド	説明
-help	ヘルプ・メッセージを出力します。
-version	バージョン番号を出力します。
-novalidate	入力ファイルを解析して、整形式かどうかを確認します。
-dtd	DTD 検証を使用して、入力ファイルを検証します。
-schema	スキーマ検証を使用して、入力ファイルを検証します。
-log <logfile>	エラーおよびログを出力ファイルに書き込みます。
-comp	入力 XML ファイルを圧縮します。
-decomp	圧縮した入力ファイルを解凍します。
-enc	入力ファイルのエンコーディングを出力します。
-warning	警告を表示します。

oraxml の構文

```
public class oraxml extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.parser.v2.oraxml
```

oraxml のメソッド

oraxml()

構文

```
public oraxml();
```

main()

構文

```
public static void main( String[] args);
```

SAXAttrList クラス

SAXAttrList の説明

このクラスは、SAX AttributeList インタフェースを実装し、名前空間をサポートします。名前空間のサポートが必要なアプリケーションは、Oracle のパーサー・クラスが戻すいずれかの属性リストを明示的に SAXAttrList に割り当てることによって、次のメソッドを使用できます。また、Attributes（SAX 2.0）インタフェースも実装します。

このインタフェースによって、次の 3 つの方法で属性リストにアクセスできます。

- 属性のインデックス
- 名前空間の修飾名
- （接頭辞付き）修飾名

このインタフェースは、現在は使用できない SAX1 インタフェースのかわりに使用できます。SAX1 インタフェースには、名前空間のサポートがありません。このインタフェースは、名前空間のサポートの他に、getIndex メソッドも追加します。

リスト内の属性順序は指定されておらず、実装ごとに異なります。

SAXAttrList の構文

```
public class SAXAttrList
```

oracle.xml.parser.v2.SAXAttrList

SAXAttrList のメソッド

表 1-7 SAXAttrList のメソッドの概要

メソッド	説明
SAXAttrList()（1-42 ページ）	クラス・コンストラクタです。
addAttr()（1-42 ページ）	属性を親要素ノードに追加します。
getExpandedName()（1-43 ページ）	リスト内の任意の位置の属性の拡張名を戻します。
getIndex()（1-44 ページ）	属性のインデックスを戻します。
getLength()（1-44 ページ）	リスト内の属性数を戻します。
getLocalName()（1-44 ページ）	インデックスが付いた任意の属性のローカル名を戻します。

表 1-7 SAXAttrList のメソッドの概要（続き）

メソッド	説明
getPrefix()（1-45 ページ）	リスト内の任意の位置の属性の名前空間接頭辞を戻します。
getQName()（1-45 ページ）	インデックスが付いた任意の属性の修飾名を戻します。
getType()（1-46 ページ）	属性の型を戻します。
getURI()（1-47 ページ）	インデックスが付いた任意の属性の名前空間 URI を戻します。
getValue()（1-47 ページ）	属性値を文字列として戻します。
reset()（1-48 ページ）	SAXAttrList をリセットします。

SAXAttrList()

説明
クラス・コンストラクタです。

構文
```
public  SAXAttrList(int elems)
```

addAttr()

説明
属性を親要素ノードに追加します。次の表に、オプションを示します。

構文	説明
public void addAttr(String pfx, String lname, String tag, String value, boolean spec, int type);	接頭辞、ローカル名、修飾名、値、指定されたフラグおよび属性の型を使用して、属性を追加します。

構文	説明
public void addAttr(String pfx, String lname, String tag, String value, boolean spec, int type, String nmsp);	接頭辞、ローカル名、修飾名、値、指定されたフラグ、属性の型および名前空間を使用して、属性を追加します。

パラメータ	説明
pfx	属性の接頭辞。
lname	属性のローカル名。
tag	属性の修飾名。
value	属性値。
spec	指定されたフラグ。
type	属性の型。
nmsp	名前空間。

getExpandedName()

説明

リスト内の任意の位置の属性の拡張名を戻します。

構文

```
public String getExpandedName( int i);
```

パラメータ	説明
i	リスト内の属性のインデックス番号。

getIndex()

説明

属性のインデックスを戻します。属性のインデックスがリスト内に存在しない場合は、-1 を戻します。次の表に、オプションを示します。

構文	説明
public int getIndex(String qName);	修飾名を持つ任意の属性のインデックスを戻します。
public int getIndex(String uri, String localName);	名前空間 URI およびローカル名を持つ任意の属性のインデックスを戻します。

パラメータ	説明
qName	（接頭辞付き）修飾名。
uri	名前空間 URI（名前に名前空間 URI が含まれない場合は空の文字列）。
localName	属性のローカル名。

getLength()

説明

このリスト内の属性数を戻します。SAX パーサーは、属性が宣言または指定された順序にかかわらず、属性を任意の順序で提供できます。属性数は、0（ゼロ）の場合もあります。

構文

```
public int getLength();
```

getLocalName()

説明

インデックスが付いた任意の属性のローカル名を戻します。名前空間処理が実行されていない場合は空の文字列、インデックスが範囲外の場合は null を戻します。

構文

```
public String getLocalName( int index);
```

パラメータ	説明
index	属性のインデックス（0（ゼロ）から開始)。

getPrefix()

説明

リスト内の任意の位置の属性の名前空間接頭辞を戻します。

構文

```
public String getPrefix( int index);
```

パラメータ	説明
index	属性のインデックス（0（ゼロ）から開始)。

パラメータ

i - リスト内の属性のインデックス。

getQName()

説明

インデックスが付いた任意の属性の XML 1.0 の修飾名を戻します。修飾名が使用できない場合は空の文字列、インデックスが範囲外の場合は null を戻します。

構文

```
public String getQName( int index);
```

パラメータ	説明
index	属性のインデックス（0（ゼロ）から開始)。

getType()

説明

属性の型を戻します。属性の型は、「CDATA」、「ID」、「IDREF」、「IDREFS」、「NMTOKEN」、「NMTOKENS」、「ENTITY」、「ENTITIES」または「NOTATION」の文字列（常に大文字）のいずれかです。パーサーが属性宣言を読み込んでいない場合、またはパーサーが属性の型をレポートしない場合は、XML 1.0 勧告（3.3.3「Attribute-Value Normalization」）に示されているとおり、「CDATA」値を戻す必要があります。表記法ではない列挙された属性については、パーサーが「NMTOKEN」として型をレポートします。次の表に、オプションを示します。

構文	説明
public String getType(int index);	インデックスが付いた任意の属性の型を戻します。インデックスが範囲外の場合は、null を戻します。
public String getType(String qName);	修飾名を持つ任意の属性の型を戻します。属性がリスト内に存在しない場合または修飾名が使用できない場合は、null を戻します。
public String getType(String uri, String localName);	名前空間を持つ任意の属性の型を戻します。属性がリスト内に存在しない場合または名前空間処理が実行されていない場合は、null を戻します。

パラメータ	説明
index	属性のインデックス（0（ゼロ）から開始）。
qName	XML 1.0 の修飾名。
uri	名前空間 URI（名前に名前空間 URI が含まれない場合は空の文字列）。
localName	属性のローカル名。

getURI()

説明

インデックスが付いた任意の属性の名前空間 URI を戻します。名前空間 URI が使用できない場合は空の文字列、インデックスが範囲外の場合は null を戻します。

構文

```
public String getURI( int index);
```

パラメータ	説明
index	属性のインデックス（0（ゼロ）から開始)。

getValue()

説明

属性値を文字列として戻します。インデックスが範囲外の場合は null を戻します。属性値がトークンのリスト（IDREFS、ENTITIES または NMTOKENS）の場合、それらのトークンは、各トークンが単一の空白で区切られた単一の文字列に連結されます。次の表に、オプションを示します。

構文	説明
public String getValue(int index);	インデックスが付いた任意の属性の値を戻します。
public String getValue(String qName);	修飾名を持つ任意の属性の値を戻します。
public String getValue(String uri, String localName);	名前空間を持つ任意の属性の値を戻します。

パラメータ	説明
index	属性のインデックス（0（ゼロ）から開始）。
qName	XML 1.0 の修飾名。
uri	名前空間 URI（名前に名前空間 URI が含まれない場合は空の文字列）。
localName	属性のローカル名。

reset()

説明

SAXAttrList をリセットします。

構文

```
public void reset();
```

SAXParser クラス

SAXParser の説明

このクラスは、World Wide Web Consortium（W3C）勧告に準拠した、eXtensible Markup Language（XML）1.0 の SAX パーサーを実装します。アプリケーションは、SAX ハンドラを登録して、様々なパーサー・イベントの通知を受け取ることができます。

XMLReader は、XML パーサーの SAX2 ドライバによって実装の必要があるインタフェースです。このインタフェースによって、アプリケーションが、パーサーの機能およびプロパティを設定および問合せしたり、ドキュメント処理用のイベント・ハンドラを登録したり、ドキュメントの解析を開始することができます。

すべての SAX インタフェースが同期していると想定されます。解析が完了するまで、解析メソッドは戻されません。また、リーダーは、イベント・ハンドラのコールバックが戻るまで待ってから、次のイベントをレポートする必要があります。

このインタフェースは、（現在は使用できない）SAX 1.0 パーサー・インタフェースのかわりに使用できます。XMLReader インタフェースには、SAX 1.0 パーサー・インタフェースに対する次の 2 つの重要な拡張が含まれています。

- 機能およびプロパティを問い合せたり、設定する標準の方法の追加
- 多くのより高レベルの XML 標準に必要な名前空間サポートの追加

SAXParser の構文

```
public class SAXParser
```

`oracle.xml.parser.v2.SAXParser`

SAXParser のメソッド

表 1-8 SAXParser のメソッドの概要

メソッド	説明
SAXParser()（1-50 ページ）	新しいパーサー・オブジェクトを作成します。
getContentHandler()（1-50 ページ）	現行のコンテンツ・ハンドラを戻します。
getDTDHandler()（1-50 ページ）	現行の DTD ハンドラを戻します。
getFeature()（1-51 ページ）	機能の現在の状態（`true` または `false`）を戻します。
getProperty()（1-51 ページ）	プロパティの値を戻します。

表 1-8 SAXParser のメソッドの概要（続き）

メソッド	説明
setContentHandler()（1-52 ページ）	コンテンツ・イベント・ハンドラを登録します。
setDTDHandler()（1-52 ページ）	DTD イベント・ハンドラを登録します。
setFeature()（1-53 ページ）	機能の状態を設定します。
setProperty()（1-54 ページ）	プロパティの値を設定します。

SAXParser()

説明

新しいパーサー・オブジェクトを作成します。

構文

```
public  SAXParser()
```

getContentHandler()

説明

現行のコンテンツ・ハンドラを戻します。コンテンツ・ハンドラが登録されていない場合は null を戻します。

構文

```
public org.xml.sax.ContentHandler getContentHandler();
```

getDTDHandler()

説明

現行の DTD ハンドラを戻します。DTD ハンドラが登録されていない場合は null を戻します。

構文

```
public org.xml.sax.DTDHandler getDTDHandler();
```

getFeature()

説明

機能の現在の状態（true または false）を戻します。機能名は、任意の完全修飾 URI です。XMLReader は、機能名は認識できますが、その値を戻すことはできません。これは、特に SAX1 パーサー用のアダプタの場合に当てはまります。このアダプタは、基礎となるパーサーが検証を実行しているか、または外部エンティティを拡張しているかを認識できません。一部の機能値は、解析前、解析中、解析後などの特定のコンテキストでのみ使用できます。次の例外が発生します。

- org.xml.sax.SAXNotRecognizedException（XMLReader が機能名を認識しない場合）
- org.xml.sax.SAXNotSupportedException（XMLReader が機能名は認識するが、値は判断できない場合）

機能 http://www.xml.org/sax/features/validation は、そのバイナリ入力値のため、DTD 検証のみを制御します。値 true は、DTD 検証を true に設定します。この機能は、XML Schema に基づく検証の制御には使用できません。

独自の URI に組み込まれた名前を使用して、独自の機能を自由に作成できます。

構文

```
public boolean getFeature( String name);
```

パラメータ	説明
name	機能名。

getProperty()

説明

プロパティの値を戻します。プロパティ名は、任意の完全修飾 URI です。XMLReader は、プロパティ名は認識できますが、その状態を戻すことはできません。これは、特に SAX1 パーサー用のアダプタの場合に当てはまります。SAX2 に初期コア・セットが含まれていますが、XMLReader は、特定のプロパティ名を認識する必要はありません。一部のプロパティ値は、解析前、解析中、解析後などの特定のコンテキストでのみ使用できます。独自の URI に組み込まれた名前を使用して、独自のプロパティを自由に作成できます。次の例外が発生します。

- org.xml.sax.SAXNotRecognizedException（XMLReader が機能名を認識しない場合）

- org.xml.sax.SAXNotSupportedException（XMLReader が機能名は認識するが、値は判別できない場合）

構文

```
public Object getProperty( String name);
```

パラメータ	説明
name	プロパティ名（完全修飾 URI）。

setContentHandler()

説明

コンテンツ・イベント・ハンドラを登録します。アプリケーションがコンテンツ・ハンドラを登録しない場合、SAX パーサーがレポートしたすべてのコンテンツ・イベントは暗黙的に無視されます。アプリケーションは、解析中に新しいハンドラまたは異なるハンドラを登録する場合があります。SAX パーサーは、すぐにその新しいハンドラの使用を開始する必要があります。ハンドラの引数が null の場合は、java.lang.NullPointerException が発生します。

構文

```
public void setContentHandler( org.xml.sax.ContentHandler handler);
```

パラメータ	説明
handler	コンテンツ・ハンドラ。

setDTDHandler()

説明

DTD イベント・ハンドラを登録します。アプリケーションが DTD ハンドラを登録しない場合、SAX パーサーがレポートしたすべての DTD イベントは暗黙的に無視されます。アプリケーションは、解析中に新しいハンドラまたは異なるハンドラを登録する場合があります。SAX パーサーは、すぐにその新しいハンドラの使用を開始する必要があります。ハンドラの引数が null の場合は、java.lang.NullPointerException が発生します。

構文

```
public void setDTDHandler( org.xml.sax.DTDHandler handler);
```

パラメータ	説明
handler	DTD ハンドラ。

setFeature()

説明

機能の状態を設定します。機能名は、任意の完全修飾 URI です。XMLReader は、機能名は認識できますが、その値を設定することはできません。これは、特に SAX1 パーサー用のアダプタの場合に当てはまります。このアダプタは、たとえば、基礎となるパーサーが検証を実行するかどうかに影響を与えることはできません。

一部の機能値は、解析前、解析中、解析後などの特定のコンテキストでのみ不変または可変となります。機能「http::/www.xml.org/sax/features/validation」は、そのバイナリ入力値のため、DTD 検証のみを制御します。値 `true` は、DTD 検証を `true` に設定します。この機能は、XML Schema に基づく検証の制御には使用できません。次の例外が発生します。

- `org.xml.sax.SAXNotRecognizedException`（XMLReader が機能名を認識しない場合）
- `org.xml.sax.SAXNotSupportedException`（XMLReader が機能名は認識するが、リクエストされた値を設定できない場合）

構文

```
public void setFeature( String name,
                        boolean state);
```

パラメータ	説明
name	機能名（完全修飾 URI）。
state	機能のリクエストされた状態（`true` または `false`）。

setProperty()

説明

プロパティの値を設定します。プロパティ名は、任意の完全修飾 URI です。XMLReader は、プロパティ名は認識できますが、その値を設定することはできません。これは、特に SAX1 パーサー用のアダプタの場合に当てはまります。SAX2 にコア・セットが含まれていますが、XMLReader は、特定のプロパティ名の設定を認識する必要はありません。一部のプロパティ値は、解析前、解析中、解析後などの特定のコンテキストでのみ不変または可変となります。このメソッドは、拡張ハンドラ設定用の標準メカニズムでもあります。次の例外が発生します。

- `org.xml.sax.SAXNotRecognizedException`（XMLReader がプロパティ名を認識しない場合）
- `org.xml.sax.SAXNotSupportedException`（XMLReader がプロパティ名は認識するが、リクエストされた値を設定できない場合）

構文

```
public void setProperty( String name,
                         Object value);
```

パラメータ	説明
name	プロパティ名（完全修飾 URI）。
value	リクエストされたプロパティの値。

XMLParseException クラス

XMLParseException の説明

このクラスは、XML 文書の処理中に、解析例外が発生したことを示します。

XMLParseException の構文

```
public class XMLParseException

oracle.xml.parser.v2.XMLParseException
```

XMLParseException のフィールド

表 1-9 XMLParseException のフィールド

フィールド	構文	説明
ERROR	public static final int ERROR	致命的でないエラーのコード。
FATAL_ERROR	public static final int FATAL_ERROR	致命的エラーのコード。
WARNING	public static final int WARNING	警告のコード。

XMLParseException のメソッド

表 1-10 XMLParseException のメソッドの概要

メソッド	説明
XMLParseException() (1-56 ページ)	XMLParseException クラスのコンストラクタです。
formatErrorMessage() (1-57 ページ)	指定されたインデックスのエラー・メッセージをフォーマットします。
getColumnNumber() (1-57 ページ)	指定されたインデックスで、エラーの発生した列番号を戻します。
getException() (1-57 ページ)	指定されたインデックスで発生した例外を戻します。
getLineNumber() (1-58 ページ)	指定されたインデックスで、エラーの発生した行番号を戻します。
getMessage() (1-58 ページ)	指定されたインデックスで、エラーのメッセージを戻します。
getMessageType() (1-58 ページ)	指定されたインデックスで、メッセージの型を戻します。

表 1-10 XMLParseException のメソッドの概要（続き）

メソッド	説明
getNumMessages()（1-59 ページ）	解析中に検出されたエラーおよび警告の合計数を戻します。
getPublicId()（1-59 ページ）	指定されたインデックスで、エラーが発生した場合に入力する公開識別子を戻します。
getSystemId()（1-59 ページ）	特定のインデックスで、エラーが発生した場合に入力するシステム識別子を戻します。

XMLParseException()

説明

クラス・コンストラクタです。

構文

```
public  XMLParseException( String mesg,
                           String pubId,
                           String sysId,
                           int line,
                           int col,
                           int type);
```

パラメータ	説明
mesg	メッセージ。
pubId	公開識別子。
sysId	システム識別子。
line	行。
col	列。
type	型。

formatErrorMessage()

説明

指定されたインデックスで、エラー・メッセージをフォーマットします。

構文

```
public String formatErrorMessage( int i);
```

パラメータ	説明
i	インデックス。

getColumnNumber()

説明

指定されたインデックスで、エラーの発生した列番号を戻します。

構文

```
public int getColumnNumber(int i);
```

パラメータ	説明
i	インデックス。

getException()

説明

指定されたインデックスで発生した例外（存在する場合）を戻します。

構文

```
public java.lang.Exception getException(int i);
```

パラメータ	説明
i	インデックス。

getLineNumber()

説明
指定されたインデックスで、エラーの発生した行番号を戻します。

構文
```
public int getLineNumber(int i);
```

パラメータ	説明
i	インデックス。

getMessage()

説明
指定されたインデックスで、エラーのメッセージを戻します。

構文
```
public String getMessage(int i)
```

パラメータ	説明
i	インデックス。

getMessageType()

説明
指定されたインデックスで、メッセージの型を戻します。

構文
```
public int getMessageType(int i)
```

パラメータ	説明
i	インデックス。

getNumMessages()

説明

解析中に検出されたエラーおよび警告の合計数を戻します。

構文

```
public int getNumMessages();
```

getPublicId()

説明

指定されたインデックスで、エラーが発生した場合に入力する公開識別子を戻します。

構文

```
public String getPublicId(int i);
```

パラメータ	説明
i	インデックス。

getSystemId()

説明

指定されたインデックスで、エラーが発生した場合に入力するシステム識別子を戻します。

構文

```
public String getSystemId(int i);
```

パラメータ	説明
i	インデックス。

XMLParser クラス

XMLParser の説明

このクラスは、DOMParser クラスおよび SAXParser クラスのベース・クラスとして機能します。World Wide Web Consortium（W3C）勧告に準拠した、eXtensible Markup Language（XML）1.0 文書を解析するメソッドが含まれます。このクラスは、インスタンス化できません（アプリケーションは、要件に応じて DOM パーサーまたは SAX パーサーを使用します）。

XMLParse の構文

```
public abstract class XMLParser
```

oracle.xml.parser.v2.XMLParser

XMLParser のフィールド

表 1-11 XMLParser のフィールド

フィールド	構文	説明
BASE_URL	public static final java.lang.String BASE_URL	エンティティの解析で使用するベース URL。setBaseURL() のかわりに使用できます。URL オブジェクトである必要があります。
DTD_OBJECT	public static final java.lang.String DTD_OBJECT	検証に使用する DTD オブジェクト。XMLParser.setDoctype() のかわりに使用できます。
SCHEMA_ OBJECT	public static final java.lang.String SCHEMA_OBJECT	検証に使用するスキーマ・オブジェクト。XMLParser.setXMLSchema() のかわりに使用できます。
STANDALONE	public static final java.lang.String STANDALONE	入力ファイルがスタンドアロンかどうかを表すプロパティ。true の場合、DTD は取得されません。
USE_DTD_ ONLY_FOR_ VALIDATION	public static final java.lang.String USE_DTD_ONLY_FOR_ VALIDATION	true の場合、DTD オブジェクトは検証のみに使用され、パーサー・ドキュメントに追加されません（Boolean.TRUE または Boolean.FALSE）。このプロパティ / 属性は、setDoctype をコールして、固定 DTD を使用する場合にのみ有効です。

XMLParser のメソッド

表 1-12 XMLParser のメソッドの概要

メソッド	説明
getAttribute()（1-62 ページ）	実際の実装の属性の値を取得します。
getBaseURL()（1-62 ページ）	ベース URL を戻します。
getEntityResolver()（1-62 ページ）	エンティティ・リゾルバを戻します。
getErrorHandler()（1-63 ページ）	エラー・ハンドラを戻します。
getReleaseVersion()（1-63 ページ）	バージョン番号を戻します。
getValidationModeValue()（1-63 ページ）	検証モードの値を戻します。
getXMLProperty()（1-64 ページ）	プロパティの値を戻します。
isXMLPropertyReadOnly()（1-64 ページ）	指定されたプロパティが読込み専用の場合は、`true` を戻します。
isXMLPropertySupported()（1-64 ページ）	指定されたプロパティがサポートされている場合は、`true` を戻します。
parse()（1-65 ページ）	XML を解析します。
reset()（1-66 ページ）	パーサーの状態をリセットします。
setAttribute()（1-66 ページ）	実際の実装に特定の属性を設定します。
setBaseURL()（1-66 ページ）	外部エンティティおよび DTD をロードするためのベース URL を設定します。
setDoctype()（1-67 ページ）	DTD を設定します。
setEntityResolver()（1-67 ページ）	エンティティ・リゾルバを設定します。
setErrorHandler()（1-68 ページ）	エラー・イベント・ハンドラを登録します。
setLocale()（1-68 ページ）	エラー・レポートのロケールを設定します。
setPreserveWhitespace()（1-69 ページ）	空白保持モードを設定します。

表 1-12 XMLParser のメソッドの概要（続き）

メソッド	説明
setValidationMode() (1-69 ページ)	検証モードを設定します。
setXMLProperty() (1-69 ページ)	XML プロパティを設定して戻します。
setXMLSchema() (1-70 ページ)	インスタンス・ドキュメントを検証するための XML Schema を設定します。

getAttribute()

説明

実際の実装の特定の属性の値を取得します。実際の実装がその属性を認識しない場合は、`IllegalArgumentException` が発生します。

構文

```
public java.lang.Object getAttribute( String name);
```

パラメータ	説明
name	属性名。

getBaseURL()

説明

ベース URL を戻します。

構文

```
public java.net.URL getBaseURL();
```

getEntityResolver()

説明

現行のエンティティ・リゾルバを戻します。

構文

```
public org.xml.sax.EntityResolver getEntityResolver();
```

戻り値

現行のエンティティ・リゾルバ（エンティティ・リゾルバが登録されていない場合は null）。

getErrorHandler()

説明

現行のエラー・ハンドラを戻します。エラー・ハンドラが登録されていない場合は、null を戻します。

構文

```
public org.xml.sax.ErrorHandler getErrorHandler();
```

getReleaseVersion()

説明

Oracle XML Parser のバージョン番号を戻します。

構文

```
public static String getReleaseVersion();
```

getValidationModeValue()

説明

次に示す検証モードの値を戻します。

- 0（ゼロ）: XML Parser が NONVALIDATING の場合
- 1: XML Parser が PARTIAL_VALIDATION の場合
- 2: XML Parser が DTD_VALIDATION の場合
- 3: XML Parser が SCHEMA_VALIDATION の場合

構文

```
public int getValidationModeValue();
```

getXMLProperty()

説明

プロパティの値を戻します。プロパティが存在し、サポートされている場合は、プロパティが戻されます。それ以外の場合は、null が戻されます。

構文

```
public java.lang.Object getXMLProperty( String name);
```

パラメータ	説明
name	プロパティ名。

isXMLPropertyReadOnly()

説明

指定されたプロパティが読込み専用の場合は、true を戻します。そのプロパティがサポートされていない場合は、true を戻します。

構文

```
public boolean isXMLPropertyReadOnly( String name);
```

パラメータ	説明
name	プロパティ名。

isXMLPropertySupported()

説明

指定されたプロパティがサポートされている場合は、true を戻します。

構文

```
public boolean isXMLPropertySupported( String name)
```

パラメータ	説明
name	プロパティ名。

parse()

説明

XML を解析します。次の例外が発生します。

- `XMLParseException`（構文エラーまたは他のエラーが発生した場合）
- `SAXException`（`SAXException` またはそのサブクラスのインスタンスで、別の例外が隠されている可能性あり）
- `IOException`（I/O エラーが発生した場合）

次の表に、オプションを示します。

構文	説明
public void parse(org.xml.sax.InputSource in);	指定された入力ソースから XML を解析します。
public final void parse(java.io.InputStream in);	指定された入力ストリームから XML を解析します。
public final void parse(java.io.Reader in);	指定されたリーダーから XML を解析します。
public void parse(String in);	指定された URL から XML を解析します。
public final void parse(java.net.URL url);	指定された URL が指す XML 文書を解析し、それに対応する XML 文書の階層を作成します。

パラメータ	説明
in	解析する XML データを含むソース。
url	解析する XML 文書を指す URL。

reset()

説明

パーサーの状態をリセットします。

構文

```
public void reset();
```

setAttribute()

説明

実際の実装に特定の属性を設定します。実際の実装がその属性を認識しない場合は、IllegalArgumentException が発生します。

構文

```
public void setAttribute( String name,
                          Object value);
```

パラメータ	説明
name	属性名。
value	属性値。

setBaseURL()

説明

外部エンティティおよび DTD をロードするためのベース URL を設定します。このメソッドは、XML 文書を解析する際に、parse() を使用する場合にコールする必要があります。

構文

```
public void setBaseURL( java.net.URL url);
```

パラメータ	説明
url	ベース URL。

setDoctype()

説明

DTD を設定します。

構文

```
public void setDoctype( DTD dtd);
```

パラメータ	説明
dtd	解析中に設定および使用する DTD。

setEntityResolver()

説明

エンティティ・リゾルバを登録します。リゾルバの引数が `null` の場合は、`NullPointerException` が発生します。

アプリケーションがエンティティ・リゾルバを登録しない場合、XMLReader は独自のデフォルトの解決を実行します。アプリケーションは、解析中に新しいリゾルバまたは異なるリゾルバを登録する場合があります。SAX パーサーは、すぐにその新しいリゾルバの使用を開始する必要があります。

構文

```
public void setEntityResolver( org.xml.sax.EntityResolver resolver);
```

パラメータ	説明
resolver	エンティティ・リゾルバ。

setErrorHandler()

説明

エラー・イベント・ハンドラを登録します。ハンドラの引数が null の場合は、java.lang.NullPointerException が発生します。

アプリケーションがエラー・ハンドラを登録しない場合、SAX パーサーがレポートしたすべてのエラー・イベントは暗黙的に無視されます。ただし、通常の処理が続行されない場合があります。予期しないエラーを回避するため、すべての SAX アプリケーションにエラー・ハンドラを実装することをお薦めします。アプリケーションは、解析中に新しいハンドラまたは異なるハンドラを登録する場合があります。SAX パーサーは、すぐにその新しいハンドラの使用を開始する必要があります。

構文

```
public void setErrorHandler( org.xml.sax.ErrorHandler handler);
```

パラメータ	説明
handler	エラー・ハンドラ。

setLocale()

説明

エラー・レポートのロケールを設定します。エラーが発生した場合は、SAXException が発生します。

構文

```
public void setLocale( java.util.Locale locale);
```

パラメータ	説明
locale	設定するロケール。

setPreserveWhitespace()

説明

空白保持モードを設定します。

構文

```
public void setPreserveWhitespace( boolean flag);
```

パラメータ	説明
flag	保持モード。

setValidationMode()

説明

検証モードを設定します。このメソッドは、パーサーの検証モードを、NONVALIDATING、PARTIAL_VALIDATION、DTD_VALIDATION および SCHEMA_VALIDATION という 4 つのタイプのいずれかに設定します。

構文

```
public void setValidationMode(int valMode)
```

パラメータ	説明
valMode	パーサーに対して設定する必要がある検証モードのタイプを判別します。

setXMLProperty()

説明

XML プロパティを設定して戻します。正常に設定された場合は、設定されたプロパティの値が戻されます。プロパティが読込み専用で、設定できない場合またはサポートされていない場合は、null が戻されます。

構文

```
public java.lang.Object setXMLProperty( String name,
                                        Object value);
```

パラメータ	説明
name	プロパティ名。
value	プロパティの値。

setXMLSchema()

説明

インスタンス・ドキュメントの検証のための XML Schema を設定します。

構文

```
public void setXMLSchema( java.lang.Object schema);
```

パラメータ	説明
schema	XMLSchema オブジェクト。

XMLToken クラス

XMLToken の説明

このクラスは、XMLToken 用の基本インタフェースです。トークン化機能を持つすべての XMLParser アプリケーションは、このインタフェースを実装する必要があります。このインタフェースは、`XMLTokenizer` の `setTokenHandler()` メソッドを使用して登録する必要があります。

XMLToken ハンドラが `null` でない場合、パーサーは、トークンが登録および検出されるたびに XMLToken のコールバック・メソッド `token()` をコールします。トークン化実行中、パーサーは、ドキュメントの検証、内部および外部のエンティティの挿入および読込みを行いません。XMLToken ハンドラが `null` の場合、パーサーは通常の解析を行います。

XMLToken のリクエストは、XMLParser のメソッド `setToken()` を使用して登録（オン / オフ）されます。リクエストは、（コールバック・メソッド内から）解析中に登録されている場合もあります。

XMLToken は、`XMLToken` インタフェースでパブリック定数として定義されます。これらの定数は、W3C による XML 構文仕様の XML 構文変数に対応しています。

XMLToken の構文

```
public interface XMLToken
```

XMLToken のフィールド

表 1-13 XMLToken のフィールド

フィールド	構文	説明
AttListDecl	public static final int AttListDecl	AttListDecl ::= '<' '!' 'ATTLIST' S Name AttDef* S? '>'
AttName	public static final int AttName	AttName ::= Name
Attribute	public static final int Attribute	Attribute ::= AttName Eq AttValue
AttValue	public static final int AttValue	AttValue ::= '"' ([^<&"] | Reference)* '"' | "'" ([^<&'] | Reference)* "'"

表 1-13 XMLToken のフィールド（続き）

フィールド	構文	説明
CDSect	public static final int CDSect	CDSect ::= CDStart CData CDEnd CDStart ::= '<' '!' '[CDATA[' CData ::= (Char* - (Char* ']]>' Char*)) CDEnd ::= ']]>'
CharData	public static final int CharData	CharData ::= [^<&]* - ([^<&]* ']]>' [^<&]*)
Comment	public static final int Comment	Comment ::= '<' '!' '--' ((Char - '-') | ('-' (Char - '-')))* '-->'
DTDName	public static final int DTDName	DTDName ::= name
ElemDeclName	public static final int ElemDeclName	ElemDeclName ::= name
elementdecl	public static final int elementdecl	elementdecl ::= '<' '!ELEMENT' S ElemDeclName S contentspec S? '>'
EmptyElemTag	public static final int EmptyElemTag	EmptyElemTag ::= '<' STagName (S Attribute)* S? '/' '>'
EntityDecl	public static final int EntityDecl	EntityDecl ::= '<' '!' ENTITY' S EntityDeclName S EntityDef S? '>' | '<' '!' ENTITY' S '%' S EntityDeclName S PEDef S? '>' EntityDef ::= EntityValue | (ExternalID NDataDecl?) PEDef ::= EntityValue | ExternalID
EntityDeclName	public static final int EntityDeclName	EntityValue ::= '"' ([^%&"] | PEReference | Reference)* '"' | "'" ([^%&'] | PEReference | Reference)* "'"
EntityValue	public static final int EntityValue	EntityDeclName ::= Name
ETag	public static final int ETag	ETag ::= '<' '/' ETagName S? '>'
ETagName	public static final int ETagName	ETagName ::= Name

表 1-13 XMLToken のフィールド（続き）

フィールド	構文	説明
ExternalID	public static final int ExternalID	ExternalID ::= 'SYSTEM' S SystemLiteral | 'PUBLIC' S PubidLiteral S SystemLiteral
NotationDecl	public static final int NotationDecl	NotationDecl ::= '<' '!NOTATION' S Name S (ExternalID | PublicID) S? '>'
PI	public static final int PI	PI ::= '<' '?' PITarget (S (Char* - (Char* '?>' Char*)))? '?' '>'
PITarget	public static final int PITarget	PITarget ::= Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))
Reference	public static final int Reference	Reference ::= EntityRef | CharRef | PEReference EntityRef ::= '&' Name ';' PEReference ::= '%' Name ';' CharRef ::= '&#' [0-9]+ ';' | '&#x' [0-9a-fA-F]+ ';'
STag	public static final int STag	STag ::= '<' STagName (S Attribute)* S? '>'
STagName	public static final int STagName	STagName ::= Name
TextDecl	public static final int TextDecl	TextDecl ::= '<' '?' 'xml' VersionInfo? EncodingDecl S? '?>'
XMLDecl	public static final int XMLDecl	XMLDecl ::= '<' '?' 'xml' VersionInfo EncodingDecl? SDDecl? S? '?' '>'

XMLToken のメソッド

token()

説明

XMLToken およびそれに対応する値を受け取ります。これは、インタフェースのコールバック・メソッドです。

構文

```
public void token( int token,
                   String value);
```

パラメータ	説明
token	インタフェースで指定された XMLToken 定数。
value	解析対象テキストの対応する部分文字列。

XMLTokenizer クラス

XMLTokenizer の説明

このクラスは、World Wide Web Consortium（W3C）勧告に準拠した、eXtensible Markup Language（XML）1.0 のパーサーを実装します。

XMLTokenizer の構文

```
public class XMLTokenizer
```

oracle.xml.parser.v2.XMLTokenizer

XMLTokenizer のメソッド

表 1-14 XMLTokenizer のメソッドの概要

メソッド	説明
XMLTokenizer()（1-76 ページ）	新しい Tokenizer オブジェクトを作成します。
parseDocument()（1-76 ページ）	ドキュメントを解析します。
setErrorHandler()（1-76 ページ）	新しいエラー・イベント・ハンドラを登録します。
setErrorStream()（1-77 ページ）	エラー用の出力ストリームを登録します。
setToken()（1-77 ページ）	XML トークン化機能の新しいトークンを登録します。
setTokenHandler()（1-77 ページ）	新しい XML トークン化機能イベント・ハンドラを登録します。
tokenize()（1-78 ページ）	XML をトークン化します。

XMLTokenizer()

説明

新しい Tokenizer オブジェクトを作成します。次の表に、オプションを示します。

構文	説明
public XMLTokenizer();	新しい Tokenizer オブジェクトを作成します。
public XMLTokenizer(XMLToken handler);	ハンドラから新しい Tokenizer オブジェクトを作成します。

パラメータ	説明
handler	ハンドラ。

parseDocument()

説明

ドキュメントを解析します。

構文

```
public void parseDocument();
```

setErrorHandler()

説明

新しいエラー・イベント・ハンドラを登録します。これによって、以前に設定したすべてのエラー処理設定が置換されます。

構文

```
public void setErrorHandler(org.xml.sax.ErrorHandler handler)
```

パラメータ	説明
handler	登録対象のハンドラ。

setErrorStream()

説明
エラー用の出力ストリームを登録します。

構文
```
public void setErrorStream( java.io.OutputStream out);
```

パラメータ	説明
out	登録対象のエラー・ストリーム。

setToken()

説明
XML トークン化機能の新しいトークンを登録します。

構文
```
public void setToken( int token,
                      boolean val);
```

パラメータ	説明
token	設定対象の XMLToken。

setTokenHandler()

説明
新しい XML トークン化機能イベント・ハンドラを登録します。

構文
```
public void setTokenHandler( XMLToken handler);
```

パラメータ	説明
handler	登録対象の XMLToken。

tokenize()

説明

XML をトークン化します。次の例外が発生します。

- `XMLParseException`（構文エラーまたは他のエラーが発生した場合）
- `SAXException`（`SAXException` またはそのサブクラスのインスタンスで、別の例外が隠されている可能性あり）
- `IOException`（I/O エラーが発生した場合）

次の表に、オプションを示します。

構文	説明
public final void tokenize(org.xml.sax.InputSource in);	指定された入力ソースから XML をトークン化します。
public final void tokenize(java.io.InputStream in);	指定された入力ストリームから XML をトークン化します。
public final void tokenize(java.io.Reader r);	指定されたリーダーから XML をトークン化します。
public final void tokenize(String in);	文字列から XML をトークン化します。
public final void tokenize(java.net.URL url);	指定された URL が指す XML 文書をトークン化し、それに対応する XML 文書の階層を作成します。

パラメータ	説明
in	解析のソース。
url	解析する XML 文書を指す URL。

NSName クラス

NSName の説明

NSName インタフェースは、orace.xml.util パッケージの一部です。このインタフェースは、要素名および属性名用の名前空間をサポートします。

NSName の構文

```
public interface oracle.xml.util.NSName
```

NSName のメソッド

表 1-15 NSName のメソッドの概要

メソッド	説明
getExpandedName()（1-79 ページ）	NSName の解決済完全名を戻します。
getLocalName()（1-80 ページ）	NSName のローカル名を戻します。
getNamespace()（1-80 ページ）	NSName の解決済名前空間を戻します。
getPrefix()（1-80 ページ）	NSName の接頭辞を戻します。
getQualifiedName()（1-80 ページ）	修飾名を戻します。

getExpandedName()

説明

NSName の解決済完全名を戻します。

構文

```
public  String  getExpandedName();
```

getLocalName()

説明

NSName のローカル名を戻します。

構文

```
public String getLocalName();
```

getNamespace()

説明

NSName の解決済名前空間を戻します。

構文

```
public String getNamespace()
```

getPrefix()

説明

NSName の接頭辞を戻します。

構文

```
public String getPrefix();
```

getQualifiedName()

説明

修飾名を戻します。

構文

```
public String getQualifiedName();
```

XMLError クラス

XMLError の説明

oracle.xml.util パッケージのこのクラスは、エラー・メッセージおよびそれが発生した行番号を保持します。

XMLError の構文

```
public class XMLError
```

oracle.xml.util.XMLError

XMLError のフィールド

表 1-16 oracle.xml.util.XMLError のフィールド

フィールド	構文
col	protected int[] col
errid	protected int[] errid
exp	protected java.lang.Exception[] exp
line	protected int[] line
mesg	protected java.lang.String[] mesg
pubId	protected java.lang.String[] pubId
sysId	protected java.lang.String[] sysId
types	protected int[] types

XMLError のメソッド

表 1-17 XMLError のメソッドの概要

メソッド	説明
XMLError()（1-83 ページ）	XMLError のデフォルトのコンストラクタです。
error()（1-84 ページ）	新しいエラーをベクトルに追加します。
error0()（1-84 ページ）	追加のパラメータを使用せずに、新しいエラーをベクトルに追加します。
error1()（1-85 ページ）	1 つの追加パラメータを使用して、新しいエラーをベクトルに追加します。
error2()（1-85 ページ）	2 つの追加パラメータを使用して、新しいエラーをベクトルに追加します。
error3()（1-86 ページ）	3 つの追加パラメータを使用して、新しいエラーをベクトルに追加します。
flushErrorStream()（1-86 ページ）	すべてのエラーを出力ストリームのデフォルトまたはエラー・ハンドラに対してフラッシュします。
formatErrorMesg()（1-87 ページ）	エラー・メッセージをフォーマットします。
getColumnNumber()（1-87 ページ）	指定されたインデックスで、エラーの発生した列番号を戻します。
getException()（1-87 ページ）	指定されたインデックスで発生した例外（存在する場合）を戻します。
getFirstError()（1-88 ページ）	最初のエラーを戻します。
getLineNumber()（1-88 ページ）	指定されたインデックスで、エラーの発生した行番号を戻します。
getLocator()（1-88 ページ）	登録されたロケータを戻します。
getMessage()（1-89 ページ）	指定されたインデックスで、エラーのメッセージを戻します。
getMessage0()（1-89 ページ）	パラメータを使用せずにエラー・メッセージを戻します。
getMessage1()（1-90 ページ）	1 つのパラメータを使用してエラー・メッセージを戻します。
getMessage2()（1-90 ページ）	2 つのパラメータを使用してエラー・メッセージを戻します。
getMessage3()（1-91 ページ）	3 つのパラメータを使用してエラー・メッセージを戻します。
getMessage4()（1-91 ページ）	4 つのパラメータを使用してエラー・メッセージを戻します。
getMessage5()（1-92 ページ）	5 つのパラメータを使用してエラー・メッセージを戻します。

表 1-17 XMLError のメソッドの概要（続き）

メソッド	説明
getMessageType()（1-93 ページ）	指定されたインデックスで、メッセージの型を戻します。
getNumMessages()（1-93 ページ）	解析中に検出されたエラーおよび警告の合計数を戻します。
getPublicId()（1-93 ページ）	指定されたインデックスで、エラーが発生した場合に入力する公開識別子を戻します。
getSystemId()（1-94 ページ）	指定されたインデックスで、エラーが発生した場合に入力するシステム識別子を戻します。
printErrorListener()（1-94 ページ）	すべての JAXP 1.1 エラーを ErrorListener に対してフラッシュします。ErrorListener が設定されていない場合は、デフォルトで System.err に対してフラッシュします。
reset()（1-94 ページ）	エラー・クラスをリセットします。
setErrorStream()（1-95 ページ）	出力ストリームを登録します。
setException()（1-95 ページ）	例外を登録します。
setLocale()（1-96 ページ）	ロケールを登録します。
setLocator()（1-96 ページ）	ロケータを登録します。
showWarnings()（1-96 ページ）	警告レポートのオン / オフを切り替えます。

XMLError()

説明

XMLError のデフォルトのコンストラクタです。

構文

```
public  XMLError();
```

error()

説明

新しいエラーをベクトルに追加します。次の表に、オプションを示します。

構文	説明
public void error(int id, int type, String msg);	1 つのメッセージを使用して、新しいエラーをベクトルに追加します。
public void error3(int id, int type, String[] params);	1 つのパラメータ配列を使用して、新しいエラーをベクトルに追加します。

パラメータ	説明
id	エラー・メッセージの ID。
type	エラーのタイプ。
MESG	エラー・メッセージ（パラメータなし）。
params	パラメータ配列。

error0()

説明

パラメータを使用せずに、新しいエラーをベクトルに追加します。

構文

```
public void error3( int id,
                    int type);
```

パラメータ	説明
id	エラー・メッセージの ID。
type	エラーのタイプ。

error1()

説明

1 つのパラメータを使用して、新しいエラーをベクトルに追加します。

構文

```
public void error3( int id,
                    int type,
                    String p1);
```

パラメータ	説明
id	エラー・メッセージの ID。
type	エラーのタイプ。
p1	パラメータ 1。

error2()

説明

2 つのパラメータを使用して、新しいエラーをベクトルに追加します。

構文

```
public void error3( int id,
                    int type,
                    String p1,
                    String p2);
```

パラメータ	説明
id	エラー・メッセージの ID。
type	エラーのタイプ。
p1	パラメータ 1。
p2	パラメータ 2。

error3()

説明

3 つのパラメータを使用して、新しいエラーをベクトルに追加します。

構文

```
public void error3( int id,
                    int type,
                    String p1,
                    String p2,
                    String p3)
```

パラメータ	説明
id	エラー・メッセージの ID。
type	エラーのタイプ。
p1	パラメータ 1。
p2	パラメータ 2。
p3	パラメータ 3。

flushErrorStream()

説明

すべてのエラーを出力ストリームのデフォルトまたはエラー・ハンドラに対してフラッシュします。

構文

```
public void flushErrorStream();
```

formatErrorMesg()

説明
エラー・メッセージをフォーマットします。

構文
```
public String formatErrorMesg(int index);
```

パラメータ	説明
index	インデックス。

getColumnNumber()

説明
指定されたインデックスで、エラーの発生した列番号を戻します。

構文
```
public int getColumnNumber(int i);
```

パラメータ	説明
i	インデックス。

getException()

説明
指定されたインデックスで発生した例外（存在する場合）を戻します。

構文
```
public java.lang.Exception getException(int i);
```

パラメータ	説明
i	インデックス。

getFirstError()

説明

最初のエラーを戻します。

構文

```
public int getFirstError();
```

getLineNumber()

説明

指定されたインデックスで、エラーの発生した行番号を戻します。

構文

```
public int getLineNumber(int i);
```

パラメータ	説明
i	インデックス。

getLocator()

説明

登録されたロケータを戻します。

構文

```
public org.xml.sax.Locator getLocator();
```

getMessage()

説明

指定されたインデックスで、エラーのメッセージを戻します。次の表に、オプションを示します。

構文	説明
public String getMessage(int i);	指定されたインデックスで、エラーのメッセージを戻します。
public String getMessage(int errId, String[] params);	1 つの配列にパッケージされた 6 つ以上のパラメータを使用して、エラー・メッセージを戻します。

パラメータ	説明
i	インデックス。
errId	エラー ID。
params	パラメータ配列。

getMessage0()

説明

パラメータを使用せずにエラー・メッセージを戻します。

構文

```
public String getMessage1( int errId);
```

パラメータ	説明
errId	エラー ID。

getMessage1()

説明

1 つのパラメータを使用してエラー・メッセージを戻します。

構文

```
public String getMessage1( int errId,
                           String a1);
```

パラメータ	説明
errId	エラー ID。
a1	パラメータ 1。

getMessage2()

説明

2 つのパラメータを使用してエラー・メッセージを戻します。

構文

```
public String getMessage3(
     int errId,
     String a1,
     String a2);
```

パラメータ	説明
errId	エラー ID。
a1	パラメータ 1。
a2	パラメータ 2。

getMessage3()

説明

3 つのパラメータを使用してエラー・メッセージを戻します。

構文

```
public String getMessage3(
     int errId,
     String a1,
     String a2,
     String a3);
```

パラメータ	説明
errId	エラー ID。
a1	パラメータ 1。
a2	パラメータ 2。
a3	パラメータ 3。

getMessage4()

説明

4 つのパラメータを使用してエラー・メッセージを戻します。

構文

```
public String getMessage4(
     int errId,
     String a1,
     String a2,
     String a3,
     String a4);
```

パラメータ	説明
errId	エラー ID。
a1	パラメータ 1。
a2	パラメータ 2。
a3	パラメータ 3。
a4	パラメータ 4。

getMessage5()

説明

5 つのパラメータを使用してエラー・メッセージを戻します。

構文

```
public String getMessage5(
     int errId,
     String a1,
     String a2,
     String a3,
     String a4,
     String a5);
```

パラメータ	説明
errId	エラー ID。
a1	パラメータ 1。
a2	パラメータ 2。
a3	パラメータ 3。
a4	パラメータ 4。
a5	パラメータ 5。

getMessageType()

説明

指定されたインデックスで、メッセージの型を戻します。

構文

```
public int getMessageType(int i);
```

パラメータ	説明
i	インデックス。

getNumMessages()

説明

解析中に検出されたエラーおよび警告の合計数を戻します。

構文

```
public int getNumMessages()
```

getPublicId()

説明

指定されたインデックスで、エラーが発生した場合に入力する公開識別子を戻します。

構文

```
public String getPublicId( int i);
```

パラメータ	説明
i	インデックス。

getSystemId()

説明

指定されたインデックスで、エラーが発生した場合に入力するシステム識別子を戻します。

構文

```
public String getSystemId(int i);
```

パラメータ	説明
i	インデックス。

printErrorListener()

説明

すべての JAXP 1.1 エラーを ErrorListener に対してフラッシュします。ErrorListener が設定されていない場合は、デフォルトで、System.err に対してフラッシュします。

構文

```
public void printErrorListener();
```

reset()

説明

エラー・クラスをリセットします。

構文

```
public void reset();
```

setErrorStream()

説明

エラー・レポートの出力ストリームを設定します。次の表に、オプションを示します。

構文	説明
public void setErrorStream(java.io.OutputStream out);	エラー・レポートの出力ストリームを設定します。出力ストリームの初期化中にエラーが発生した場合は、IOException が発生します。
public void setErrorStream(java.io.OutputStream out, String enc);	エラー・レポートの出力ストリームおよびそのエンコーディングを設定します。出力ストリームの初期化中にエラーが発生した場合は、IOException が発生します。
public void setErrorStream(java.io.PrintWriter out);	エラー・レポート用のプリント・ライターを設定します。

パラメータ	説明
out	エラーおよび警告の出力先。
enc	出力ストリームのエンコーディング。

setException()

説明

例外を登録します。

構文

```
public void setException( java.lang.Exception exp);
```

パラメータ	説明
exp	発生した最後の例外。

setLocale()

説明
エラー・レポートのロケールを登録します。

構文
```
public void setLocale( java.util.Locale locale);
```

パラメータ	説明
locale	エラー・レポートのロケール。

setLocator()

説明
ロケータを登録します。

構文
```
public void setLocator( org.xml.sax.Locator locator);
```

パラメータ	説明
locator	行 / 列 / システム識別子 / 公開識別子の情報を取得するためのロケータ。

showWarnings()

説明
警告レポートのオン / オフを切り替えます。

構文
```
public void showWarnings(boolean flag);
```

パラメータ	説明
flag	警告のレポートを制御します。

XMLException クラス

XMLException の説明

oracle.xml.util パッケージ内のこのクラスは、XML 文書の処理中に解析例外が発生したことを示します。

XMLException の構文

```
public class XMLException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--oracle.xml.util.XMLException
```

XMLException の実装済インタフェース

```
java.io.Serializable
```

XMLException のフィールド

表 1-18 XMLException のフィールド

フィールド	構文	説明
ERROR	public static final int ERROR	致命的でないエラーのコード。
FATAL_ERROR	public static final int FATAL_ERROR	致命的エラーのコード。
WARNING	public static final int WARNING	警告のコード。

XMLException のメソッド

表 1-19 oracle.xml.util.XMLException のメソッドの概要

メソッド	説明
XMLException()（1-99 ページ）	XMLException を生成します。
formatErrorMessage()（1-100 ページ）	指定されたインデックスで、エラーのメッセージを戻します。
getColumnNumber()（1-100 ページ）	指定されたインデックスで、エラーの発生した列番号を戻します。
getException()（1-100 ページ）	指定されたインデックスで発生した例外（存在する場合）を戻します。
getLineNumber()（1-101 ページ）	指定されたインデックスで、エラーの発生した行番号を戻します。
getMessage()（1-101 ページ）	指定されたインデックスで、エラーのメッセージを戻します。
getMessageType()（1-101 ページ）	指定されたインデックスで、メッセージの型を戻します。
getNumMessages()（1-102 ページ）	解析中に検出されたエラーおよび警告の合計数を戻します。
getPublicId()（1-102 ページ）	指定されたインデックスで、エラーが発生した場合に入力する公開識別子を戻します。
getSystemId()（1-102 ページ）	指定されたインデックスで、エラーが発生した場合に入力するシステム識別子を戻します。
getXMLError()（1-103 ページ）	XMLException 内の XMLError オブジェクトを戻します。
printStackTrace()（1-103 ページ）	XMLException の `Throwable` およびそのバックトレースを出力します。
setException()（1-103 ページ）	基礎となる例外（存在する場合）を設定します。
toString()（1-104 ページ）	任意の埋込み例外を戻します。

XMLException()

説明

XMLException を生成します。次の表に、オプションを示します。

構文	説明
public XMLException(String mesg, String pubId, String sysId, int line, int col, int type);	メッセージ、公開識別子、システム識別子、行、列および型から XMLException を生成します。
public XMLException(XMLError err, java.lang.Exception e);	エラーおよび例外から XMLException を生成します。
public XMLException(XMLError err, int firsterr);	発生したエラーおよび最初のエラーから XMLException を生成します。
public XMLException(XMLError err, int firsterr, java.lang.Exception e);	発生したエラー、例外および最初のエラーから XMLException を生成します。

パラメータ	説明
mesg	メッセージ。
pubId	公開識別子。
sysId	システム識別子。
line	行。
col	列。
type	型。
err	エラー。

パラメータ	説明
e	例外。
firsterr	最初のエラー。

formatErrorMessage()

説明
指定されたインデックスで、エラーのメッセージを戻します。

構文
```
public String formatErrorMessage( int i);
```

パラメータ	説明
i	インデックス。

getColumnNumber()

説明
指定されたインデックスで、エラーの発生した列番号を戻します。

構文
```
public int getColumnNumber( int i);
```

パラメータ	説明
i	インデックス。

getException()

説明
指定されたインデックスで発生した例外（存在する場合）を戻します。

構文
```
public java.lang.Exception getException( int i);
```

パラメータ	説明
i	インデックス。

getLineNumber()

説明
指定されたインデックスで、エラーの発生した行番号を戻します。

構文
```
public int getLineNumber(int i);
```

パラメータ	説明
i	インデックス。

getMessage()

説明
指定されたインデックスで、エラーのメッセージを戻します。

構文
```
public String getMessage(int i).
```

パラメータ	説明
i	インデックス。

getMessageType()

説明
指定されたインデックスで、メッセージの型を戻します。

構文
```
public int getMessageType(int i);
```

パラメータ	説明
i	インデックス。

getNumMessages()

説明
解析中に検出されたエラーおよび警告の合計数を戻します。

構文
```
public int getNumMessages();
```

getPublicId()

説明
指定されたインデックスで、エラーが発生した場合に入力する公開識別子を戻します。

構文
```
public String getPublicId(int i);
```

パラメータ	説明
i	インデックス。

getSystemId()

説明
指定されたインデックスで、エラーが発生した場合に入力するシステム識別子を戻します。

構文
```
public String getSystemId(int i);
```

パラメータ	説明
i	インデックス。

getXMLError()

説明

XMLException 内の XMLError オブジェクトを取得します。

構文

```
public XMLError getXMLError();
```

printStackTrace()

説明

Throwable およびそのバックトレースを出力します。次の表に、オプションを示します。

構文	説明
public void printStackTrace();	XMLException の Throwable およびそのバックトレースを標準エラー・ストリームに出力します。
public void printStackTrace(java.io.PrintStream s);	XMLException の Throwable およびそのバックトレースを指定された出力ストリームに出力します。
public void printStackTrace(java.io.PrintWriter s);	XMLException の Throwable およびそのバックトレースを指定されたプリント・ライターに出力します。

パラメータ	説明
s	スタック・トレースの出力。

setException()

説明

基礎となる例外（存在する場合）を設定します。

構文

```
public void setException(java.lang.Exception ex);
```

パラメータ	説明
ex	例外。

toString()

説明

任意の埋込み例外を戻します。

構文

```
public String toString();
```

2

ドキュメント・オブジェクト・モデル（DOM）

XML 文書は、開始タグと終了タグの間の情報で構成されるノードを含むツリーとして表示できます。ドキュメント・オブジェクト・モデル（DOM）というこのツリー表現は、XML Parser によるドキュメントの解析時にメモリー内に作成されます。DOM API は、oracle.xml.parser.v2 パッケージに含まれています。

この章では、このツリーにアクセスし、ナビゲートするための API について説明します。この章の内容は次のとおりです。

- XMLEntityReference クラス
- XMLNode クラス
- XMLNotation クラス
- XMLNSNode クラス
- XMLOutputStream クラス
- XMLPI クラス
- XMLPrintDriver クラス
- XMLRangeException クラス
- XMLText クラス

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

NSResolver インタフェース

NSResolver の構文

```
public interface NSResolver
```

NSResolver の説明

このインタフェースは、名前空間の解決をサポートします。

NSResolver クラスの実装

```
XMLElement
```

NSResolver のメソッド

resolveNamespacePrefix()

説明

指定された名前空間の接頭辞の有効範囲内で、名前空間の定義を検索して戻します。接頭辞を解決できない場合は、`null` を戻します。

構文

```
public String resolveNamespacePrefix( String prefix);
```

パラメータ	説明
prefix	解決する名前空間の接頭辞。

PrintDriver インタフェース

PrintDriver の説明

このインタフェースは、DOM ツリーとして表現される XML 文書の出力に使用されるメソッドを定義します。

PrintDriver の構文

```
public interface PrintDriver
```

PrintDriver クラスの実装

```
XMLPrintDriver
```

PrintDriver のメソッド

表 2-1 PrintDriver のメソッドの概要

メソッド	説明
close()（2-5 ページ）	出力ストリームまたはプリント・ライターをクローズします。
flush()（2-5 ページ）	出力ストリームまたはプリント・ライターをフラッシュします。
printAttribute()（2-5 ページ）	XMLAttr ノードを出力します。
printAttributeNodes()（2-6 ページ）	XMLElement の属性ごとに出力メソッドをコールします。
printCDATASection()（2-6 ページ）	XMLCDATA ノードを出力します。
printChildNodes()（2-6 ページ）	XMLNode の子ごとに出力メソッドをコールします。
printComment()（2-7 ページ）	XMLComment ノードを出力します。
printDoctype()（2-7 ページ）	DTD を出力します。
printDocument()（2-7 ページ）	XMLDocument を出力します。
printDocumentFragment()（2-8 ページ）	空の XMLDocumentFragment オブジェクトを出力します。
printElement()（2-8 ページ）	XMLElement を出力します。
printEntityReference()（2-8 ページ）	XMLEntityReference ノードを出力します。

表 2-1 PrintDriver のメソッドの概要（続き）

メソッド	説明
printProcessingInstruction()（2-9 ページ）	XMLPI ノードを出力します。
printTextNode()（2-9 ページ）	XMLText ノードを出力します。
setEncoding()（2-9 ページ）	出力ドライバのエンコーディングを設定します。

close()

説明

出力ストリームまたはプリント・ライターをクローズします。

構文

```
public void close();
```

flush()

説明

出力ストリームまたはプリント・ライターをフラッシュします。

構文

```
public void flush();
```

printAttribute()

説明

XMLAttr ノードを出力します。

構文

```
public void printAttribute( XMLAttr attr);
```

パラメータ	説明
attr	XMLAttr ノード。

printAttributeNodes()

説明

XMLElement の属性ごとに出力メソッドをコールします。

構文

```
public void printAttributeNodes( XMLElement elem);
```

パラメータ	説明
elem	属性が出力される要素。

printCDATASection()

説明

XMLCDATA ノードを出力します。

構文

```
public void printCDATASection( XMLCDATA cdata);
```

パラメータ	説明
cdata	XMLCDATA ノード。

printChildNodes()

説明

XMLNode の子ごとに出力メソッドをコールします。

構文

```
public void printChildNodes( XMLNode node);
```

パラメータ	説明
node	子が出力されるノード。

printComment()

説明

XMLComment ノードを出力します。

構文

```
public void printComment( XMLComment comment);
```

パラメータ	説明
comment	コメント・ノード。

printDoctype()

説明

DTD を出力します。

構文

```
public void printDoctype( DTD dtd);
```

パラメータ	説明
dtd	出力する DTD。

printDocument()

説明

XMLDocument を出力します。

構文

```
public void printDocument( XMLDocument doc);
```

パラメータ	説明
doc	出力するドキュメント。

printDocumentFragment()

説明

空の XMLDocumentFragment オブジェクトを出力します。

構文

```
public void printDocumentFragment( XMLDocumentFragment dfrag);
```

パラメータ	説明
dfrag	出力するドキュメント・フラグメント。

printElement()

説明

XMLElement を出力します。

構文

```
public void printElement( XMLElement elem);
```

パラメータ	説明
elem	出力する要素。

printEntityReference()

説明

XMLEntityReference ノードを出力します。

構文

```
public void printEntityReference( XMLEntityReference eref);
```

パラメータ	説明
eref	出力する XMLEntityReference ノード。

printProcessingInstruction()

説明

XMLPI ノードを出力します。

構文

```
public void printProcessingInstruction( XMLPI pi);
```

パラメータ	説明
pi	出力する XMLPI ノード。

printTextNode()

説明

XMLText ノードを出力します。

構文

```
public void printTextNode( XMLText text);
```

パラメータ	説明
text	Text ノード。

setEncoding()

説明

出力ドライバのエンコーディングを設定します。

構文

```
public void setEncoding( String enc);
```

パラメータ	説明
enc	出力するドキュメントのエンコーディング。

AttrDecl クラス

AttrDecl の説明

このクラスは、Document Type Definition 内の属性リストに宣言されている各属性の情報を保持します。

AttrDecl の構文

```
public class AttrDecl implements java.io.Externalizable
```

oracle.xml.parser.v2.AttrDecl

AttrDecl の実装済インタフェース

```
java.io.Externalizable, java.io.Serializable
```

AttrDecl のフィールド

表 2-2 AttrDecl のフィールド

フィールド	構文	説明
CDATA	public static final int CDATA	属性の型 - 文字型 - CDATA
DEFAULT	public static final int DEFAULT	属性が存在するかどうかを表す型 - デフォルト
ENTITIES	public static final int ENTITIES	属性の型 - トークン型 - 複数のエンティティ
ENTITY	public static final int ENTITY	属性の型 - トークン型 - エンティティ
ENUMERATION	public static final int ENUMERATION	属性の型 - 列挙型 - 列挙
FIXED	public static final int FIXED	属性が存在するかどうかを表す型 - 固定
ID	public static final int ID	属性の型 - トークン型 - ID
IDREF	public static final int IDREF	属性の型 - トークン型 - ID リファレンス
IDREFS	public static final int IDREFS	属性の型 - トークン型 - 複数の ID リファレンス
IMPLIED	public static final int IMPLIED	属性が存在するかどうかを表す型 - 暗黙
NMTOKEN	public static final int NMTOKEN	属性の型 - トークン型 - 名前トークン

表 2-2 AttrDecl のフィールド（続き）

フィールド	構文	説明
NMTOKENS	public static final int NMTOKENS	属性の型 - トークン型 - 複数の名前トークン
NOTATION	public static final int NOTATION	属性の型 - 列挙型 - 表記法
REQUIRED	public static final int REQUIRED	属性が存在するかどうかを表す型 - 必須

AttrDecl のメソッド

表 2-3 AttrDecl のメソッドの概要

メソッド	説明
AttrDecl()（2-12 ページ）	デフォルトのコンストラクタです。
getAttrPresence()（2-12 ページ）	属性が存在するかどうかを表す型を戻します。
getAttrType()（2-12 ページ）	属性の型を戻します。
getDefaultValue()（2-12 ページ）	属性のデフォルト値を戻します。
getEnumerationValues()（2-13 ページ）	属性値のリストを戻します。
getNodeName()（2-13 ページ）	AttrDecl の名前を戻します。
getNodeType()（2-13 ページ）	実際の実装オブジェクトの型を表すコードを戻します。
readExternal()（2-13 ページ）	writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
typeToString()（2-14 ページ）	属性の型の文字列表現を戻します。
writeExternal()（2-14 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

AttrDecl()

説明

デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。ノードを非シリアル化して、シリアル化 / 圧縮したストリームから DOM ノードを作成するには、オブジェクトのハンドルを作成する必要があります。

構文

```
public static final int REQUIRED public  AttrDecl();
```

getAttrPresence()

説明

属性が存在するかどうかを表す型を戻します。

構文

```
public int getAttrPresence();
```

getAttrType()

説明

属性の型を戻します。

構文

```
public int getAttrType();
```

getDefaultValue()

説明

属性のデフォルト値を戻します。

構文

```
public String getDefaultValue();
```

getEnumerationValues()

説明

属性値のリストを戻します。

構文

```
public java.util.Vector getEnumerationValues();
```

getNodeName()

説明

AttrDecl の名前を戻します。

構文

```
public String getNodeName();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

readExternal()

説明

writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- `IOException`（入力ストリームの読込み中にエラーが発生した場合）
- `ClassNotFoundException`（クラスが検出されない場）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

typeToString()

説明

属性の型の文字列表現を戻します。

構文

```
public static String typeToString( int type);
```

パラメータ	説明
type	属性の型の文字列表現。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。シリアル化 / 圧縮したストリームの書込み中に例外が発生した場合は、IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	シリアル化 / 圧縮したストリームの書込みに使用する ObjectOutput ストリーム。

DTD クラス

DTD の説明

このクラスは、DOM DocumentType インタフェースを実装し、XML 文書の Document Type Definition 情報を保持します。

DTD の構文

```
public class DTD implements java.io.Externalizable
```

oracle.xml.parser.v2.DTD

DTD の実装済インタフェース

java.io.Externalizable, java.io.Serializable

DTD のメソッド

表 2-4 DTD のメソッドの概要

メソッド	説明
DTD()（2-16 ページ）	デフォルトのコンストラクタです。
findElementDecl()（2-17 ページ）	指定されたタグ名に対する要素宣言を検索して戻します。
findEntity()（2-17 ページ）	DTD から名前付きエンティティを検索して戻します。エンティティが検出されない場合は、null を戻します。
findNotation()（2-18 ページ）	DTD から名前表記法を取得します。名前表記法が検出されない場合は、null を戻します。
getChildNodes()（2-18 ページ）	ノードのすべての子ノードを含む NodeList を戻します。
getElementDecls()（2-18 ページ）	DTD の要素宣言を含む名前付きノード・マップを戻します。
getEntities()（2-19 ページ）	DTD で宣言されている外部および内部の汎用エンティティを含む名前付きノード・マップを戻します。
getInternalSubset()（2-19 ページ）	DTD の内部サブセットを戻します。
getName()（2-19 ページ）	DTD の名前（DOCTYPE キーワードの直後の名前）を戻します。

表 2-4 DTD のメソッドの概要（続き）

メソッド	説明
getNodeName()（2-19 ページ）	DTD の名前（DOCTYPE キーワードの直後の名前）を戻します。
getNodeType()（2-20 ページ）	実際の実装オブジェクトの型を表すコードを戻します。
getNotations()（2-20 ページ）	DTD で宣言されている表記法を含む名前付きノード・マップを戻します。
getOwnerImplementation()（2-20 ページ）	DTD 実装の所有者を戻します。
getPublicId()（2-20 ページ）	DTD に対応する公開識別子（指定されている場合）を戻します。
getRootTag()（2-21 ページ）	DTD に対するルート・タグを戻します。
getSystemId()（2-21 ページ）	DTD に対応するシステム識別子（指定されている場合）を戻します。システム識別子が指定されていない場合は、`null` になります。
hasChildNodes()（2-21 ページ）	ノードが子ノードを持っているかどうかを判別します。DTD では XMLNode にオーバーライド・メソッドを含めることができないため、常に `false` を戻します。
normalize()（2-21 ページ）	DTD を正規化します。
printExternalDTD()（2-22 ページ）	ドキュメントのコンテンツを指定された出力先に書き込みます。
readExternal()（2-22 ページ）	writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
setRootTag()（2-23 ページ）	DTD に対するルート・タグを設定します。
writeExternal()（2-23 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

DTD()

説明

デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。ノードを非シリアル化して、シリアル化 / 圧縮したストリームから DOM ノードを作成するには、オブジェクトのハンドルを作成する必要があります。

構文

```
public  DTD();
```

findElementDecl()

説明

指定されたタグ名に対する要素宣言を検索して戻します。

構文

```
public final ElementDecl findElementDecl( String name);
```

パラメータ	説明
name	タグ名。

findEntity()

説明

DTD から名前付きエンティティを検索して戻します。エンティティが検出されない場合は、null を戻します。

構文

```
public final org.w3c.dom.Entity findEntity( String n,
                                            boolean par);
```

パラメータ	説明
n	エンティティ名。
par	エンティティがパラメータ・エンティティであるかどうかを示すブール値。

findNotation()

説明

DTD から名前表記法を取得します。名前表記法が検出されない場合は、null を戻します。

構文

```
public final org.w3c.dom.Notation findNotation( String name);
```

パラメータ	説明
name	表記法名。

getChildNodes()

説明

ノードのすべての子を含む NodeList を戻します。子が存在しない場合は、ノードを含まない NodeList になります。戻された NodeList の内容は、作成される基となったノード・オブジェクトの子ノードに変更があった場合、NodeList アクセッサが戻すノードにすぐに反映されるという点などからみて「最新」のもので、ノード内容の静的スナップショットではありません。これは、getElementsByTagName メソッドが戻す NodeList を含め、すべての NodeList に当てはまります。

構文

```
public org.w3c.dom.NodeList getChildNodes();
```

getElementDecls()

説明

DTD の要素宣言を含む名前付きノード・マップを戻します。このマップ内のすべてのノードは、ElementDecl オブジェクト（DTD 内の要素宣言）です。DOM レベル 1 では要素宣言の編集がサポートされないため、ElementDecl は変更できません。

構文

```
public org.w3c.dom.NamedNodeMap getElementDecls();
```

getEntities()

説明

DTD で宣言されている外部および内部の汎用エンティティを含む名前付きノード・マップを戻します。複製は廃棄されます。たとえば、<!DOCTYPE ex SYSTEM "ex.dtd" [<!ENTITY foo "foo"> <!ENTITY bar "bar"> <!ENTITY % baz "baz">]> <ex/> の場合、インタフェースは foo および bar へのアクセスは提供しますが、baz へのアクセスは提供しません。また、このマップ内のすべてのノードは、Entity インタフェースも実装します。DOM レベル 1 ではエンティティの編集がサポートされないため、エンティティは変更できません。

構文

```
public org.w3c.dom.NamedNodeMap getEntities();
```

getInternalSubset()

説明

DTD の内部サブセットを戻します。

構文

```
public String getInternalSubset();
```

getName()

説明

DTD の名前（DOCTYPE キーワードの直後の名前）を戻します。

構文

```
public String getName();
```

getNodeName()

説明

DTD の名前（DOCTYPE キーワードの直後の名前）を戻します。

構文

```
public String getNodeName();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

getNotations()

説明

DTD で宣言されている表記法を含む名前付きノード・マップを戻します。複製は廃棄されます。また、このマップ内のすべてのノードは、Notation インタフェースも実装します。DOM レベル 1 では表記法の編集がサポートされないため、表記法は変更できません。

構文

```
public org.w3c.dom.NamedNodeMap getNotations();
```

getOwnerImplementation()

説明

DTD 実装の所有者を戻します。

構文

```
public XMLDOMImplementation getOwnerImplementation();
```

getPublicId()

説明

DTD に対応する公開識別子（指定されている場合）を戻します。公開識別子が指定されていない場合は、null になります。

構文

```
public String getPublicId();
```

getRootTag()

説明

DTD に対するルート・タグを戻します。

構文

```
public String getRootTag();
```

getSystemId()

説明

DTD に対応するシステム識別子（指定されている場合）を戻します。システム識別子が指定されていない場合は、null になります。

構文

```
public String getSystemId();
```

hasChildNodes()

説明

ノードが子ノードを持っているかどうかを判別します。DTD は XMLNode にオーバーライド・メソッドを含めることができないため、常に false を戻します。

構文

```
public boolean hasChildNodes();
```

normalize()

説明

DTD を正規化します。

構文

```
public void normalize();
```

printExternalDTD()

説明

ドキュメントのコンテンツを指定された出力先に書き込みます。無効なエンコーディングを指定した場合または別のエラーが発生した場合は、IOException が発生します。次の表に、オプションを示します。

構文	説明
public void printExternalDTD(java.io.OutputStream out);	ドキュメントのコンテンツを指定された出力ストリームに書き込みます。
public void printExternalDTD(java.io.OutputStream out, String enc);	ドキュメントのコンテンツを指定されたエンコーディング済出力ストリームに書き込みます。
public void printExternalDTD(java.io.PrintWriter out);	ドキュメントのコンテンツを指定されたプリント・ライターに書き込みます。

パラメータ	説明
out	出力。
enc	出力に使用するエンコーディング。

readExternal()

説明

writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- IOException（入力ストリームの読込み中にエラーが発生した場合）
- ClassNotFoundException（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

setRootTag()

説明

DTD に対するルート・タグを設定します。

構文

```
public void setRootTag( String root);
```

パラメータ	説明
root	ルート・タグ。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。シリアル化 / 圧縮したストリームの書込み中に例外が発生した場合は、`IOException` が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	シリアル化 / 圧縮したストリームの書込みに使用する ObjectOutput ストリーム。

ElementDecl クラス

ElementDecl の説明

このクラスは、DTD の要素宣言を表します。

ElementDecl の構文

```
public class ElementDecl implements java.io.Serializable, java.io.Externalizable
```

oracle.xml.parser.v2.ElementDecl

ElementDecl の実装済インタフェース

```
java.io.Externalizable, java.io.Serializable
```

ElementDecl のフィールド

表 2-5 ElementDecl のフィールド

フィールド	構文	説明
ANY	public static final byte ANY	要素のコンテンツ型 - 子は、すべての要素になることができます。
ASTERISK	public static final int ASTERISK	ContentModelParseTreeNode 型 -「*」ノード（1 つの子を持ちます）。
COMMA	public static final int COMMA	ContentModelParseTreeNode 型 -「,」ノード（2 つの子を持ちます）。
ELEMENT	public static final int ELEMENT	ContentModelParseTreeNode 型 -「リーフ」・ノード（子を持ちません）。
ELEMENT_ DECLARED	public static final int ELEMENT_ DECLARED	DTD の要素宣言を表すノード・フラグ。
ELEMENTS	public static final byte ELEMENTS	要素のコンテンツ型 - 子は、要素になることができます。コンテンツ・モデルを参照してください。
EMPTY	public static final byte EMPTY	要素のコンテンツ型 - 子を持ちません。

表 2-5 ElementDecl のフィールド（続き）

フィールド	構文	説明
ID_ATTR_DECL	public static final int ID_ATTR_DECL	DTD の ID 属性宣言を表すノード・フラグ。
MIXED	public static final byte MIXED	要素のコンテンツ型 - 子は、PCDATA および要素になることができます。コンテンツ・モデルを参照してください。
OR	public static final int OR	ContentModelParseTreeNode 型 -「|」ノード（2 つの子を持ちます）。
PLUS	public static final int PLUS	ContentModelParseTreeNode 型 -「+」ノード（1 つの子を持ちます）。
QMARK	public static final int QMARK	ContentModelParseTreeNode 型 -「?」ノード（1 つの子を持ちます）。

ElementDecl のメソッド

表 2-6 ElementDecl のメソッドの概要

メソッド	説明
ElementDecl()（2-26 ページ）	デフォルトのコンストラクタです。
cloneNode()（2-26 ページ）	ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。
expectedElements()（2-27 ページ）	要素に追加できる要素名の一覧をベクトルで戻します。
findAttrDecl()（2-27 ページ）	属性宣言オブジェクトを戻します。オブジェクトが検出されない場合は、`null` を戻します。
getAttrDecls()（2-28 ページ）	列挙された属性宣言を戻します。
getContentElements()（2-28 ページ）	要素に追加できる要素のベクトルを戻します。
getContentType()（2-28 ページ）	要素のコンテンツ・モデルを戻します。
getNodeName()（2-28 ページ）	ElementDecl の名前を戻します。
getNodeType()（2-29 ページ）	実際の実装オブジェクトの型を表すコードを戻します。

表 2-6 ElementDecl のメソッドの概要

メソッド	説明
getParseTree()（2-29 ページ）	コンテンツ・モデルを解析したツリーのルート・ノードを戻します。
readExternal()（2-29 ページ）	writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
validateContent()（2-30 ページ）	要素ノードの内容を検証します。
writeExternal()（2-30 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

ElementDecl()

説明

デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。ノードを非シリアル化して、シリアル化 / 圧縮したストリームから DOM ノードを作成するには、オブジェクトのハンドルを作成する必要があります。

構文

```
public  ElementDecl();
```

cloneNode()

説明

ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。この複製ノードは親を持ちません（parentNode は null を戻します）。Element を複製すると、デフォルトの属性を表すために XML プロセッサによって生成された属性を含む、すべての属性およびその値がコピーされます。ただし、このメソッドでは、ノードに含まれるテキストは、子の Text ノードに含まれているため、ディープ・クローンでないかぎりコピーされません。他のタイプのノードを複製すると、単にそのノードのコピーが戻されます。

構文

```
public org.w3c.dom.Node cloneNode(boolean deep);
```

パラメータ	説明
deep	true の場合は、指定されたノードの下位サブツリーが再帰的に複製されます。false の場合は、ノードのみが（ノードが Element の場合は、その属性も）複製されます。

expectedElements()

説明

要素に追加できる要素名の一覧をベクトルで戻します。

構文

```
public java.util.Vector expectedElements( org.w3c.dom.Element e);
```

パラメータ	説明
e	要素。

findAttrDecl()

説明

属性宣言オブジェクトを戻します。オブジェクトが検出されない場合は、null を戻します。

構文

```
public final AttrDecl findAttrDecl( String name);
```

パラメータ	説明
name	検索する属性宣言。

getAttrDecls()

説明

列挙された属性宣言を戻します。

構文

```
public org.w3c.dom.NamedNodeMap getAttrDecls();
```

getContentElements()

説明

要素に追加できる要素のベクトルを戻します。

構文

```
public final java.util.Vector getContentElements();
```

getContentType()

説明

要素のコンテンツ・モデルを戻します。

構文

```
public int getContentType();
```

getNodeName()

説明

ElementDecl の名前を戻します。

構文

```
public String getNodeName();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

getParseTree()

説明

コンテンツ・モデルを解析したツリーのルート・ノードを戻します。
Node.getFirstChild() および Node.getLastChild() は、解析ツリー・ブランチを戻します。Node.getNodeType() および Node.getNodeName() は、解析ツリーのノードのタイプおよび名前を戻します。

構文

```
public final org.w3c.dom.Node getParseTree();
```

readExternal()

説明

writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- IOException（入力ストリームの読込み中にエラーが発生した場合）
- ClassNotFoundException（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

validateContent()

説明

要素ノードの内容を検証します。内容が妥当である場合は true、妥当でない場合は false を戻します。

構文

```
public boolean validateContent(org.w3c.dom.Element e);
```

パラメータ	説明
e	検証する要素ノード。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。シリアル化 / 圧縮したストリームの書込み中に例外が発生した場合は、IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	シリアル化 / 圧縮したストリームの書込みに使用する ObjectOutput ストリーム。

XMLAttr クラス

XMLAttr の説明

このクラスは、DOM Attr インタフェースを実装し、要素の各属性情報を保持します。
Attr、NodeFactory および DOMParser.setNodeFactory() も参照してください。

XMLAttr の構文

```
public class XMLAttr implements oracle.xml.parser.v2.NSName, java.io.Externalizable
```

oracle.xml.parser.v2.XMLAttr

XMLAttr の実装済インタフェース

java.io.Externalizable, NSName, oracle.xml.util.NSName, java.io.Serializable

XMLAttr のメソッド

表 2-7 XMLAttr のメソッドの概要

メソッド	説明
addText()（2-32 ページ）	XMLNode にテキストを追加します。
cloneNode()（2-33 ページ）	ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。
getExpandedName()（2-33 ページ）	属性の解決済完全名を戻します。
getLocalName()（2-33 ページ）	属性のローカル名を戻します。
getName()（2-34 ページ）	属性名を戻します。
getNamespaceURI()（2-34 ページ）	属性の名前空間を戻します。
getNextAttribute()（2-34 ページ）	次の属性（存在する場合）を戻します。
getNextSibling()（2-34 ページ）	次の兄弟関係（存在する場合）を戻します。
getNodeType()（2-35 ページ）	実際の実装オブジェクトの型を表すコードを戻します。
getNodeValue()（2-35 ページ）	ノードのタイプに応じて、ノードの値を戻します。
getOwnerElement()（2-35 ページ）	属性を所有する要素を戻します。

表 2-7 XMLAttr のメソッドの概要（続き）

メソッド	説明
getParentNode()（2-35 ページ）	ノードの親を戻します。
getPrefix()（2-36 ページ）	要素の名前空間の接頭辞を戻します。
getPreviousSibling()（2-36 ページ）	前の兄弟関係（存在する場合）を戻します。
getSpecified()（2-36 ページ）	属性が要素で明示的に指定されている場合は `true`、明示的に指定されていない場合は `false` を戻します。
getValue()（2-36 ページ）	属性値を戻します。
readExternal()（2-37 ページ）	writeExternal によって書き込まれた情報をリストアします。
setNodeValue()（2-37 ページ）	ノードのタイプに応じて、ノードの値を設定します。
setValue()（2-38 ページ）	値を設定します。
writeExternal()（2-38 ページ）	バイナリ圧縮ストリームにオブジェクトの状態を保存します。

addText()

説明

XMLNode にテキストを追加します。

構文

```
public XMLNode addText( String str);
```

パラメータ	説明
str	追加するテキスト。

cloneNode()

説明

ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。この複製ノードは親を持ちません（parentNode は null を戻します）。Element を複製すると、デフォルトの属性を表すために XML プロセッサによって生成された属性を含む、すべての属性およびその値がコピーされます。ただし、このメソッドでは、ノードに含まれるテキストは、子の Text ノードに含まれているため、ディープ・クローンでないかぎりコピーされません。他のタイプのノードを複製すると、単にそのノードのコピーが戻されます。

構文

```
public org.w3c.dom.Node cloneNode( boolean deep);
```

パラメータ	説明
deep	true の場合は、指定されたノードの下位サブツリーが再帰的に複製されます。false の場合は、ノードのみが（ノードが Element の場合は、その属性も）複製されます。

getExpandedName()

説明

属性の解決済完全名を戻します。

構文

```
public String getExpandedName();
```

getLocalName()

説明

属性のローカル名を戻します。

構文

```
public String getLocalName();
```

getName()

説明
属性名を戻します。

構文
```
public String getName();
```

getNamespaceURI()

説明
属性の名前空間を戻します。

構文
```
public String getNamespaceURI();
```

getNextAttribute()

説明
次の属性（存在する場合）を戻します。

構文
```
public XMLAttr getNextAttribute();
```

getNextSibling()

説明
次の兄弟関係（存在する場合）を戻します。

構文
```
public org.w3c.dom.Node getNextSibling();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

getNodeValue()

説明

ノードのタイプに応じて、ノードの値を戻します。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- DOMSTRING_SIZE_ERR（実装プラットフォーム上で DOMString 変数に収まらない文字が戻された場合）

構文

```
public String getNodeValue();
```

getOwnerElement()

説明

属性を所有する要素を戻します。

構文

```
public org.w3c.dom.Element getOwnerElement();
```

getParentNode()

説明

ノードの親を戻します。Document、DocumentFragment および Attr 以外のすべてのノードは、親を持ちます。ただし、ノードが作成された直後で、ツリーに追加されていない場合、またはノードがツリーから削除されている場合は、null になります。

構文

```
public org.w3c.dom.Node getParentNode();
```

getPrefix()

説明
要素の名前空間の接頭辞を戻します。

構文
```
public String getPrefix();
```

getPreviousSibling()

説明
前の兄弟関係（存在する場合）を戻します。

構文
```
public org.w3c.dom.Node getPreviousSibling();
```

getSpecified()

説明
属性が要素で明示的に指定されている場合は `true`、明示的に指定されていない場合は `false` を戻します。

構文
```
public boolean getSpecified();
```

getValue()

説明
属性値を戻します。

構文
```
public String getValue();
```

readExternal()

説明

writeExternal によって書き込まれた情報をリストアします。次の例外が発生します。

- IOException（圧縮ストリームの読込み中に例外が発生した場合）
- ClassNotFoundException（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

setNodeValue()

説明

ノードのタイプに応じて、ノードの値を設定します。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- DOMSTRING_SIZE_ERR（実装プラットフォーム上で DOMString 変数に収まらない文字が戻された場合）

構文

```
public void setNodeValue( String nodeValue);
```

パラメータ	説明
nodeValue	設定するノードの値。

setValue()

説明

値を設定します。

構文

```
public void setValue( String val);
```

パラメータ	説明
val	設定する値。

writeExternal()

説明

バイナリ圧縮ストリームにオブジェクトの状態を保存します。圧縮ストリームの書込み中に例外が発生した場合は、IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	圧縮ストリームの書込みに使用する ObjectOutput ストリーム。

XMLCDATA クラス

XMLCDATA の説明

このクラスは、DOM CDATASection インタフェースを実装します。CDATASection、NodeFactory および DOMParser.setNodeFactory(NodeFactory) も参照してください。

XMLCDATA の構文

```
public class XMLCDATA implements java.io.Externalizable
```

oracle.xml.parser.v2.XMLCDATA

XMLCDATA の実装済インタフェース

java.io.Externalizable, java.io.Serializable

XMLCDATA のメソッド

表 2-8 XMLCDATA のメソッドの概要

メソッド	説明
XMLCDATA()（2-40 ページ）	デフォルトのコンストラクタです。
getNodeName()（2-40 ページ）	ノード名を戻します。
getNodeType()（2-40 ページ）	実際の実装オブジェクトの型を表すコードを戻します。
readExternal()（2-40 ページ）	writeExternal() メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
writeExternal()（2-41 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

XMLCDATA()

説明

デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。ノードを非シリアル化して、シリアル化 / 圧縮したストリームから DOM ノードを作成するには、オブジェクトのハンドルを作成する必要があります。

構文

```
public  XMLCDATA();
```

getNodeName()

説明

ノード名を戻します。

構文

```
public String getNodeName();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

readExternal()

説明

`writeExternal()` メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- `IOException`（入力ストリームの読込み中にエラーが発生した場合）
- `ClassNotFoundException`（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。圧縮ストリームの書込み中に例外が発生した場合、IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	圧縮ストリームの書込みに使用する ObjectOutput ストリーム。

XMLComment クラス

XMLComment の説明

このクラスは、DOM Comment インタフェースを実装します。Comment、NodeFactory および DOMParser.setNodeFactory() も参照してください。

XMLComment の構文

```
public class XMLComment implements java.io.Externalizable
```

oracle.xml.parser.v2.XMLComment

XMLComment の実装済インタフェース

java.io.Externalizable, java.io.Serializable

XMLComment のメソッド

表 2-9 XMLComment の概要

メソッド	説明
XMLComment()（2-43 ページ）	デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。
addText()（2-43 ページ）	コメント・テキストを追加します。
getNodeName()（2-43 ページ）	ノード名を戻します。
getNodeType()（2-44 ページ）	実際の実装オブジェクトの型を表すコードを戻します。
readExternal()（2-44 ページ）	writeExternal() メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
reportSAXEvents()（2-44 ページ）	DOM ツリーの SAX イベントをレポートします。
writeExternal()（2-45 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

XMLComment()

説明

デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。ノードを非シリアル化して、シリアル化 / 圧縮したストリームから DOM ノードを作成するには、オブジェクトのハンドルを作成する必要があります。

構文

```
public  XMLComment();
```

addText()

説明

コメント・テキストを追加します。

構文

```
public XMLNode addText( String str);
```

パラメータ	説明
str	コメント・テキスト。

getNodeName()

説明

ノード名を戻します。

構文

```
public String getNodeName();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

readExternal()

説明

writeExternal() メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- IOException（入力ストリームの読込み中にエラーが発生した場合）
- ClassNotFoundException（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

reportSAXEvents()

説明

DOM ツリーの SAX イベントをレポートします。SAXException が発生します。

構文

```
public void reportSAXEvents( org.xml.sax.ContentHandler cntHandler);
```

パラメータ	説明
cntHandler	コンテンツ・ハンドラ。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。圧縮ストリームの書込み中に例外が発生した場合、IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	圧縮ストリームの書込みに使用する ObjectOutput ストリーム。

XMLDeclPI クラス

XMLDeclPI の説明

このクラスは、XML Decl Processing Instruction を実装します。XMLPI クラスも参照してください。

XMLDeclPI の構文

```
public class XMLDeclPI extends oracle.xml.parser.v2.XMLPI implements
java.io.Externalizable

oracle.xml.parser.v2.XMLPI
  |
  +--oracle.xml.parser.v2.XMLDeclPI
```

XMLDeclPI の実装済インタフェース

```
java.io.Externalizable, java.io.Serializable
```

XMLDeclPI のメソッド

表 2-10 XMLDeclPI のメソッドの概要

メソッド	説明
XMLDeclPI()（2-47 ページ）	XMLDeclPI の新しいインスタンスを作成します。
cloneNode()（2-48 ページ）	ノードの複製を戻し、汎用コピー・コンストラクタとして機能します。
getData()（2-48 ページ）	「version=1.0 ....」という完全な文字列を戻します。
getEncoding()（2-48 ページ）	キャラクタ・エンコーディング情報（<?xml ...?> タグに格納されたエンコーディング情報）またはユーザー定義の出力エンコーディング（より近い時点で設定されている場合）を戻します。
getNodeValue()（2-49 ページ）	ノードの値を戻します。
getStandalone()（2-49 ページ）	スタンドアロン情報（<?xml ...?> タグに格納されたスタンドアロン属性）を戻します。
getVersion()（2-49 ページ）	バージョン情報（<?xml ...?> タグに格納されたバージョン番号）を戻します。

表 2-10 XMLDeclPI のメソッドの概要（続き）

メソッド	説明
readExternal()（2-49 ページ）	writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
setEncoding()（2-50 ページ）	出力用のキャラクタ・エンコーディングを設定します。
setStandalone()（2-50 ページ）	<?xml ...?> タグに格納されたスタンドアロン情報を設定します。
setVersion()（2-51 ページ）	<?xml ...?> タグに格納されたバージョン番号を設定します。
writeExternal()（2-51 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

XMLDeclPI()

説明

XMLDeclPI の新しいインスタンスを作成します。次の表に、オプションを示します。

構文	説明
public XMLDeclPI();	XMLDeclPI の新しいインスタンスを作成するデフォルトのコンストラクタです。
public XMLDeclPI(String version, String encoding, String standalone, boolean textDecl)	バージョン、エンコーディング、単独宣言およびテキスト宣言の情報を使用して、XMLDeclP の新しいインスタンスを作成します。

パラメータ	説明
version	新しい XMLDeclPI の作成に使用するバージョン。
encoding	新しい XMLDeclPI の作成に使用するエンコーディング。
standaolone	新しい XMLDeclPI が単独かどうかを指定します。
textDecl	新しい XMLDeclPI のテキスト宣言。

cloneNode()

説明

ノードの複製を戻し、汎用コピー・コンストラクタとして機能します。

構文

```
public org.w3c.dom.Node cloneNode( boolean deep);
```

パラメータ	説明
deep	true の場合は、指定されたノードの下位サブツリーが再帰的に複製されます。false の場合は、ノードのみが（ノードが Element の場合は、その属性も）複製されます。

getData()

説明

「version=1.0 ....」という完全な文字列を戻します。次の DOMException が発生します。

- DOMSTRING_SIZE_ERR: 実装プラットフォーム上で DOMString 変数に収まらない文字が戻された場合に発生します。

構文

```
public String getData();
```

getEncoding()

説明

キャラクタ・エンコーディング情報（<?xml ...?> タグに格納されたエンコーディング情報）またはユーザー定義の出力エンコーディング（より近い時点で設定されている場合）を戻します。

構文

```
public final String getEncoding();
```

getNodeValue()

説明

ノードの値を戻します。次の DOMException が発生します。

- DOMSTRING_SIZE_ERR（実装上で DOMString 変数に収まらない文字が戻された場合）

構文

```
public String getNodeValue();
```

getStandalone()

説明

スタンドアロン情報（<?xml ...?> タグに格納されたスタンドアロン属性）を戻します。

構文

```
public final String getStandalone();
```

getVersion()

説明

バージョン情報（<?xml ...?> タグに格納されたバージョン番号）を戻します。

構文

```
public final String getVersion();
```

readExternal()

説明

writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- IOException（入力ストリームの読込み中にエラーが発生した場合）
- ClassNotFoundException（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

setEncoding()

説明

出力用のキャラクタ・エンコーディングを設定します。最終的には <?xml ...?> タグに格納されたエンコーディングを設定しますが、ドキュメントが保存されるまでは設定しません。ドキュメントがロードされるまでは、このメソッドをコールしないでください。

構文

```
public final void setEncoding( String encoding);
```

パラメータ	説明
encoding	設定するキャラクタ・エンコーディング。

setStandalone()

説明

<?xml ...?> タグに格納されたスタンドアロン情報を設定します。

構文

```
public final boolean setStandalone( String value);
```

パラメータ	説明
value	XMLDeclPI クラスが単独かどうかを指定します。単独の場合は `true` を、単独ではない場合は `false` を指定します。

setVersion()

説明

<?xml ...?> タグに格納されたバージョン番号を設定します。

構文

```
public final void setVersion( String version);
```

パラメータ	説明
version	設定するバージョン情報。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。圧縮ストリームの書込み中に例外が発生した場合、IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	圧縮ストリームの書込みに使用する ObjectOutput ストリーム。

XMLDocument クラス

XMLDocument の説明

このクラスは、DOM Document インタフェースを実装します。XML 文書全体を表し、ドキュメント・オブジェクト・モデル・ツリーのルートとして機能します。各 XML タグは、このツリーのノードまたはリーフを表します。

XML 仕様では、ツリーのルートはコメントと処理命令の任意の組合せで構成されますが、ルート要素は 1 つのみです。補助メソッド `getDocumentElement` が、ルート要素を検索するためのショートカットとして提供されています。

XMLDocument の構文

```
public class XMLDocument implements java.io.Externalizable
```

`oracle.xml.parser.v2.XMLDocument`

XMLDocument の実装済インタフェース

`java.io.Externalizable, java.io.Serializable`

XMLDocument のメソッド

表 2-11 XMLDocument のメソッドの概要

メソッド	説明
XMLDocument()（2-56 ページ）	空のドキュメントを作成します。
addID()（2-56 ページ）	ドキュメントに対応する ID 要素を追加します。
adoptNode()（2-56 ページ）	ドキュメントに、別のドキュメントのノードを採用します。
appendChild()（2-57 ページ）	新しいノードをドキュメントに追加します。
cloneNode()（2-57 ページ）	ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。
createAttribute()（2-58 ページ）	指定された名前の Attr を作成します。
createAttributeNS()（2-58 ページ）	指定された修飾名および名前空間 URI を持つ属性を作成します。
createCDATASection()（2-59 ページ）	指定された文字列を値として持つ CDATASection ノードを作成します。

表 2-11 XMLDocument のメソッドの概要（続き）

メソッド	説明
createComment()（2-59 ページ）	指定された文字列を含む Comment ノードを作成します。
createDocumentFragment()（2-59 ページ）	空の DocumentFragment オブジェクトを作成します。
createElement()（2-60 ページ）	指定された型の要素を作成します。
createElementNS()（2-60 ページ）	指定された修飾名および名前空間 URI の要素を作成します。
createEntityReference()（2-61 ページ）	EntityReference オブジェクトを作成します。
createEvent()（2-61 ページ）	指定された型のイベント・オブジェクトを作成します。
createMutationEvent()（2-61 ページ）	指定された型の変更イベント・オブジェクトを作成します。
createNodeIterator()（2-62 ページ）	指定されたルート、ノード・イテレータの論理ビューに含めるノードのタイプを制御するフラグ、ノードをフィルタ処理するためのフィルタ、実体参照およびその子孫を含めることができるかどうかを判別するフラグを使用して、ノード・イテレータを作成します。
createProcessingInstruction()（2-62 ページ）	指定された名前およびデータ文字列を持つ ProcessingInstruction ノードを作成します。DOMException が発生します。
createRange()（2-63 ページ）	ドキュメントの先頭に開始と終了の境界点を持つ新しいドキュメント範囲オブジェクトを作成します。
createRangeEvent()（2-63 ページ）	指定された型の範囲イベント・オブジェクトを作成します。
createTextNode()（2-63 ページ）	指定された文字列を含む Text ノードを作成します。
createTraversalEvent()（2-64 ページ）	指定された型の横断イベント・オブジェクトを作成します。
createTreeWalker()（2-64 ページ）	指定されたルート、ノード・イテレータの論理ビューに含めるノードのタイプを制御するフラグ、ノードをフィルタ処理するためのフィルタ、実体参照およびその子孫を含めることができるかどうかを判別するフラグを使用して、ノード・イテレータを作成します。
expectedElements()（2-65 ページ）	要素に追加できる要素名の一覧をベクトルで戻します。
getColumnNumber()（2-65 ページ）	列番号デバッグ情報を戻します。
getDebugMode()（2-65 ページ）	デバッグ・フラグを戻します。

表 2-11 XMLDocument のメソッドの概要（続き）

メソッド	説明
getDoctype()（2-66 ページ）	ドキュメントに対応する Document Type Declaration（DTD）を戻します。
getDocumentElement()（2-66 ページ）	ドキュメントのルート要素である子ノードにアクセスします。
getElementById()（2-66 ページ）	elementId に指定された ID を持つ要素を戻します。
getElementsByTagName()（2-67 ページ）	指定されたタグ名を持つすべての要素の NodeList を、ドキュメント・ツリーの先行順走査で検出された順に戻します。
getElementsByTagNameNS()（2-67 ページ）	指定されたローカル名および名前空間 URI を持つすべての要素の NodeList を、ドキュメント・ツリーの先行順走査で検出された順に戻します。
getEncoding()（2-67 ページ）	<?xml ...?> タグに格納されたキャラクタ・エンコーディング情報またはユーザー定義の出力エンコーディング（より近い時点で設定されている場合）を戻します。
getIDHashtable()（2-68 ページ）	XML DOM ツリー内の ID 要素ハッシュテーブルを戻します。
getImplementation()（2-68 ページ）	ドキュメントを処理する DOMImplementation オブジェクトを戻します。
getLineNumber()（2-68 ページ）	行番号デバッグ情報を戻します。
getNodeType()（2-68 ページ）	実際の実装オブジェクトの型を表すコードを戻します。
getOwnerDocument()（2-69 ページ）	ノードに対応する Document オブジェクトを戻します。
getStandalone()（2-69 ページ）	スタンドアロン情報を取得します。これは、<?xml ...?> タグに格納されたスタンドアロン属性です。
getSystemId()（2-69 ページ）	ノードを含むエンティティのシステム識別子を戻します。
getText()（2-69 ページ）	要素に含まれている非マークアップ・テキストを戻します。
getVersion()（2-70 ページ）	<?xml ...?> タグに格納されたバージョン情報を取得します。
importNode()（2-70 ページ）	ドキュメントに、別のドキュメントからノードをインポートします。
insertBefore()（2-71 ページ）	既存の子ノード refChild の前に newChild ノードを挿入します。
print()（2-71 ページ）	ドキュメントのコンテンツを指定された出力先に書き込みます。

表 2-11 XMLDocument のメソッドの概要（続き）

メソッド	説明
printExternalDTD()（2-72 ページ）	ドキュメントのコンテンツを指定された出力ストリームに書き込みます。
readExternal()（2-73 ページ）	writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
removeChild()（2-73 ページ）	子ノードのドキュメント・リストから要素を削除します。
replaceChild()（2-74 ページ）	子のリスト内の子ノード oldChild を newChild で置換し、oldChild ノードを戻します。
reportSAXEvents()（2-74 ページ）	DOM ツリーから SAX イベントをレポートします。
setDoctype()（2-75 ページ）	ドキュメントの DOCTYPE URI を設定します。
setEncoding()（2-75 ページ）	出力用のキャラクタ・エンコーディングを設定します。
setLocale()（2-76 ページ）	エラー・レポートのロケールを設定します。
setNodeContext()（2-76 ページ）	ノードのコンテキストを設定します。
setParsedDoctype()（2-76 ページ）	システム識別子を解析することによって、DOCTYPE オブジェクトを設定します。
setStandalone()（2-77 ページ）	<?xml ...?> タグに格納されたスタンドアロン情報を設定します。
setVersion()（2-77 ページ）	<?xml ...?> タグに格納されたバージョン番号を設定します。
validateElementContent()（2-78 ページ）	要素ノードの内容を検証します。
writeExternal()（2-78 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

XMLDocument()

説明
空のドキュメントを作成します。

構文
```
public  XMLDocument();
```

addID()

説明
ドキュメントに対応する ID 要素を追加します。

構文
```
public void addID( String name,
                   XMLElement e);
```

パラメータ	説明
name	ID 値。
e	ID に対応する XMLElement。

adoptNode()

説明
ドキュメントに、別のドキュメントのノードを採用します。戻されたノードは親を持たず、parentNode は `null` です。ソース・ノードは元のドキュメントから削除されます。次の `DOMException` が発生します。

- NOT_SUPPORTED_ERR（採用するノードのタイプがサポートされていない場合）

構文
```
public org.w3c.dom.Node adoptNode( org.w3c.dom.Node srcNode);
```

パラメータ	説明
srcNode	採用するノード。

appendChild()

説明

新しいノードをドキュメントに追加します。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR（ノードが elem ノードのタイプの子を許可しないタイプである場合）
- WRONG_DOCUMENT_ERR（elem が異なるドキュメントから作成されている場合）

構文

```
public org.w3c.dom.Node appendChild( org.w3c.dom.Node newNode);
```

パラメータ	説明
newNode	追加する新しいノード。

cloneNode()

説明

ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。この複製ノードは親を持ちません（parentNode は null を戻します）。Element を複製すると、デフォルトの属性を表すために XML プロセッサによって生成された属性を含む、すべての属性およびその値がコピーされます。ただし、このメソッドでは、ノードに含まれるテキストは、子の Text ノードに含まれているため、ディープ・クローンでないかぎりコピーされません。他のタイプのノードを複製すると、単にそのノードのコピーが戻されます。

構文

```
public org.w3c.dom.Node cloneNode( boolean deep);
```

パラメータ	説明
deep	true の場合は、指定されたノードの下位サブツリーが再帰的に複製されます。false の場合は、ノードのみが（ノードが Element の場合は、その属性も）複製されます。

createAttribute()

説明

指定された名前の Attr を作成します。この Attr インスタンスは、setAttribute メソッドを使用して、Element に対して設定できます。このメソッドでは、次の DOMException が発生します。

- INVALID_CHARACTER_ERR（指定された名前が無効な文字を含む場合）

構文

```
public org.w3c.dom.Attr createAttribute( String name);
```

パラメータ	説明
name	新しい属性の名前。

createAttributeNS()

説明

指定された修飾名および名前空間 URI を持つ属性を作成します。次の DOMException が発生します。

- INVALID_CHARACTER_ERR（指定された修飾名が無効な文字を含む場合）
- NAMESPACE_ERR（修飾名の形式が正しくない場合、修飾名が接頭辞を持ち、名前空間 URI が null または空の文字列である場合、あるいは修飾名が「xml」という接頭辞を持ち、名前空間 URI が「http://www.w3.org/2000/xmlns/」とは異なる場合）

構文

```
public org.w3c.dom.Attr createAttributeNS( String namespaceURI,
                                           String qualifiedName);
```

パラメータ	説明
namespaceURI	作成する属性または要素の名前空間。
qualifiedName	作成する属性または要素の修飾名。

createCDATASection()

説明

指定された文字列を値として持つ CDATASection ノードを作成します。DOMException が発生します。

構文

```
public org.w3c.dom.CDATASection createCDATASection( String data);
```

パラメータ	説明
data	CDATASection の内容のデータ。

createComment()

説明

指定された文字列を含む Comment ノードを作成します。

構文

```
public org.w3c.dom.Comment createComment( String data);
```

パラメータ	説明
data	ノードのデータ。

createDocumentFragment()

説明

空の DocumentFragment オブジェクトを作成します。

構文

```
public org.w3c.dom.DocumentFragment createDocumentFragment();
```

createElement()

説明

指定された型の要素を作成します。戻されたインスタンスは Element インタフェースを実装するため、戻されたオブジェクトに属性を直接指定できます。次の DOMException が発生します。

- INVALID_CHARACTER_ERR（指定された名前が無効な文字を含む場合）

構文

```
public org.w3c.dom.Element createElement( String tagName);
```

パラメータ	説明
tagName	インスタンス化する要素型の名前。この名前では、大 / 小文字が区別されます。

createElementNS()

説明

指定された修飾名および名前空間 URI の要素を作成します。次の DOMException が発生します。

- INVALID_CHARACTER_ERR（指定された修飾名が無効な文字を含む場合）
- NAMESPACE_ERR（修飾名の形式が正しくない場合、修飾名が接頭辞を持ち、名前空間 URI が null または空の文字列である場合、あるいは修飾名が「xml」という接頭辞を持ち、名前空間 URI が「http://www.w3.org/XML/1998/namespace」とは異なる場合）

構文

```
public org.w3c.dom.Element createElementNS( String namespaceURI,
                                             String qualifiedName);
```

パラメータ	説明
namespaceURI	作成する属性または要素の名前空間。
qualifiedName	作成する属性または要素の修飾名。

createEntityReference()

説明

EntityReference オブジェクトを作成します。次の DOMException が発生します。

- INVALID_CHARACTER_ERR（指定された名前が無効な文字を含む場合）

構文

```
public org.w3c.dom.EntityReference createEntityReference( String name);
```

パラメータ	説明
name	参照するエンティティの名前。

createEvent()

説明

指定された型のイベント・オブジェクトを作成します。

構文

```
public org.w3c.dom.events.Event createEvent( String type);
```

パラメータ	説明
type	イベントの型。

createMutationEvent()

説明

指定された型の変更イベント・オブジェクトを作成します。

構文

```
public org.w3c.dom.events.MutationEvent createMutationEvent( String type);
```

パラメータ	説明
type	変更イベントの型。

createNodeIterator()

説明

指定されたルート、ノード・イテレータの論理ビューに含めるノードのタイプを制御するフラグ、ノードをフィルタ処理するためのフィルタ、実体参照およびその子孫を含めることができるかどうかを判別するフラグを使用して、ノード・イテレータを作成します。次の DOMException が発生します。

- NOT_SUPPORTED_ERR（指定されたルートを使用してノード・イテレータを作成できなかった場合）

構文

```
public org.w3c.dom.traversal.NodeIterator createNodeIterator(
                            org.w3c.dom.Node root,
                            int whatToShow,
                            org.w3c.dom.traversal.NodeFilter filter,
                            boolean expandEntityReferences);
```

パラメータ	説明
root	イテレータのルート・ノード。
whatToShow	イテレータおよびツリー・ウォーカに含まれるノードのタイプを示すフラグ。
filter	イテレータおよびツリー・ウォーカから不要なノードをフィルタ処理する NodeFilter。
expandEntityReferences	実体参照の検索を示すフラグ。

createProcessingInstruction()

説明

指定された名前およびデータ文字列を持つ ProcessingInstruction ノードを作成します。次の DOMException が発生します。

- INVALID_CHARACTER_ERR（無効な文字を指定した場合）

構文

```
public org.w3c.dom.ProcessingInstruction createProcessingInstruction(
                        String target,
                        String data);
```

パラメータ	説明
target	処理命令のターゲット部分。
data	ノードのデータ。

createRange()

説明

ドキュメントの先頭に開始と終了の境界点を持つ新しい文書範囲オブジェクトを作成します。

構文

```
public org.w3c.dom.ranges.Range createRange();
```

createRangeEvent()

説明

指定された型の範囲イベント・オブジェクトを作成します。

構文

```
public org.w3c.dom.events.Event createRangeEvent( String type);
```

パラメータ	説明
type	範囲イベントの型。

createTextNode()

説明

指定された文字列を含む Text ノードを作成します。

構文

```
public org.w3c.dom.Text createTextNode( String data);
```

パラメータ	説明
data	ノードのデータ。

createTraversalEvent()

説明

指定された型の横断イベント・オブジェクトを作成します。

構文

```
public org.w3c.dom.events.Event createTraversalEvent( String type);
```

パラメータ	説明
type	横断イベントの型。

createTreeWalker()

説明

指定されたルート、ノード・イテレータの論理ビューに含めるノードのタイプを制御するフラグ、ノードをフィルタ処理するためのフィルタ、実体参照およびその子孫を含めることができるかどうかを判別するフラグを使用して、ノード・イテレータを作成します。次の DOMException が発生します。

- NOT_SUPPORTED_ERR（指定されたルートを使用してノード・イテレータを作成できなかった場合）

構文

```
public org.w3c.dom.traversal.TreeWalker createTreeWalker(
                          org.w3c.dom.Node root,
                          int whatToShow,
                          org.w3c.dom.traversal.NodeFilter filter,
                          boolean expandEntityReferences);
```

パラメータ	説明
root	イテレータのルート・ノード。
whatToShow	イテレータおよびツリー・ウォーカに含まれるノードのタイプを示すフラグ。

パラメータ	説明
filter	イテレータおよびツリー・ウォーカから不要なノードをフィルタ処理するノード・フィルタ。
expandEntityReferences	実体参照の検索を示すフラグ。

expectedElements()

説明

要素に追加できる要素名の一覧をベクトルで戻します。

構文

```
public java.util.Vector expectedElements( org.w3c.dom.Element e);
```

パラメータ	説明
e	要素。

getColumnNumber()

説明

列番号デバッグ情報を戻します。

構文

```
public int getColumnNumber();
```

getDebugMode()

説明

デバッグ・フラグを戻します。

構文

```
public boolean getDebugMode();
```

getDoctype()

説明

ドキュメントに対応する Document Type Declaration（DTD）を戻します。DTD なしの XML 文書の場合は、null を戻します。DOM レベル 1 仕様は DTD の編集をサポートしないことに注意してください。

構文

```
public org.w3c.dom.DocumentType getDoctype();
```

getDocumentElement()

説明

ドキュメントのルート要素である子ノードにアクセスします。

構文

```
public org.w3c.dom.Element getDocumentElement();
```

getElementById()

説明

elementId に指定された ID を持つ要素を戻します。該当する要素が存在しない場合は、null を戻します。複数の要素がこの ID を持つ場合は、動作が定義されません。

構文

```
public org.w3c.dom.Element getElementById( String elementId);
```

パラメータ	説明
elementId	一致する ID 要素の取得に使用する要素 ID。

getElementsByTagName()

説明

指定されたタグ名を持つすべての要素の NodeList を、ドキュメント・ツリーの先行順走査で検出された順に戻します。

構文

```
public org.w3c.dom.NodeList getElementsByTagName( String tagname);
```

パラメータ	説明
tagname	一致させるタグの名前。特殊な値「*」は、すべてのタグに一致します。

getElementsByTagNameNS()

説明

指定されたローカル名および名前空間 URI を持つすべての要素の NodeList を、ドキュメント・ツリーの先行順走査で検出された順に戻します。

構文

```
public org.w3c.dom.NodeList getElementsByTagNameNS( String namespaceURI,
                                                    String localName);
```

パラメータ	説明
namespaceURI	リクエストする要素の名前空間。
localName	リクエストする要素のローカル名。

getEncoding()

説明

<?xml ...?> タグに格納されたキャラクタ・エンコーディング情報またはユーザー定義の出力エンコーディング（より近い時点で設定されている場合）を戻します。

構文

```
public final String getEncoding();
```

getIDHashtable()

説明

XML DOM ツリー内の ID 要素ハッシュテーブルを戻します。

構文

```
public java.util.Hashtable getIDHashtable();
```

getImplementation()

説明

ドキュメントを処理する DOMImplementation オブジェクトを戻します。DOM アプリケーションは、複数の実装のオブジェクトを使用できます。

構文

```
public org.w3c.dom.DOMImplementation getImplementation();
```

getLineNumber()

説明

行番号デバッグ情報を戻します。

構文

```
public int getLineNumber();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

getOwnerDocument()

説明

ノードに対応する Document オブジェクトを戻します。このノードが Document である場合は、null になります。

構文

```
public org.w3c.dom.Document getOwnerDocument();
```

getStandalone()

説明

スタンドアロン情報を取得します。これは、<?xml ...?> タグに格納されたスタンドアロン属性です。

構文

```
public final String getStandalone();
```

getSystemId()

説明

ノードを含むエンティティのシステム識別子を戻します。

構文

```
public String getSystemId();
```

getText()

説明

要素に含まれている非マークアップ・テキストを戻します。テキスト要素の場合は、生データになります。子ノードを持つ要素の場合は、サブツリー全体を検索し、末端のテキスト要素ごとにテキストを追加して、サブツリーの XML マークアップを効果的に削除します。たとえば、XML 文書に「William Shakespeare」が含まれる場合、XMLDocument.getText は「William Shakespeare」を戻します。

構文

```
public String getText();
```

getVersion()

説明

<?xml ...?> タグに格納されたバージョン情報を取得します。

構文

```
public final String getVersion();
```

importNode()

説明

ドキュメントに、別のドキュメントからノードをインポートします。戻されたノードは親を持ちません（parentNode は null です）。ソース・ノードは、元のドキュメントから変更または削除されません。このメソッドは、ソース・ノードの新しいコピーを作成します。どのノードの場合でも、ノードをインポートすると、ソース・ノードの nodeName および nodeType と同じ属性値、および名前空間に関連する属性（prefix、localName および namespaceURI）を使用して、インポート先のドキュメントが所有するノード・オブジェクトが作成されます。ノードに対する cloneNode 操作の場合と同様に、ソース・ノードは変更されません。このメソッドでは、次の DOMException が発生します。

- NOT_SUPPORTED_ERR（インポートするノードのタイプがサポートされていない場合）

構文

```
public org.w3c.dom.Node importNode( org.w3c.dom.Node importedNode,
                                    boolean deep);
```

パラメータ	説明
importedNode	インポートするノード。
deep	ノードの子孫をインポートするかどうかを指定します。

insertBefore()

説明

既存の子ノード refChild の前に newChild ノードを挿入します。refChild が null の場合は、子のリストの最後に newChild を挿入します。newChild が DocumentFragment オブジェクトの場合は、newChild のすべての子が refChild の前に同じ順序で挿入されます。newChild がすでにツリー内に存在する場合は、最初にその newChild が削除されます。このメソッドでは、次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR（ノードが newChild ノードのタイプの子を許可しないタイプである場合、または挿入するノードがノードの祖先のいずれかである場合）
- WRONG_DOCUMENT_ERR（newChild がノードを作成したドキュメントとは異なるドキュメントから作成されている場合）
- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- NOT_FOUND_ERR（refChild がノードの子ではない場合）

構文

```
public org.w3c.dom.Node insertBefore( org.w3c.dom.Node newChild,
                                      org.w3c.dom.Node refChild);
```

パラメータ	説明
newChild	挿入するノード。
refChild	参照ノード（新しいノードが前に挿入されるノード）。

print()

説明

ドキュメントのコンテンツを指定された出力先に書き込みます。IOException が発生します。次の表に、オプションを示します。

構文	説明
public void print(java.io.OutputStream out);	ドキュメントのコンテンツを指定された出力ストリームに書き込みます。
public void print(java.io.OutputStream out, String enc);	ドキュメントのコンテンツを指定されたエンコーディング済出力ストリームに書き込みます。

構文	説明
public void print(java.io.PrintWriter out);	ドキュメントのコンテンツを指定されたプリント・ライターに書き込みます。
public void print(PrintDriver pd);	ドキュメントのコンテンツを指定された出力ドライバに書き込みます。

パラメータ	説明
out	書込み先の出力。
enc	出力に使用するエンコーディング。
pd	各ノードの書込みに使用する出力ドライバ。

printExternalDTD()

説明

ドキュメントのコンテンツを指定された出力ストリームに書き込みます。`IOException` が発生します。次の表に、オプションを示します。

構文	説明
public void printExternalDTD(java.io.OutputStream out);	ドキュメントのコンテンツを指定された出力ストリームに書き込みます。
public void printExternalDTD(java.io.OutputStream out, String enc);	ドキュメントのコンテンツを指定されたエンコーディング済出力ストリームに書き込みます。
public void printExternalDTD(java.io.PrintWriter out);	ドキュメントのコンテンツを指定されたプリント・ライターに書き込みます。

パラメータ	説明
out	書込み先の出力。
enc	出力に使用するエンコーディング。

readExternal()

説明

writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- `IOException`（入力ストリームの読込み中にエラーが発生した場合）
- `ClassNotFoundException`（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

removeChild()

説明

子ノードのドキュメント・リストから要素を削除します。次の `DOMException` が発生します。

- NO_MODIFICATION_ALLOWED_ERR（ドキュメントが読込み専用の場合）
- NOT_FOUND_ERR（`oldChild` がノードの子ではない場合）

構文

```
public org.w3c.dom.Node removeChild( org.w3c.dom.Node elem);
```

パラメータ	説明
elem	削除する要素。

replaceChild()

説明

子のリスト内の子ノード oldChild を newChild で置換し、oldChild ノードを戻します。newChild がすでにツリー内に存在する場合は、最初に削除されます。これは、XMLNode.removeChild() メソッドのオーバーライドです。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR（ノードが newChild ノードのタイプの子を許可しないタイプである場合）
- WRONG_DOCUMENT_ERR（newChild が異なるドキュメントから作成されている場合）
- NOT_FOUND_ERR（oldChild がノードの子ではない場合）

構文

```
public org.w3c.dom.Node replaceChild( org.w3c.dom.Node newChild,
                                      org.w3c.dom.Node oldChild);
```

パラメータ	説明
newChild	子のリストに挿入する新しいノード。
oldChild	リスト内で置換されるノード。

reportSAXEvents()

説明

DOM ツリーの SAX イベントをレポートします。SAXException が発生します。

構文

```
public void reportSAXEvents( org.xml.sax.ContentHandler cntHandler);
```

パラメータ	説明
cntHandler	コンテンツ・ハンドラ。

setDoctype()

説明

ドキュメントの DOCTYPE URI を設定します。

構文

```
public void setDoctype( String rootname,
                        String sysid,
                        String pubid);
```

パラメータ	説明
rootname	ルート要素の名前。
sysid	DOCTYPE のシステム識別子。
pubid	DOCTYPE の公開識別子（null になる場合があります）。

setEncoding()

説明

出力用のキャラクタ・エンコーディングを設定します。最終的には <?xml ...?> タグに格納されたエンコーディングを設定しますが、ドキュメントが保存されるまでは設定しません。ドキュメントがロードされるまでは、このメソッドをコールしないでください。

構文

```
public final void setEncoding( String encoding);
```

パラメータ	説明
encoding	設定するキャラクタ・エンコーディング。

setLocale()

説明

エラー・レポートのロケールを設定します。

構文

```
public final void setLocale( java.util.Locale locale);
```

パラメータ	説明
locale	エラー・レポートのロケール。

setNodeContext()

説明

ノードのコンテキストを設定します。

構文

```
public void setNodeContext( oracle.xml.util.NodeContext nctx);
```

パラメータ	説明
nctx	設定するコンテキスト。

setParsedDoctype()

説明

システム識別子を解析することによって、DOCTYPE オブジェクトを設定します。

構文

```
public void setParsedDoctype( String rootname,
                              String sysid,
                              String pubid);
```

パラメータ	説明
rootname	ルート要素の名前。
sysid	DOCTYPE のシステム識別子。
pubid	DOCTYPE の公開識別子（null になる場合があります）。

setStandalone()

説明

<?xml ...?> タグに格納されたスタンドアロン情報を設定します。

構文

```
public final void setStandalone( String value);
```

パラメータ	説明
value	属性値。

setVersion()

説明

<?xml ...?> タグに格納されたバージョン番号を設定します。

構文

```
public final void setVersion( String version);
```

パラメータ	説明
version	設定するバージョン情報。

validateElementContent()

説明

要素ノードの内容を検証します。内容が妥当である場合は true、妥当でない場合は false を戻します。

構文

```
public boolean validateElementContent( org.w3c.dom.Element elem);
```

パラメータ	説明
elem	検証する要素。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。シリアル化 / 圧縮したストリームの書込み中に例外が発生した場合、IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	シリアル化 / 圧縮したストリームの書込みに使用する ObjectOutput ストリーム。

XMLDocumentFragment クラス

XMLDocumentFragment の説明

このクラスは、DOM の DocumentFragment インタフェースを実装します。要素として処理できるように、XMLNode ではなく XMLElement を拡張します。`DocumentFragment`、`NodeFactory` および `DOMParser.setNodeFactory()` も参照してください。

XMLDocumentFragment の構文

```
public class XMLDocumentFragment implements java.io.Serializable
```

`oracle.xml.parser.v2.XMLDocumentFragment`

XMLDocumentFragment の実装済インタフェース

`java.io.Serializable`

XMLDocumentFragment のメソッド

表 2-12 XMLDocumentFragment のメソッドの概要

メソッド	説明
getAttributes()（2-79 ページ）	XMLDocumentFragment の属性を戻します。
getNodeType()（2-80 ページ）	実際の実装オブジェクトの型を表すコードを戻します。
getParentNode()（2-80 ページ）	ノードの親を戻します。

getAttributes()

説明

XMLDocumentFragment の属性（空の名前付きノード・マップ）を戻します。

構文

```
public org.w3c.dom.NamedNodeMap getAttributes();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

getParentNode()

説明

ノードの親を戻します。

構文

```
public org.w3c.dom.Node getParentNode();
```

XMLDOMException クラス

XMLDOMException の説明

このクラスは、DOM 例外を発生させるために使用されます。

XMLDOMException の構文

```
public class XMLDOMException
```

`oracle.xml.parser.v2.XMLDOMException`

XMLDOMException のメソッド

XMLDOMException()

説明

`XMLDOMException` 例外を発生させます。次の表に、オプションを示します。

構文	説明
public XMLDOMException(short code);	指定されたコードを使用して、`XMLDOMException` 例外を作成します。
public XMLDOMException(short code, String mess);	指定されたメッセージおよびエラー・コードを使用して、`XMLDOMException` 例外を作成します。

パラメータ	説明
code	DOM インタフェースに示されるコード。デフォルトのメッセージを使用します。
mess	XMLDOMException の作成に使用するメッセージ。

XMLDOMImplementation

XMLDOMImplementation の説明

このクラスは、DOMImplementation を実装します。

XMLDOMImplementation の構文

```
public class XMLDOMImplementation implements java.io.Serializable
```

oracle.xml.parser.v2.XMLDOMImplementation

XMLDOMImplementation の実装済インタフェース

```
java.io.Serializable
```

XMLDOMImplementation のメソッド

表 2-13 XMLDOMImplementation のメソッドの概要

メソッド	説明
XMLDOMImplementation() (2-82 ページ)	XMLDOMImplementation の新しいインスタンスを作成します。
createDocument() (2-83 ページ)	指定された名前および空の DocumentType ノードを使用して、指定された DocumentType ノードおよびルート要素を含む XMLDocument オブジェクトを作成します。
createDocumentType() (2-83 ページ)	ルート要素名およびシステム識別子 / 公開識別子を使用して、空の DocumentType ノードを作成します。
hasFeature()（2-84 ページ）	DOM インプリメンテーションによって固有の機能が実装されているかどうかをテストします。
setFeature()（2-84 ページ）	指定された機能を設定します。

XMLDOMImplementation()

説明

XMLDOMImplementation の新しいインスタンスを作成します。

構文

```
public  XMLDOMImplementation();
```

createDocument()

説明

指定された名前と空の DocumentType ノードを使用して、指定された DocumentType ノードおよびルート要素を含む XMLDocument オブジェクトを作成します。次の DOMException が発生します。

- INVALID_CHARACTER_ERR（指定された修飾名が無効な文字を含む場合）
- NAMESPACE_ERR（修飾名の形式が正しくない場合、修飾名が接頭辞を持ち、名前空間 URI が null または空の文字列である場合、あるいは修飾名が「xml」という接頭辞を持ち、名前空間 URI が「http://www.w3.org/XML/1998/namespace」とは異なる場合）
- WRONG_DOCUMENT_ERR（DOCTYPE がすでに別のドキュメントで使用されているか、または別の実装から作成された場合）

構文

```
public org.w3c.dom.Document createDocument( String namespaceURI,
                                            String qualifiedName,
                                            org.w3c.dom.DocumentType doctype);
```

パラメータ	説明
namespaceURI	ドキュメント内のルート要素の名前空間。
qualifiedName	ドキュメント内のルート要素の修飾名。
doctype	ドキュメントに対応する DocumentType（DTD）。

createDocumentType()

説明

ルート要素の名前およびシステム識別子 / 公開識別子を使用して、空の DocumentType ノードを作成します。作成された DocumentType オブジェクトを戻します。次の DOMException が発生します。

- INVALID_CHARACTER_ERR（指定された修飾名が無効な文字を含む場合）
- NAMESPACE_ERR（修飾名の形式が正しくない場合）

構文

```
public org.w3c.dom.DocumentType createDocumentType( String qualifiedName,
                                                    String publicId,
                                                    String systemId);
```

パラメータ	説明
qualifiedName	ルート要素の修飾名。
systemId	DocumentType ノードのシステム識別子。
publicId	DocumentType ノードの公開識別子。

hasFeature()

説明

DOM インプリメンテーションによって固有の機能が実装されているかどうかをテストします。機能が実装されている場合は `true`、実装されていない場合は `false` を戻します。

構文

```
public boolean hasFeature( String feature,
                           String version);
```

パラメータ	説明
feature	テスト対象の機能。
version	テスト対象の機能のバージョン。

setFeature()

説明

指定された機能を設定します。機能を設定できなかった場合は、`DOMException` が発生します。

構文

```
public void setFeature( String feature);
```

パラメータ	説明
feature	DOM の機能。

XMLElement クラス

XMLElement の説明

このクラスは、DOM の Element インタフェースを実装します。

XMLElement の構文

```
public class XMLElement implements oracle.xml.parser.v2.NSName,
oracle.xml.parser.v2.NSResolver, java.io.Externalizable
```

oracle.xml.parser.v2.XMLElement

XMLElement の実装済インタフェース

```
java.io.Externalizable, NSName, oracle.xml.util.NSName, NSResolver,
java.io.Serializable
```

XMLElement のメソッド

表 2-14 XMLElement のメソッドの概要

メソッド	説明
XMLElement()（2-87 ページ）	デフォルトのコンストラクタです。
cloneNode()（2-88 ページ）	ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。
getAttribute()（2-88 ページ）	属性名を持つ任意の属性値を戻します。その属性が指定された値またはデフォルト値を持たない場合は、空の文字列を戻します。
getAttributeNode()（2-88 ページ）	属性名を持つ任意の Attr ノードを戻します。該当する属性が存在しない場合は、null を戻します。
getAttributeNodeNS()（2-89 ページ）	指定された名前空間 URI およびローカル名を持つ属性（存在する場合）を戻します。該当する属性が存在しない場合は、null を戻します。
getAttributeNS()（2-89 ページ）	名前空間 URI およびローカル名を持つ属性（存在する場合）の値を戻します。名前空間 URI およびローカル名を持たない属性の場合は、null を戻します。
getAttributes()（2-90 ページ）	ノードが要素の場合は、その属性を含む名前付きノード・マップを戻します。ノードが要素でない場合は、null を戻します。

表 2-14 XMLElement のメソッドの概要（続き）

メソッド	説明
getChildrenByTagName() (2-90 ページ)	指定されたタグ名を持つすべての子ノードの NodeList を戻します。
getElementsByTagName() (2-91 ページ)	指定されたタグ名を持つすべての要素の NodeList を、ドキュメント・ツリーの先行順走査で検出された順に戻します。
getElementsByTagNameNS() (2-91 ページ)	指定されたローカル名および名前空間 URI を持つすべての子孫要素の NodeList を、要素ツリーの先行順走査で検出された順に戻します。
getExpandedName() (2-91 ページ)	要素の解決済完全名を戻します。
getFirstAttribute() (2-92 ページ)	最初の Attr ノードを取得します。属性が存在しない場合は、`null` を戻します。
getLocalName() (2-92 ページ)	要素のローカル名を戻します。
getNamespaceURI() (2-92 ページ)	要素の名前空間 URI を戻します。
getNodeType() (2-92 ページ)	実際の実装オブジェクトの型を表すコードを戻します。
getPrefix() (2-93 ページ)	要素の名前空間接頭辞を戻します。
getQualifiedName() (2-93 ページ)	要素の修飾名を戻します。
getTagName() (2-93 ページ)	要素の名前を戻します。
hasAttribute() (2-93 ページ)	指定された名前を持つ属性が要素に対して指定されているか、またはデフォルト値を持つ場合は、`true` を戻します。
hasAttributeNS() (2-94 ページ)	指定されたローカル名および名前空間 URI を持つ属性が要素に対して指定されているか、またはデフォルト値を持つ場合は、`true` を戻します。
hasAttributes() (2-94 ページ)	ノードが属性を持つ場合、`true` を戻します。
readExternal() (2-94 ページ)	入力ストリームを読み込み、その入力ストリームの情報に基づいてオブジェクトを再生成することによって、`writeExternal()` で書き込んだ情報をリストアします。
removeAttribute() (2-95 ページ)	属性名を持つ任意の属性を削除します。
removeAttributeNode() (2-95 ページ)	指定された属性を削除して戻します。

表 2-14 XMLElement のメソッドの概要（続き）

メソッド	説明
removeAttributeNS()（2-96 ページ）	ローカル名および名前空間 URI を持つ任意の属性を削除します。
reportSAXEvents()（2-96 ページ）	DOM ツリーから SAX イベントをレポートします。
resolveNamespacePrefix()（2-96 ページ）	名前空間の接頭辞を指定し、要素の有効範囲内で名前空間の定義を検索します。
setAttribute()（2-97 ページ）	新しい属性を追加します。
setAttributeNode()（2-98 ページ）	新しい属性ノードを追加します。
setAttributeNodeNS()（2-98 ページ）	名前空間を認識する新しい属性ノードを追加します。
validateContent()（2-99 ページ）	要素ノードの内容を検証します。
writeExternal()（2-100 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

XMLElement()

説明

デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。ノードを非シリアル化して、シリアル化 / 圧縮したストリームから DOM ノードを作成するには、オブジェクトのハンドルを作成する必要があります。通常のすべての XMLElement の作成では、XMLDocument の `createElement()` を使用してください。

構文

```
public  XMLElement();
```

cloneNode()

説明

ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。この複製ノードは親を持ちません（parentNode は null を戻します）。Element を複製すると、デフォルトの属性を表すために XML プロセッサによって生成された属性を含む、すべての属性およびその値がコピーされます。ただし、このメソッドでは、ノードに含まれるテキストは、子の Text ノードに含まれているため、ディープ・クローンでないかぎりコピーされません。他のタイプのノードを複製すると、単にそのノードのコピーが戻されます。

構文

```
public org.w3c.dom.Node cloneNode(boolean deep);
```

パラメータ	説明
deep	true の場合は、指定されたノードの下位サブツリーが再帰的に複製されます。false の場合は、ノードのみが（ノードが Element の場合は、その属性も）複製されます。

getAttribute()

説明

属性名を持つ任意の属性値を戻します。その属性が指定された値またはデフォルト値を持たない場合は、空の文字列を戻します。

構文

```
public String getAttribute( String name);
```

パラメータ	説明
name	取得する属性の名前。

getAttributeNode()

説明

属性名を持つ任意の Attr ノードを戻します。該当する属性が存在しない場合は、null を戻します。

構文

```
public org.w3c.dom.Attr getAttributeNode( String name);
```

パラメータ	説明
name	取得する属性ノードの名前。

getAttributeNodeNS()

説明

指定された名前空間 URI およびローカル名を持つ属性（存在する場合）を戻します。該当する属性が存在しない場合は、null を戻します。

構文

```
public org.w3c.dom.Attr getAttributeNodeNS( String namespaceURI,
                                            String localName);
```

パラメータ	説明
namespaceURI	リクエストする属性ノードの名前空間。
localName	リクエストする属性ノードのローカル名。

getAttributeNS()

説明

名前空間 URI およびローカル名を持つ属性（存在する場合）の値を戻します。名前空間 URI およびローカル名を持たない属性の場合は、null を戻します。

構文

```
public String getAttributeNS( String namespaceURI,
                              String localName);
```

パラメータ	説明
namespaceURI	リクエストする属性の名前空間。
localName	リクエストする属性のローカル名。

getAttributes()

説明

ノードが要素の場合は、その属性を含む名前付きノード・マップを戻します。ノードが要素でない場合は、null を戻します。

構文

```
public org.w3c.dom.NamedNodeMap getAttributes();
```

getChildrenByTagName()

説明

指定されたタグ名を持つすべての子ノードの NodeList を戻します。次の表に、オプションを示します。

構文	説明
public org.w3c.dom.NodeList getChildrenByTagName(String name);	指定されたタグ名を持つすべての子ノードの NodeList を戻します。
public org.w3c.dom.NodeList getChildrenByTagName(String name, String ns);	指定されたタグ名および名前空間を持つすべての子ノードの NodeList を戻します。

パラメータ	説明
name	一致させるタグの名前（ローカル名である必要があります）。
ns	名前空間。

getElementsByTagName()

説明

指定されたタグ名を持つすべての要素の NodeList をドキュメント・ツリーの先行順走査で検出された順に戻します。

構文

```
public org.w3c.dom.NodeList getElementsByTagName( String tagname);
```

パラメータ	説明
tagname	一致させるタグの名前。特殊な値「*」は、すべてのタグに一致します。

getElementsByTagNameNS()

説明

指定されたローカル名および名前空間 URI を持つすべての子孫要素の NodeList を要素ツリーの先行順走査で検出された順に戻します。

構文

```
public org.w3c.dom.NodeList getElementsByTagNameNS( String namespaceURI,
                                                    String localName);
```

パラメータ	説明
namespaceURI	要素の名前空間。
localName	要素のローカル名。

getExpandedName()

説明

要素の解決済完全名を戻します。

構文

```
public String getExpandedName();
```

getFirstAttribute()

説明

最初の Attr ノードを取得します。属性が存在しない場合は、null を戻します。

構文

```
public XMLNode getFirstAttribute();
```

getLocalName()

説明

要素のローカル名を戻します。

構文

```
public String getLocalName();
```

getNamespaceURI()

説明

要素の名前空間 URI を戻します。

構文

```
public String getNamespaceURI();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

getPrefix()

説明

要素の名前空間接頭辞を戻します。

構文

```
public String getPrefix();
```

getQualifiedName()

説明

要素の修飾名を戻します。

構文

```
public String getQualifiedName();
```

getTagName()

説明

要素名を戻します。たとえば、<elementExample id="demo"> ... </elementExample> の場合、tagName は「elementExample」という値を持ちます。XML では、すべての DOM 操作の場合と同様に、大 / 小文字が保持されることに注意してください。HTML DOM は、ソース HTML ドキュメントでの大 / 小文字にかかわらず、HTML 要素の tagName を正規の大文字形式で戻します。

構文

```
public String getTagName();
```

hasAttribute()

説明

指定された名前を持つ属性が要素に対して指定されているか、またはデフォルト値を持つ場合は、true を戻します。それ以外の場合は、false を戻します。

構文

```
public boolean hasAttribute( String name);
```

パラメータ	説明
name	存在を確認する属性の名前。

hasAttributeNS()

説明

指定されたローカル名および名前空間 URI を持つ属性が要素に対して指定されているか、またはデフォルト値を持つ場合は、`true` を戻します。それ以外の場合は、`false` を戻します。

構文

```
public boolean hasAttributeNS( String namespaceURI,
                               String localName);
```

パラメータ	説明
namespaceURI	存在を確認する属性の名前空間。
localName	存在を確認する属性のローカル名。

hasAttributes()

説明

ノードが属性を持つ場合は `true`、属性を持たない場合は `false` を戻します。

構文

```
public boolean hasAttributes();
```

readExternal()

説明

入力ストリームを読み込み、その入力ストリームの情報に基づいてオブジェクトを再生成することによって、writeExternal で書き込んだ情報をリストアします。次の例外が発生します。

- `IOException`（圧縮ストリームの読込み中に例外が発生した場合）
- `ClassNotFoundException`（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

removeAttribute()

説明

属性名を持つ任意の属性を削除します。削除対象の属性がデフォルト値を持つ場合は、すぐに置換されます。このメソッドでは、次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）

構文

```
public void removeAttribute( String name);
```

パラメータ	説明
name	削除する属性の名前。

removeAttributeNode()

説明

指定された属性を削除して戻します。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- NOT_FOUND_ERR（oldAttr が要素の属性ではない場合）

構文

```
public org.w3c.dom.Attr removeAttributeNode( org.w3c.dom.Attr oldAttr);
```

パラメータ	説明
oldAttr	属性リストから削除する Attr ノード。削除対象の属性がデフォルト値を持つ場合は、すぐに置換されます。

removeAttributeNS()

説明

ローカル名および名前空間 URI を持つ任意の属性を削除します。次の DOMException が発生します。

- NO_MODIFICATIONS_ALLOWED_ERR（要素が読込み専用の場合）

構文

```
public void removeAttributeNS( String namespaceURI,
                               String localName);
```

パラメータ	説明
namespaceURI	削除する属性の名前空間。
localName	削除する属性のローカル名。

reportSAXEvents()

説明

DOM ツリーから SAX イベントをレポートします。SAXException が発生します。

構文

```
public void reportSAXEvents( org.xml.sax.ContentHandler cntHandler);
```

パラメータ	説明
cntHandler	コンテンツ・ハンドラ。

resolveNamespacePrefix()

説明

名前空間の接頭辞を指定し、要素の有効範囲内で名前空間の定義を検索します。

構文

```
public String resolveNamespacePrefix( String prefix);
```

パラメータ	説明
prefix	接頭辞の場合は、解決する名前空間の接頭辞。デフォルトの場合は、デフォルトの名前空間を戻します。

setAttribute()

説明

新しい属性を追加します。指定した属性名を持つ属性がすでに要素内に存在する場合は、その値が value パラメータの値に変更されます。この値は単純な文字列で、設定されている場合は解析されません。したがって、すべてのマークアップ（実体参照として認識される必要がある構文など）はリテラル・テキストとして処理され、作成時に実装によって適切にエスケープされる必要があります。実体参照を含む属性の値を割り当てるには、`Attr` ノードに加えて、任意の `Text` ノードおよび `EntityReference` ノードを作成し、適切なサブツリーを構築して、`setAttributeNode` によって属性値として割り当てる必要があります。このメソッドは名前空間を認識しないため、このメソッドを介して新しい属性を追加した場合、名前空間表は更新されません。このメソッドでは、次の `DOMException` が発生します。

- INVALID_CHARACTER_ERR（指定された名前が無効な文字を含む場合）
- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）

構文

```
public void setAttribute( String name,
                          String value);
```

パラメータ	説明
name	作成または変更する属性の名前。
value	文字列形式で設定する値。

setAttributeNode()

説明

新しい属性を追加します。指定した属性名を持つ属性がすでに要素内に存在する場合は、新しい属性によって置換されます。同じ属性名を持つ既存の属性が newAttr 属性によって置換されると、以前に存在した Attr ノードが戻されます。置換されない場合は、null が戻されます。このメソッドでは、次の DOMException が発生します。

- WRONG_DOCUMENT_ERR（newAttr が要素を作成したドキュメントとは異なるドキュメントから作成されている場合）
- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- INUSE_ATTRIBUTE_ERR（newAttr がすでに別の Element オブジェクトの属性である場合。DOM ユーザーは、他の要素で再利用できるように、Attr ノードを明示的に複製する必要があります。）

構文

```
public org.w3c.dom.Attr setAttributeNode ( org.w3c.dom.Attr newAttr);
```

パラメータ	説明
newAttr	属性リストに追加する属性。

setAttributeNodeNS()

説明

新しい属性を追加します。次の DOMException が発生します。

- INVALID_CHARACTER_ERR（指定された修飾名が無効な文字を含む場合）
- NAMESPACE_ERR（修飾名の形式が正しくない場合、修飾名が接頭辞を持ち、名前空間 URI が null または空の文字列である場合、修飾名が「xmlns」で、名前空間 URI が「http://www.w3.org/2000/xmlns/」とは異なる場合、あるいは修飾名が「xml」という接頭辞を持ち、名前空間 URI が「http://www.w3.org/XML/1998/namespace」とは異なる場合）
- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- WRONG_DOCUMENT_ERR（newAttr が要素を作成したドキュメントとは異なるドキュメントから作成されている場合）
- INUSE_ATTRIBUTE_ERR（newAttr がすでに別の Element オブジェクトの属性である場合）

次の表に、オプションを示します。

構文	説明
public org.w3c.dom.Attr setAttributeNodeNS(org.w3c.dom.Attr newAttr);	新しい属性ノードを追加して戻します。指定したローカル名および名前空間 URI を持つ属性がすでに要素内に存在する場合は、新しい属性によって置換されます。
public void setAttributeNS(String namespaceURI, String qualifiedName, String value);	名前空間 URI、修飾名および値から新しい属性ノードを作成して追加します。同じローカル名および名前空間 URI を持つ属性がすでに要素に存在する場合、その接頭辞は修飾名の接頭辞部分になるように変更され、その値は value パラメータの値になるように変更されます。この値は単純な文字列で、設定されている場合は解析されません。したがって、すべてのマークアップ（実体参照として認識される必要がある構文など）はリテラル・テキストとして処理され、作成時に実装によって適切にエスケープされる必要があります。

パラメータ	説明
newAttr	属性リストに追加する属性。
namespaceURI	追加する属性の名前空間。
qualifiedName	追加する属性のローカル名。
value	追加する属性の値。

validateContent()

説明

要素ノードの内容を検証します。内容が妥当である場合は `true`、妥当でない場合は `false` を戻します。次の表に、オプションを示します。

構文	説明
public boolean validateContent(DTD dtd);	DTD を使用して、要素ノードの内容を検証します。
public boolean validateContent(oracle.xml.parser.schema.XMLSchema schema);	指定された XML Schema の schema パラメータに対して要素ノードの内容を検証します。

構文	説明
public boolean validateContent(oracle.xml.parser.schema.XMLSchema schema, String mode);	要素の内容を、指定された XML Schema に対して指定されたモードで検証します。

パラメータ	説明
dtd	要素の検証に使用する DTD。
schema	要素の検証に使用する XMLSchema オブジェクト。
mode	検証モード。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	シリアル化 / 圧縮したストリームの書込みに使用する ObjectOutput ストリーム。

XMLEntity クラス

XMLEntity の説明

このクラスは、DOM の Entity インタフェースを実装し、内部または外部の XML エンティティを XML の Document Type Definition（DTD）で定義されたとおりに表します。

XMLEntity の構文

```
public class XMLEntity implements java.io.Externalizable
```

oracle.xml.parser.v2.XMLEntity

XMLEntity の実装済インタフェース

java.io.Externalizable, java.io.Serializable

XMLEntity のメソッド

表 2-15 XMLEntity のメソッドの概要

メソッド	説明
XMLEntity()（2-102 ページ）	デフォルトのコンストラクタです。
cloneNode()（2-102 ページ）	ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。
getNodeType()（2-102 ページ）	実際の実装オブジェクトの型を表すコードを戻します。
getNodeValue()（2-103 ページ）	ノードのタイプに応じて、ノードの値を戻します。
getNotationName()（2-103 ページ）	解析対象外のエンティティに対する表記法の名前を戻します。解析対象エンティティの場合は、null になります。
getPublicId()（2-103 ページ）	公開識別子を戻します。
getSystemId()（2-103 ページ）	システム識別子を戻します。
readExternal()（2-104 ページ）	writeExternal() メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
setNodeValue()（2-104 ページ）	エンティティの値を設定します。
writeExternal()（2-105 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

XMLEntity()

説明

デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。ノードを非シリアル化して、シリアル化 / 圧縮したストリームから DOM ノードを作成するには、オブジェクトのハンドルを作成する必要があります。

構文

```
public  XMLEntity();
```

cloneNode()

説明

ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。この複製ノードは親を持ちません（parentNode は null を戻します）。Element を複製すると、デフォルトの属性を表すために XML プロセッサによって生成された属性を含む、すべての属性およびその値がコピーされます。ただし、このメソッドでは、ノードに含まれるテキストは、子の Text ノードに含まれているため、ディープ・クローンでないかぎりコピーされません。他のタイプのノードを複製すると、単にそのノードのコピーが戻されます。

構文

```
public org.w3c.dom.Node cloneNode(boolean deep);
```

パラメータ	説明
deep	true の場合は、指定されたノードの下位サブツリーが再帰的に複製されます。false の場合は、ノードのみが（ノードが Element の場合は、その属性も）複製されます。

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

getNodeValue()

説明

ノードのタイプに応じて、ノードの値を戻します。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- DOMSTRING_SIZE_ERR（実装プラットフォーム上で DOMString 変数に収まらない文字が戻された場合）

構文

```
public String getNodeValue();
```

getNotationName()

説明

解析対象外のエンティティに対する表記法の名前を戻します。解析対象エンティティの場合は、null になります。

構文

```
public String getNotationName();
```

getPublicId()

説明

エンティティに対応する公開識別子（指定されている場合）を戻します。公開識別子が指定されていない場合は、null になります。

構文

```
public String getPublicId();
```

getSystemId()

説明

エンティティに対応するシステム識別子（指定されている場合）を戻します。システム識別子が指定されていない場合は、null になります。

構文

```
public String getSystemId();
```

readExternal()

説明

writeExternal() メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- `IOException`（入力ストリームの読込み中にエラーが発生した場合）
- `ClassNotFoundException`（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

setNodeValue()

説明

エンティティの値を設定します。

構文

```
public void setNodeValue( String arg);
```

パラメータ	説明
arg	エンティティの新しい値。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	シリアル化 / 圧縮したストリームの書込みに使用する ObjectOutput ストリーム。

XMLEntityReference クラス

XMLEntityReference の説明

このクラスは、DOM の EntityReference インタフェースを実装します。

XMLEntityReference の構文

```
public class XMLEntityReference implements java.lang.Cloneable,
java.io.Externalizable
```

oracle.xml.parser.v2.XMLEntityReference

XMLEntityReference の実装済インタフェース

java.lang.Cloneable, java.io.Externalizable, java.io.Serializable

XMLEntityReference のメソッド

表 2-16 XMLEntityReference のメソッドの概要

メソッド	説明
XMLEntityReference()（2-107 ページ）	デフォルトのコンストラクタです。
getNodeType()（2-107 ページ）	実際の実装オブジェクトの型を表すコードを戻します。
readExternal()（2-107 ページ）	writeExternal() メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
writeExternal()（2-108 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

XMLEntityReference()

説明

デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。ノードを非シリアル化して、シリアル化 / 圧縮したストリームから DOM ノードを作成するには、オブジェクトのハンドルを作成する必要があります。

構文

```
public  XMLEntityReference();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

readExternal()

説明

writeExternal() メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- `IOException`（入力ストリームの読込み中にエラーが発生した場合）
- `ClassNotFoundException`（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。圧縮ストリームの書込み中に例外が発生した場合、IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	圧縮ストリームの書込みに使用する ObjectOutput ストリーム。

XMLNode クラス

XMLNode の説明

このクラスは、DOM の Node インタフェースを実装し、ドキュメント・オブジェクト・モデル全体のプライマリ・データ型として機能します。これは、ドキュメント・ツリーの単一ノードを表します。

属性 nodeName、nodeValue および attributes は、特定の派生したインスタンスに依存せずに、ノード情報を取得するメカニズムとして含まれます。特定の nodeType に対するこれらの属性（Element に対する nodeValue や Comment に対する attributes など）が明白にマップされていない場合は、null を戻します。派生クラスには、関連する情報を取得および設定するために、より有効な追加のメカニズムが含まれる場合があります。XMLNSNode ではなく XMLNode を拡張する DOM ノードは、DOM 仕様によって定義されている固定ノード名を持ちます。また、子ノードを持つことができないノードのみが、このクラスを拡張します。

XMLNode の構文

```
public abstract class XMLNode implements java.lang.Cloneable, java.io.Externalizable
```

oracle.xml.parser.v2.XMLNode

XMLNode のサブクラス

XMLNSNode

XMLNode の実装済インタフェース

java.lang.Cloneable, java.io.Externalizable, java.io.Serializable

XMLNode のフィールド

表 2-17 XMLNode のフィールド

フィールド	構文	説明
ATTRDECL	public static final short ATTRDECL	属性宣言ノード。
Auto_Events	public static final String Auto_Events	自動イベントを設定するフラグ。

表 2-17 XMLNode のフィールド（続き）

フィールド	構文	説明
capturing	public static final String capturing	イベント・ターゲットによって処理される前に、イベント・ターゲットの祖先のいずれかによって処理できます。
DOMAttrModified	public static final String DOMAttrModified	ノードの属性が変更されたことを示します。
DOMCharacterDataModified	public static final String DOMCharacterDataModified	ノード内の文字データが変更されたことを示します。
DOMNodeInserted	public static final String DOMNodeInserted	ノードが別のノードの子として追加されたことを示します。
DOMNodeInsertedIntoDocument	public static final String DOMNodeInsertedIntoDocument	ノードの直接挿入またはノードが含まれているサブツリーの挿入によって、ノードがドキュメントに挿入されたことを示します。
DOMNodeRemoved	public static final String DOMNodeRemoved	ノードがその親ノードから削除されたことを示します。
DOMNodeRemovedFromDocument	public static final String DOMNodeRemovedFromDocument	ノードを直接削除したか、またはノードが含まれているサブツリーを削除することによって、ノードがドキュメントから削除されたことを示します。
DOMSubtreeModified	public static final String DOMSubtreeModified	ドキュメントへのすべての変更を通知するための汎用イベントです。特定のイベントのかわりに使用できます。
ELEMENTDECL	public static final short ELEMENTDECL	要素宣言。
noncapturing	public static final String noncapturing	イベント・ターゲットの祖先のいずれかによって処理される前に、イベント・ターゲットによって処理されます。

表 2-17 XMLNode のフィールド（続き）

フィールド	構文	説明
RANGE_DELETE_EVENT	public static final String RANGE_DELETE_EVENT	範囲イベントを削除するフラグ。
RANGE_DELETETEXT_EVENT	public static final String RANGE_DELETETEXT_EVENT	範囲削除テキスト・イベントを設定するフラグ。
RANGE_INSERT_EVENT	public static final String RANGE_INSERT_EVENT	範囲イベントを設定するフラグ。
RANGE_INSERTTEXT_EVENT	public static final String RANGE_INSERTTEXT_EVENT	範囲挿入テキスト・イベントを設定するフラグ。
RANGE_REPLACE_EVENT	public static final String RANGE_REPLACE_EVENT	範囲イベントを置換するフラグ。
RANGE_SETTEXT_EVENT	public static final String RANGE_SETTEXT_EVENT	範囲テキスト・イベントを設定するフラグ。
TRAVERSAL_DELETE_EVENT	public static final String TRAVERSAL_DELETE_EVENT	横断削除イベントを設定するフラグ。
TRAVERSAL_REPLACE_EVENT	public static final String TRAVERSAL_REPLACE_EVENT	横断置換イベントを設定するフラグ。
XMLDECL_NODE	public static final short XMLDECL_NODE	属性宣言ノード。

XMLNode のメソッド

表 2-18 XMLNode のメソッドの概要

メソッド	説明
XMLNode()（2-114 ページ）	新しい XMLNode を作成します。
addEventListener()（2-114 ページ）	イベント・ターゲット（ノード）にイベント・リスナーを登録します。
appendChild()（2-115 ページ）	ノードの子のリストの最後にノード newChild を追加し、その新しいノードを戻します。

表 2-18 XMLNode のメソッドの概要（続き）

メソッド	説明
cloneNode()（2-115 ページ）	ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。
dispatchEvent()（2-116 ページ）	イベントを実装イベント・モデルにディスパッチします。
getAttributes()（2-116 ページ）	ノードの属性を含む名前付きノード・マップを戻します。
getChildNodes()（2-116 ページ）	ノードのすべての子を NodeList で戻します。
getColumnNumber()（2-117 ページ）	列番号デバッグ情報を戻します。
getDebugMode()（2-117 ページ）	デバッグ情報モードを戻します。
getFirstChild()（2-117 ページ）	ノードの最初の子を戻します。
getLastChild()（2-117 ページ）	ノードの最後の子を戻します。
getLineNumber()（2-118 ページ）	行番号デバッグ情報を戻します。
getLocalName()（2-118 ページ）	名前空間をサポートするノードのタイプによってオーバーライドされたノードのローカル名を戻します。
getNamespaceURI()（2-118 ページ）	名前空間をサポートするノードのタイプによってオーバーライドされたノードの名前空間 URI を戻します。
getNextSibling()（2-118 ページ）	ノードの直後のノードを戻します。
getNodeName()（2-119 ページ）	ノードの名前を戻します。
getNodeType()（2-119 ページ）	ノードのタイプを戻します。
getNodeValue()（2-119 ページ）	ノードのタイプに応じて、ノードの値を戻します。
getOwnerDocument()（2-119 ページ）	ノードに対応するドキュメント・オブジェクトを戻します。
getParentNode()（2-120 ページ）	ノードの親を戻します。
getPrefix()（2-120 ページ）	名前空間をサポートするノードのタイプによってオーバーライドされたノードの接頭辞を戻します。
getPreviousSibling()（2-120 ページ）	ノードの直前のノードを戻します。
getProperty()（2-120 ページ）	ノードのプロパティの値を戻します。
getSystemId()（2-121 ページ）	ノードを含むエンティティのシステム識別子を戻します。
getText()（2-121 ページ）	要素に含まれている非マークアップ・テキストを戻します。

表 2-18 XMLNode のメソッドの概要（続き）

メソッド	説明
hasAttributes()（2-121 ページ）	ノードが要素の場合は、属性を持っているかどうかを判別します。
hasChildNodes()（2-121 ページ）	ノードが子を持っているかどうかを判別します。
insertBefore()（2-122 ページ）	既存の子ノード refChild の前にノード newChild を挿入します。
isNodeFlag()（2-122 ページ）	ノード・フラグ情報が設定されている場合は、`true` を戻します。
isSupported()（2-123 ページ）	DOM インプリメンテーションによって固有の機能が実装され、その機能がノードによってサポートされているかどうかをテストします。
print()（2-123 ページ）	ノードの内容を出力先に書き込みます。
readExternal()（2-124 ページ）	writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
removeChild()（2-124 ページ）	oldChild に指定された子ノードを子のリストから削除します。
removeEventListener()（2-124 ページ）	イベント・ターゲット（ノード）からイベント・リスナーを削除します。
replaceChild()（2-125 ページ）	子のリスト内の子ノード oldChild を newChild と置換し、置換されたノードを戻します。
reportSAXEvents()（2-126 ページ）	DOM ツリーから SAX イベントをレポートします。SAXException が発生します。
resetNodeFlag()（2-126 ページ）	ノード・フラグ情報をリセットします。
selectNodes()（2-126 ページ）	ツリーから、指定したパターンに一致するノードを NodeList として戻します。
selectSingleNode()（2-127 ページ）	ツリーから、指定したパターンに一致する最初のノードを戻します。
setDebugInfo()（2-128 ページ）	ノードのデバッグ情報を設定します。
setNodeFlag()（2-128 ページ）	ノード・フラグ情報を設定します。
setNodeValue()（2-128 ページ）	ノードのタイプに応じて、ノードの値を設定します。
setPrefix()（2-129 ページ）	名前空間をサポートするノードのタイプによってオーバーライドされたノードの接頭辞を設定します。

表 2-18 XMLNode のメソッドの概要（続き）

メソッド	説明
setProperty()（2-129 ページ）	ノードのプロパティを設定します。
transformNode()（2-130 ページ）	指定されたスタイルシートを使用してツリー内のノードを変換し、結果のドキュメント・フラグメントを戻します。
valueOf()（2-130 ページ）	ツリーから、パターンに一致する最初のノードの値を選択します。
writeExternal()（2-131 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

XMLNode()

説明

新しい XMLNode を作成します。

構文

```
protected  XMLNode();
```

addEventListener()

説明

イベント・ターゲット（ノード）にイベント・リスナーを登録します。

構文

```
public void addEventListener( String type,
                              org.w3c.dom.events.EventListener listener,
                              boolean useCapture);
```

パラメータ	説明
type	リスナーを登録するイベントの型。
listener	リスナー・オブジェクト。
useCapture	リスナーが取得を開始する必要があるかどうかを示すフラグ。

appendChild()

説明

ノードの子のリストの最後に newChild ノードを追加し、その新しいノードを戻します。newChild がすでにツリー内に存在する場合は、最初に削除されます。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR（ノードが newChild ノードのタイプの子を許可しないタイプである場合、または追加するノードがこのノードの祖先のいずれかである場合）
- WRONG_DOCUMENT_ERR（newChild がノードを作成したドキュメントとは異なるドキュメントから作成された場合）
- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）

構文

```
public org.w3c.dom.Node appendChild( org.w3c.dom.Node newChild);
```

パラメータ	説明
newChild	追加するノード。このノードが DocumentFragment オブジェクトの場合は、ドキュメント・フラグメントの内容全体がノードの子のリストに移動されます。

cloneNode()

説明

ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。この複製ノードは親を持ちません（parentNode は null を戻します）。Element を複製すると、デフォルトの属性を表すために XML プロセッサによって生成された属性を含む、すべての属性およびその値がコピーされます。ただし、このメソッドでは、ノードに含まれるテキストは、子の Text ノードに含まれているため、ディープ・クローンでないかぎりコピーされません。他のタイプのノードを複製すると、単にそのノードのコピーが戻されます。

構文

```
public org.w3c.dom.Node cloneNode( boolean deep);
```

パラメータ	説明
deep	true の場合は、指定されたノードの下位サブツリーが再帰的に複製されます。false の場合は、ノードのみが（ノードが Element の場合は、その属性も）複製されます。

dispatchEvent()

説明

イベントを実装イベント・モデルにディスパッチします。dispatchEvent をコールする前に、イベントを初期化してイベントの型を指定していない場合は、UNSPECIFIED_EVENT_TYPE という値の例外が発生します。

構文

```
public boolean dispatchEvent(org.w3c.dom.events.Event evt);
```

パラメータ	説明
evt	preventDefault() または stopPropogation() がコールされたかどうかを示します。

getAttributes()

説明

ノードが Element の場合は、その属性を含む名前付きノード・マップを戻します。ノードが Element でない場合は、null を戻します。

構文

```
public org.w3c.dom.NamedNodeMap getAttributes();
```

getChildNodes()

説明

ノードのすべての子を NodeList で戻します。子が存在しない場合は、ノードを含まない NodeList になります。戻された NodeList の内容は、作成される基となったノード・オブジェクトの子ノードに変更があった場合、NodeList アクセッサが戻すノードにすぐに反映されるという点などからみて「最新」のもので、ノード内容の静的スナップショットではありません。これは、getElementsByTagName メソッドが戻す NodeList を含め、すべての NodeList に当てはまります。

構文

```
public org.w3c.dom.NodeList getChildNodes();
```

getColumnNumber()

説明
列番号デバッグ情報を戻します。

構文
```
public int getColumnNumber();
```

getDebugMode()

説明
デバッグ情報モードを戻します。

構文
```
public boolean getDebugMode();
```

getFirstChild()

説明
ノードの最初の子を戻します。該当するノードが存在しない場合は、null を戻します。

構文
```
public org.w3c.dom.Node getFirstChild();
```

getLastChild()

説明
ノードの最後の子を戻します。該当するノードが存在しない場合は、null を戻します。

構文
```
public org.w3c.dom.Node getLastChild();
```

getLineNumber()

説明

行番号デバッグ情報を戻します。

構文

```
public int getLineNumber();
```

getLocalName()

説明

名前空間をサポートするノードのタイプによってオーバーライドされたノードのローカル名を戻します。

構文

```
public String getLocalName();
```

getNamespaceURI()

説明

名前空間をサポートするノードのタイプによってオーバーライドされたノードの名前空間 URI を戻します。

構文

```
public String getNamespaceURI();
```

getNextSibling()

説明

ノードの直後のノードを戻します。該当するノードが存在しない場合は、`null` を戻します。

構文

```
public org.w3c.dom.Node getNextSibling();
```

getNodeName()

説明

ノード名を戻します。

構文

```
public String getNodeName();
```

getNodeType()

説明

ノードのタイプを戻します。

構文

```
public short getNodeType();
```

getNodeValue()

説明

ノードのタイプに応じて、ノードの値を戻します。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- DOMSTRING_SIZE_ERR（実装プラットフォーム上で DOMString 変数に収まらない文字が戻された場合）

構文

```
public String getNodeValue();
```

getOwnerDocument()

説明

ノードに対応する Document オブジェクトを戻します。これは、新しいノードの作成に使用される Document オブジェクトでもあります。ノードが Document の場合は、null になります。

構文

```
public org.w3c.dom.Document getOwnerDocument();
```

getParentNode()

説明

ノードの親を戻します。Document、DocumentFragment および Attr 以外のすべてのノードは、親を持つ場合があります。ただし、ノードが作成された直後で、ツリーに追加されていない場合、またはノードがツリーから削除されている場合は、null になります。

構文

```
public org.w3c.dom.Node getParentNode();
```

getPrefix()

説明

名前空間をサポートするノードのタイプによってオーバーライドされたノードの接頭辞を戻します。

構文

```
public String getPrefix();
```

getPreviousSibling()

説明

ノードの直前のノードを戻します。該当するノードが存在しない場合は、null を戻します。

構文

```
public org.w3c.dom.Node getPreviousSibling();
```

getProperty()

説明

ノードのプロパティの値を戻します。

構文

```
public Object getProperty( String propName);
```

パラメータ	説明
propName	プロパティ名。

getSystemId()

説明

ノードを含むエンティティのシステム識別子を戻します。

構文

```
public String getSystemId();
```

getText()

説明

要素に含まれている非マークアップ・テキストを戻します。テキスト要素の場合は、生データになります。子ノードを持つ要素の場合は、サブツリー全体を検索し、末端のテキスト要素ごとにテキストを追加して、サブツリーの XML マークアップを効果的に削除します。

構文

```
public String getText();
```

hasAttributes()

説明

ノードが要素の場合は、属性を持っているかどうかを判断します。ノードが属性を持つ場合は `true`、属性を持たない場合は `false` を戻します。

構文

```
public boolean hasAttributes();
```

hasChildNodes()

説明

ノードが子を持っているかどうかを判断します。ノードが子を持つ場合は `true`、子を持たない場合は `false` を戻します。

構文

```
public boolean hasChildNodes();
```

insertBefore()

説明

既存の子ノード refChild の前に newChild ノードを挿入し、そのノードを戻します。refChild が null の場合は、子のリストの最後に newChild を挿入します。newChild が DocumentFragment オブジェクトの場合は、そのすべての子が同じ順序で refChild の前に挿入されます。newChild がすでにツリー内に存在する場合は、最初に削除されます。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR（ノードが newChild ノードのタイプの子を許可しないタイプである場合、または挿入するノードがノードの祖先のいずれかである場合）
- WRONG_DOCUMENT_ERR（newChild がノードを作成したドキュメントとは異なるドキュメントから作成された場合）
- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- NOT_FOUND_ERR（refChild がノードの子ではない場合）

構文

```
public org.w3c.dom.Node insertBefore( org.w3c.dom.Node newChild,
                                      org.w3c.dom.Node refChild);
```

パラメータ	説明
newChild	挿入するノード。
refChild	参照ノード（新しいノードが前に挿入されるノード）。

isNodeFlag()

説明

ノード・フラグ情報が設定されている場合は、true を戻します。

構文

```
public boolean isNodeFlag( int flag);
```

パラメータ	説明
flag	フラグ。

isSupported()

説明

DOM インプリメンテーションによって固有の機能が実装され、その機能がノードによってサポートされているかどうかをテストします。機能がサポートされている場合は `true`、サポートされていない場合は `false` を戻します。

構文

```
public boolean isSupported( String feature,
                            String version);
```

パラメータ	説明
feature	テスト対象の機能。
version	テスト対象の機能のバージョン。

print()

説明

ノードの内容を出力先に書き込みます。`IOException` が発生します。次の表に、オプションを示します。

構文	説明
public void print(java.io.OutputStream out);	ノードの内容を指定された出力ストリームに書き込みます。
public void print(java.io.OutputStream out, String enc);	ノードの内容を指定されたエンコーディング済出力ストリームに書き込みます。
public void print(java.io.PrintWriter out);	指定されたプリント・ライターを使用して、ノードの内容を書き込みます。

パラメータ	説明
out	出力。
enc	出力に使用するエンコーディング。

readExternal()

説明

writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- IOException（入力ストリームの読込み中にエラーが発生した場合）
- ClassNotFoundException（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

removeChild()

説明

oldChild に指定された子ノードを子のリストから削除します。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- NOT_FOUND_ERR（oldChild がノードの子ではない場合）

構文

```
public org.w3c.dom.Node removeChild( org.w3c.dom.Node oldChild);
```

パラメータ	説明
oldChild	削除するノード。

removeEventListener()

説明

イベント・ターゲット（ノード）からイベント・リスナーを削除します。

構文

```
public void removeEventListener( String type,
                                 org.w3c.dom.events.EventListener listener,
                                 boolean useCapture);
```

パラメータ	説明
type	リスナーを登録するイベントの型。
listener	リスナー・オブジェクト。
useCapture	リスナーが取得を開始する必要があるかどうかを示すフラグ。

replaceChild()

説明

子のリスト内の子ノード oldChild を newChild で置換し、置換したノードを戻します。newChild がすでにツリー内に存在する場合は、最初に削除されます。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR（ノードが newChild ノードのタイプの子を許可しないタイプである場合、または挿入するノードがノードの祖先のいずれかである場合）
- WRONG_DOCUMENT_ERR（newChild がノードを作成したドキュメントとは異なるドキュメントから作成された場合）
- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- NOT_FOUND_ERR（oldChild がノードの子ではない場合）

構文

```
public org.w3c.dom.Node replaceChild( org.w3c.dom.Node newChild,
                                      org.w3c.dom.Node oldChild);
```

パラメータ	説明
newChild	子のリストに挿入する新しいノード。
oldChild	リスト内で置換されるノード。

reportSAXEvents()

説明

DOM ツリーの SAX イベントをレポートします。SAXException が発生します。

構文

```
public void reportSAXEvents( org.xml.sax.ContentHandler cntHandler);
```

パラメータ	説明
cntHandler	コンテンツ・ハンドラ。

resetNodeFlag()

説明

ノード・フラグ情報をリセットします。

構文

```
public void resetNodeFlag( int flag);
```

パラメータ	説明
flag	ノード・フラグ。

selectNodes()

説明

ツリーから、指定されたパターンに一致するノードを NodeList として戻します。このメソッドは、パターンに名前空間の接頭辞が含まれていないことを前提としています。検索中にエラーが発生した場合は、XSLException が発生します。次の表に、オプションを示します。

構文	説明
public org.w3c.dom.NodeList selectNodes(String pattern);	パターンを使用して検索します。

構文	説明
public org.w3c.dom.NodeList selectNodes(String pattern, NSResolver nsr);	パターンおよび名前空間リゾルバを使用して検索します。

パラメータ	説明
pattern	一致させる XSL パターン。
nsr	指定されたパターン内に現れるすべての接頭辞を解決するための名前空間リゾルバ。

selectSingleNode()

説明

ツリーから、指定されたパターンに一致する最初のノードを戻します。検索中にエラーが発生した場合は、`XSLException` が発生します。次の表に、オプションを示します。

構文	説明
public org.w3c.dom.Node selectSingleNode(String pattern);	パターンを使用して検索します。
public org.w3c.dom.Node selectSingleNode(String pattern, NSResolver nsr);	パターンおよび名前空間リゾルバを使用して検索します。

パラメータ	説明
pattern	一致させる XSL パターン。
nsr	指定されたパターン内に現れるすべての接頭辞を解決するための名前空間リゾルバ。

setDebugInfo()

説明

ノードのデバッグ情報を設定します。

構文

```
public void setDebugInfo( int line,
                          int col,
                          String sysid);
```

パラメータ	説明
line	行番号。
col	列番号。
sysid	システム識別子。

setNodeFlag()

説明

ノード・フラグ情報を設定します。

構文

```
public void setNodeFlag( int flag);
```

パラメータ	説明
flag	ノード・フラグ。

setNodeValue()

説明

ノードのタイプに応じて、ノードの値を設定します。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- DOMSTRING_SIZE_ERR（実装プラットフォーム上で DOMString 変数に収まらない文字が戻された場合）

構文

```
public void setNodeValue( String nodeValue);
```

パラメータ	説明
nodeValue	設定するノード値。

setPrefix()

説明

名前空間をサポートするノードのタイプによってオーバーライドされるノードの接頭辞を設定します。DOMException が発生します。

構文

```
public void setPrefix( String prefix);
```

パラメータ	説明
prefix	設定する接頭辞。

setProperty()

説明

ノードのプロパティを設定します。

構文

```
public void setProperty( String propName,
                         Object propValue);
```

パラメータ	説明
propName	プロパティ名。
propValue	プロパティの値。

transformNode()

説明

指定されたスタイルシートを使用してツリー内のノードを変換し、結果のドキュメント・フラグメントを戻します。XSLException が発生します。

構文

```
public org.w3c.dom.DocumentFragment transformNode( XSLStylesheet xsl);
```

パラメータ	説明
xsl	変換に使用する XSL スタイルシート。

valueOf()

説明

ツリーから、パターンに一致する最初のノードの値を選択します。次の表に、オプションを示します。検索中にエラーが発生した場合は、XSLException が発生します。

構文	説明
public String valueOf(String pattern);	パターンを使用して検索します。
public String valueOf(String pattern, NSResolver nsr);	パターンおよび名前空間リゾルバを使用して検索します。

パラメータ	説明
pattern	一致させる XSL パターン。
nsr	指定されたパターン内に現れるすべての接頭辞を解決するための名前空間リゾルバ。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	シリアル化 / 圧縮したストリームの書込みに使用する ObjectOutput ストリーム。

XMLNotation クラス

XMLNotation の説明

このクラスは、DOM の Notation インタフェースを実装し、Document Type Definition で宣言されている表記法を表します。

XMLNotation の構文

```
public class XMLNotation implements java.io.Externalizable
```

oracle.xml.parser.v2.XMLNotation

XMLNotation の実装済インタフェース

```
java.io.Externalizable, java.io.Serializable
```

XMLNotation のメソッド

表 2-19 XMLNotation のメソッドの概要

メソッド	説明
XMLNotation()（2-133 ページ）	デフォルトのコンストラクタです。
cloneNode()（2-133 ページ）	ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。
getNodeName()（2-133 ページ）	表記法名を戻します。
getNodeType()（2-134 ページ）	実際の実装オブジェクトの型を表すコードを戻します。
getPublicId()（2-134 ページ）	公開識別子を戻します。公開識別子が指定されていない場合は、null を戻します。
getSystemId()（2-134 ページ）	システム識別子を戻します。システム識別子が指定されていない場合は、null を戻します。
readExternal()（2-135 ページ）	writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
setPublicId()（2-135 ページ）	公開識別子を設定します。
setSystemId()（2-136 ページ）	システム識別子を設定します。
writeExternal()（2-136 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

XMLNotation()

説明

デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。ノードを非シリアル化して、シリアル化 / 圧縮したストリームから DOM ノードを作成するには、オブジェクトのハンドルを作成する必要があります。通常のすべての XMLElement の作成では、`XMLNotation()` を使用してください。

構文

```
public  XMLNotation();
```

cloneNode()

説明

ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。この複製ノードは親を持ちません（`parentNode` は `null` を戻します）。`Element` を複製すると、デフォルトの属性を表すために XML プロセッサによって生成された属性を含む、すべての属性およびその値がコピーされます。ただし、このメソッドでは、ノードに含まれるテキストは、子の `Text` ノードに含まれているため、ディープ・クローンでないかぎりコピーされません。他のタイプのノードを複製すると、単にそのノードのコピーが戻されます。

構文

```
public org.w3c.dom.Node cloneNode( boolean deep);
```

パラメータ	説明
deep	`true` の場合は、指定されたノードの下位サブツリーが再帰的に複製されます。`false` の場合は、ノードのみが（ノードが `Element` の場合は、その属性も）複製されます。

getNodeName()

説明

表記法名を戻します。

構文

```
public String getNodeName();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
public short getNodeType();
```

getPublicId()

説明

公開識別子を戻します。公開識別子が指定されていない場合は、null を戻します。

構文

```
public String getPublicId();
```

getSystemId()

説明

システム識別子を戻します。システム識別子が指定されていない場合は、null を戻します。

構文

```
public String getSystemId();
```

readExternal()

説明

writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- `IOException`（入力ストリームの読込み中にエラーが発生した場合）
- `ClassNotFoundException`（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput inArg);
```

パラメータ	説明
inArg	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

setPublicId()

説明

公開識別子を設定します。

構文

```
public void setPublicId( String pubid);
```

パラメータ	説明
pubid	設定する公開識別子。

setSystemId()

説明

システム識別子を設定します。

構文

```
public void setSystemId( String url);
```

パラメータ	説明
url	設定するシステム識別子。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	シリアル化 / 圧縮したストリームの書込みに使用する ObjectOutput ストリーム。

XMLNSNode クラス

XMLNSNode の説明

このクラスは、XMLNode を拡張して、名前空間名および子のサポートを追加します。

XMLNSNode の構文

```
public class XMLNSNode extends oracle.xml.parser.v2.XMLNode

oracle.xml.parser.v2.XMLNode
  |
  +--oracle.xml.parser.v2.XMLNSNode
```

XMLNSNode の実装済インタフェース

```
java.lang.Cloneable, java.io.Externalizable, java.io.Serializable
```

XMLNSNode のメソッド

表 2-20 XMLNSNode のメソッドの概要

メソッド	説明
XMLNSNode()（2-138 ページ）	新しい XMLNSNode を作成します。
addText()（2-139 ページ）	ノードにテキストを追加します。最後の子が Text ノードの場合は、最後の子にテキストを追加します。
appendChild()（2-139 ページ）	ノードの子のリストの最後に newChild ノードを追加します。
getChildNodes()（2-140 ページ）	ノードのすべての子を NodeList として戻します。
getFirstChild()（2-140 ページ）	ノードの最初の子を戻します。
getLastChild()（2-140 ページ）	ノードの最後の子を戻します。
getLocalName()（2-141 ページ）	名前空間をサポートするノードのタイプによってオーバーライドされたノードのローカル名を戻します。
getNamespaceURI()（2-141 ページ）	名前空間をサポートするノードのタイプによってオーバーライドされたノードの名前空間 URI を戻します。
getNodeName()（2-141 ページ）	ノードのタイプに応じて、ノードの名前を戻します。
getPrefix()（2-141 ページ）	名前空間をサポートするノードのタイプによってオーバーライドされたノードの接頭辞を戻します。

表 2-20 XMLNSNode のメソッドの概要（続き）

メソッド	説明
getText()（2-142 ページ）	要素に含まれている非マークアップ・テキストを戻します。
hasChildNodes()（2-142 ページ）	ノードが子を持っているかどうかを判断します。
insertBefore()（2-142 ページ）	既存の子ノードの前に子ノードを挿入します。
normalize()（2-143 ページ）	属性ノードを含め、ノードの下位サブツリーの全階層に存在するすべての Text ノードを正規形に変換します。正規形では、構造のみによって Text ノードが分離されます。
removeChild()（2-143 ページ）	oldChild に指定された子ノードを子のリストから削除します。
replaceChild()（2-144 ページ）	子のリスト内の子ノード oldChild を newChild で置換します。
setPrefix()（2-144 ページ）	名前空間をサポートするノードのタイプによってオーバーライドされるノードの接頭辞を設定します。

XMLNSNode()

説明

新しい XMLNSNode を作成します。

構文

```
protected  XMLNSNode( String tag);
```

パラメータ	説明
tag	ノード名。

addText()

説明

ノードにテキストを追加します。最後の子が Text ノードである場合は、最後の子にテキストを追加します。ノードにテキストを追加できない場合は、XMLDOMException が発生します。次の表に、オプションを示します。

構文	説明
public void addText(char[] ch, int start, int length);	文字配列からテキストを追加します。
public XMLNode addText(String str);	文字列からテキストを追加します。

パラメータ	説明
ch	追加する文字配列。
start	文字配列内の開始インデックス。
length	追加する文字数。
str	追加するテキスト。

appendChild()

説明

ノードの子のリストの最後に newChild ノードを追加し、そのノードを戻します。newChild がすでにツリー内に存在する場合は、最初に削除されます。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR（ノードが newChild ノードのタイプの子を許可しないタイプである場合、または追加するノードがノードの祖先のいずれかである場合）
- WRONG_DOCUMENT_ERR（newChild がノードを作成したドキュメントとは異なるドキュメントから作成された場合）
- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）

構文

```
public org.w3c.dom.Node appendChild( org.w3c.dom.Node newChild);
```

パラメータ	説明
newChild	追加するノード。これが DocumentFragment オブジェクトである場合は、ドキュメント・フラグメントの内容全体がノードの子のリストに移動されます。

getChildNodes()

説明

ノードのすべての子を NodeList として戻します。子が存在しない場合は、ノードを含まない NodeList になります。戻された NodeList の内容は、作成される基となったノード・オブジェクトの子ノードに変更があった場合、NodeList アクセッサが戻すノードにすぐに反映されるという点などからみて「最新」のもので、ノード内容の静的スナップショットではありません。これは、getElementsByTagName メソッドが戻す NodeList を含め、すべての NodeList に当てはまります。

構文

```
public org.w3c.dom.NodeList getChildNodes();
```

getFirstChild()

説明

ノードの最初の子を戻します。該当するノードが存在しない場合は、null を戻します。

構文

```
public org.w3c.dom.Node getFirstChild();
```

getLastChild()

説明

ノードの最後の子を戻します。該当するノードが存在しない場合は、null を戻します。

構文

```
public org.w3c.dom.Node getLastChild();
```

getLocalName()

説明

名前空間をサポートするノードのタイプによってオーバーライドされたノードのローカル名を戻します。

構文

```
public String getLocalName();
```

getNamespaceURI()

説明

名前空間をサポートするノードのタイプによってオーバーライドされたノードの名前空間 URI を戻します。

構文

```
public String getNamespaceURI();
```

getNodeName()

説明

ノードのタイプに応じて、ノードの名前を戻します。

構文

```
public String getNodeName();
```

getPrefix()

説明

名前空間をサポートするノードのタイプによってオーバーライドされたノードの接頭辞を戻します。

構文

```
public String getPrefix();
```

getText()

説明

要素に含まれている非マークアップ・テキストを戻します。テキスト要素の場合は、生データになります。子ノードを持つ要素の場合は、サブツリー全体を検索し、末端のテキスト要素ごとにテキストを追加して、サブツリーの XML マークアップを効果的に削除します。たとえば、XML 文書に「William Shakespeare」が含まれる場合、XMLDocument.getText は「William Shakespeare」を戻します。

構文

```
public String getText();
```

hasChildNodes()

説明

ノードが子を持っているかどうかを判別します。ノードが子を持つ場合は true、子を持っていない場合は false を戻します。

構文

```
public boolean hasChildNodes();
```

insertBefore()

説明

既存の子ノード refChild の前に newChild ノードを挿入し、挿入したノードを戻します。refChild が null の場合は、子のリストの最後に newChild を挿入します。newChild が DocumentFragment オブジェクトの場合は、すべての子が refChild の前に同じ順序で挿入されます。newChild がすでにツリー内に存在する場合は、最初に削除されます。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR（ノードが newChild ノードのタイプの子を許可しないタイプである場合、または挿入するノードがノードの祖先のいずれかである場合）
- WRONG_DOCUMENT_ERR（newChild がノードを作成したドキュメントとは異なるドキュメントから作成された場合）
- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- NOT_FOUND_ERR（refChild がノードの子ではない場合）

構文

```
public org.w3c.dom.Node insertBefore( org.w3c.dom.Node newChild,
                                      org.w3c.dom.Node refChild);
```

パラメータ	説明
newChild	子のリストに挿入する新しいノード。
refChild	参照ノード（新しいノードが前に挿入されるノード）。

normalize()

説明

属性ノードを含め、ノードの下位サブツリーの全階層に存在するすべての Text ノードを正規形に変換します。正規形では、構造（要素、コメント、処理命令、CDATA セクション、実体参照など）のみによって Text ノードが分離されるため、隣接した Text ノードや空の Text ノードは存在しません。このメソッドを使用すると、ドキュメントを、保存して再ロードした場合と同様に DOM 表示できるため、特定のドキュメント・ツリー構造に依存する操作（XPointer 検索など）を使用する場合に有効です。

構文

```
public void normalize();
```

removeChild()

説明

`oldChild` に指定された子ノードを子のリストから削除します。次の `DOMException` が発生します。

- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- NOT_FOUND_ERR（`oldChild` がノードの子ではない場合）

構文

```
public org.w3c.dom.Node removeChild( org.w3c.dom.Node oldChild);
```

パラメータ	説明
oldChild	削除するノード。

replaceChild()

説明

子のリスト内の子ノード oldChild を newChild で置換し、oldChild ノードを戻します。newChild がすでにツリー内に存在する場合は、最初に削除されます。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR（ノードが newChild ノードのタイプの子を許可しないタイプである場合、または挿入するノードがノードの祖先のいずれかである場合）
- WRONG_DOCUMENT_ERR（newChild がノードを作成したドキュメントとは異なるドキュメントから作成された場合）
- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- NOT_FOUND_ERR（oldChild がノードの子ではない場合）

構文

```
public org.w3c.dom.Node replaceChild( org.w3c.dom.Node newChild,
                                      org.w3c.dom.Node oldChild);
```

パラメータ	説明
newChild	子のリストに挿入する新しいノード。
oldChild	リスト内で置換されるノード。

setPrefix()

説明

名前空間をサポートするノードのタイプによってオーバーライドされるノードの接頭辞を設定します。

構文

```
public void setPrefix( String prefix);
```

パラメータ	説明
prefix	ノードの接頭辞。

XMLOutputStream クラス

XMLOutputStream の説明

このクラスは、出力ストリームを書き込み、XML エンコーディングを処理できます。

XMLOutputStream の構文

```
public class XMLOutputStream extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.parser.v2.XMLOutputStream
```

XMLOutputStream のフィールド

表 2-21 XMLOutputStream のフィールド

フィールド	構文	説明
COMPACT	public static int COMPACT	追加のインデントおよび新しい行なし。
DEFAULT	public static int DEFAULT	追加のインデントおよび新しい行なし。
PRETTY	public static int PRETTY	可読性を高めるためにインデントおよび新しい行を追加します。

XMLOutputStream のメソッド

表 2-22 XMLOutputStream のメソッドの概要

メソッド	説明
XMLOutputStream()（2-146 ページ）	ASCII 出力を作成します。
addIndent()（2-147 ページ）	出力用のインデント・レベルを設定します。
close()（2-147 ページ）	出力ストリームをクローズします。
flush()（2-147 ページ）	出力ストリームをフラッシュします。
getOutputStyle()（2-147 ページ）	現行の出力スタイルを戻します。
setEncoding()（2-148 ページ）	出力のキャラクタ・エンコーディングを設定します。
setOutputStyle()（2-148 ページ）	出力スタイルを設定します。

表 2-22 XMLOutputStream のメソッドの概要（続き）

メソッド	説明
write()（2-149 ページ）	出力ストリームの型に基づいて、文字を出力します。
writeChars()（2-149 ページ）	文字列を出力先に書き込みます。
writeIndent()（2-149 ページ）	インデントを書き込みます。
writeNewLine()（2-150 ページ）	新しい行を書き込みます。
writeQuotedString()（2-150 ページ）	引用符で囲んだ文字列を書き込みます。

XMLOutputStream()

説明

ASCII 出力を作成します。次の表に、オプションを示します。

構文	説明
public XMLOutputStream(java.io.OutputStream out);	OutputStream を使用して出力を作成します。
public XMLOutputStream(java.io.PrintWriter out);	PrintWriter を使用して出力を作成します。

パラメータ	説明
out	出力。

addIndent()

説明
出力用のインデント・レベルを設定します。

構文
```
public void addIndent( int offset);
```

パラメータ	説明
offset	インデント・レベル。

close()

説明
出力ストリームをクローズします。エラーが発生した場合は、IOException が発生します。

構文
```
public void close();
```

flush()

説明
出力ストリームをフラッシュします。エラーが発生した場合は、IOException が発生します。

構文
```
public void flush();
```

getOutputStyle()

説明
現行の出力スタイルを戻します。

構文
```
public int getOutputStyle();
```

setEncoding()

説明

出力のキャラクタ・エンコーディングを設定します。エンコーディングのタイプの設定でエラーが発生した場合は、IOException が発生します。

構文

```
public void setEncoding( String encoding,
                         boolean lendian,
                         boolean byteOrderMark);
```

パラメータ	説明
encoding	ストリームのエンコーディング。
lendian	エンコーディングのタイプがリトル・エンディアンかどうかを示すフラグ。
byteOrderMark	バイト順序マークが設定されているかどうかを示すフラグ。

setOutputStyle()

説明

出力スタイルを設定します。

構文

```
public void setOutputStyle( int style);
```

パラメータ	説明
style	出力スタイル。

write()

説明

出力ストリームの型に基づいて、文字を出力します。文字の書込み中にエラーが発生した場合は、IOException が発生します。

構文

```
public void write( int c);
```

パラメータ	説明
c	書き込む文字。

writeChars()

説明

文字列を出力先に書き込みます。文字列の書込み中にエラーが発生した場合は、IOException が発生します。

構文

```
public void writeChars( String str);
```

パラメータ	説明
str	出力ストリームに書き込む文字列。

writeIndent()

説明

インデントを書き込みます。文字列の書込み中にエラーが発生した場合は、IOException が発生します。

構文

```
public void writeIndent();
```

writeNewLine()

説明

新しい行を書き込みます。文字列の書込み中にエラーが発生した場合は、IOException が発生します。

構文

```
public void writeNewLine();
```

writeQuotedString()

説明

引用符で囲んだ文字列を書き込みます。文字列の書込み中にエラーが発生した場合は、IOException が発生します。

構文

```
public void writeQuotedString( String str);
```

パラメータ	説明
str	出力ストリームに書き込む文字列。

XMLPI クラス

XMLPI の説明

このクラスは、DOM の Processing Instruction インタフェースを実装します。`ProcessingInstruction`、`NodeFactory` および `DOMParser.setNodeFactory()` も参照してください。

XMLPI の構文

```
public class XMLPI implements java.io.Externalizable
```

`oracle.xml.parser.v2.XMLPI`

XMLPI のサブクラス

`XMLDeclPI`

XMLPI の実装済インタフェース

`java.io.Externalizable, java.io.Serializable`

XMLPI のメソッド

表 2-23 XMLPI のメソッドの概要

メソッド	説明
XMLPI()（2-152 ページ）	XMLPI の新しいインスタンスを作成します。
addText()（2-152 ページ）	ノードにテキスト文字列を追加し、更新したノードを戻します。
getNodeName()（2-152 ページ）	処理命令ノードの名前を戻します。
getNodeType()（2-152 ページ）	実際の実装オブジェクトのノードのタイプを戻します。
getTarget()（2-153 ページ）	処理命令のターゲットを戻します。
readExternal()（2-153 ページ）	writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。
reportSAXEvents()（2-153 ページ）	DOM ツリーの SAX イベントをレポートします。
writeExternal()（2-154 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

XMLPI()

説明

デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。ノードを非シリアル化して、シリアル化 / 圧縮したストリームから DOM ノードを作成するには、オブジェクトのハンドルを作成する必要があります。

構文

```
public  XMLPI();
```

addText()

説明

ノードにテキスト文字列を追加し、更新したノードを戻します。

構文

```
public XMLNode addText( String str);
```

パラメータ	説明
str	追加するテキスト文字列。

getNodeName()

説明

処理命令ノードの名前を戻します。

構文

```
public String getNodeName();
```

getNodeType()

説明

実際の実装オブジェクトのノードのタイプを戻します。

構文

```
public short getNodeType();
```

getTarget()

説明

処理命令のターゲットを戻します。XML は、このターゲットを、処理命令を開始するマークアップの後に続く最初のトークンとして定義します。

構文

```
public String getTarget();
```

readExternal()

説明

writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。次の例外が発生します。

- `IOException`（入力ストリームの読込み中にエラーが発生した場合）
- `ClassNotFoundException`（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

reportSAXEvents()

説明

DOM ツリーの SAX イベントをレポートします。`SAXException` が発生します。

構文

```
public void reportSAXEvents( org.xml.sax.ContentHandler cntHandler);
```

パラメータ	説明
cntHandler	コンテンツ・ハンドラ。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。圧縮ストリームの書込み中に例外が発生した場合、IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	圧縮ストリームの書込みに使用する ObjectOutput ストリーム。

XMLPrintDriver クラス

XMLPrintDriver の説明

このクラスは、PrintDriver インタフェースを実装します。

XMLPrintDriver の構文

```
public class XMLPrintDriver extends Object implements
oracle.xml.parser.v2.PrintDriver

java.lang.Object
  |
  +--oracle.xml.parser.v2.XMLPrintDriver
```

XMLPrintDriver の実装済インタフェース

PrintDriver

XMLPrintDriver のフィールド

表 2-24 XMLPrintDriver のフィールド

フィールド	構文	説明
out	protected XMLOutputStream out	XMLOutputStream オブジェクト。

XMLPrintDriver のメソッド

表 2-25 XMLPrintDriver のメソッドの概要

メソッド	説明
XMLPrintDriver()（2-156 ページ）	XMLPrintDriver のインスタンスを作成します。
close()（2-157 ページ）	出力ストリームまたはプリント・ライターをクローズします。
flush()（2-157 ページ）	出力ストリームまたはプリント・ライターをフラッシュします。
printAttribute()（2-157 ページ）	XMLAttr ノードを出力します。
printAttributeNodes()（2-158 ページ）	XMLElement の属性ごとに出力メソッドをコールします。

表 2-25 XMLPrintDriver のメソッドの概要（続き）

メソッド	説明
printCDATASection()（2-158 ページ）	XMLCDATA ノードを出力します。
printChildNodes()（2-158 ページ）	XMLNode の子ごとに出力メソッドをコールします。
printComment()（2-159 ページ）	XMLComment ノードを出力します。
printDoctype()（2-159 ページ）	DTD を出力します。
printDocument()（2-159 ページ）	XML 文書を出力します。
printDocumentFragment()（2-160 ページ）	空の XMLDocumentFragment オブジェクトを出力します。
printElement()（2-160 ページ）	XMLElement を出力します。
printEntityReference()（2-160 ページ）	XMLEntityReference ノードを出力します。
printProcessingInstruction()（2-161 ページ）	XMLPI ノードを出力します。
printTextNode()（2-161 ページ）	XMLText ノードを出力します。
setEncoding()（2-161 ページ）	出力ドライバのエンコーディングを設定します。

XMLPrintDriver()

説明

XMLPrintDriver のインスタンスを作成します。次の表に、オプションを示します。

構文	説明
public XMLPrintDriver(OutputStream os);	出力ストリームから XMLPrintDriver のインスタンスを作成します。
public XMLPrintDriver(PrintWriter pw);	プリント・ライターから XMLPrintDriver のインスタンスを作成します。

パラメータ	説明
os	OutputStream。
pw	PrintWriter。

close()

説明
出力ストリームまたはプリント・ライターをクローズします。

構文
```
public void close();
```

flush()

説明
出力ストリームまたはプリント・ライターをフラッシュします。

構文
```
public void flush();
```

printAttribute()

説明
XMLAttr ノードを出力します。

構文
```
public void printAttribute( XMLAttr attr);
```

パラメータ	説明
attr	XMLAttr ノード。

printAttributeNodes()

説明

XMLElement の属性ごとに出力メソッドをコールします。

構文

```
public final void printAttributeNodes( XMLElement elem);
```

パラメータ	説明
elem	属性が出力される要素。

printCDATASection()

説明

XMLCDATA ノードを出力します。

構文

```
public void printCDATASection( XMLCDATA cdata);
```

パラメータ	説明
cdata	XMLCDATA ノード。

printChildNodes()

説明

XMLNode の子ごとに出力メソッドをコールします。

構文

```
public final void printChildNodes( XMLNode node);
```

パラメータ	説明
node	子が出力されるノード。

printComment()

説明

XMLComment ノードを出力します。

構文

```
public void printComment( XMLComment comment);
```

パラメータ	説明
comment	コメント・ノード。

printDoctype()

説明

DTD を出力します。

構文

```
public void printDoctype( DTD dtd);
```

パラメータ	説明
dtd	出力する DTD。

printDocument()

説明

XML 文書を出力します。

構文

```
public void printDocument( XMLDocument doc);
```

パラメータ	説明
doc	出力するドキュメント。

printDocumentFragment()

説明

空の XMLDocumentFragment オブジェクトを出力します。

構文

```
public void printDocumentFragment( XMLDocumentFragment dfrag);
```

パラメータ	説明
dfrag	出力するドキュメント・フラグメント。

printElement()

説明

XMLElement を出力します。

構文

```
public void printElement( XMLElement elem);
```

パラメータ	説明
elem	出力する要素。

printEntityReference()

説明

XMLEntityReference ノードを出力します。

構文

```
public void printEntityReference( XMLEntityReference en);
```

パラメータ	説明
en	XMLEntityReference ノード。

printProcessingInstruction()

説明

XMLPI ノードを出力します。

構文

```
public void printProcessingInstruction( XMLPI pi);
```

パラメータ	説明
pi	XMLPI ノード。

printTextNode()

説明

XMLText ノードを出力します。

構文

```
public void printTextNode( XMLText text);
```

パラメータ	説明
text	Text ノード。

setEncoding()

説明

出力ドライバのエンコーディングを設定します。

構文

```
public void setEncoding( String enc);
```

パラメータ	説明
enc	出力するドキュメントのエンコーディング。

XMLRangeException クラス

XMLRangeException の説明

このクラスは、RangeException をカスタマイズします。

XMLRangeException の構文

```
public class XMLRangeException
```

oracle.xml.parser.v2.XMLRangeException

XMLRangeException のメソッド

XMLRangeException()

説明

XMLRangeException インスタンスを生成します。

構文

```
public  XMLRangeException(short code);
```

パラメータ	説明
code	

XMLText クラス

XMLText の説明

このクラスは、DOM の Text インタフェースを実装します。Text、NodeFactory および DOMParser.setNodeFactory() も参照してください。

XMLText の構文

```
public class XMLText implements java.io.Serializable, java.io.Externalizable
```

oracle.xml.parser.v2.XMLText

XMLText の実装済インタフェース

java.io.Externalizable, java.io.Serializable

XMLText のメソッド

表 2-26 XMLText のメソッドの概要

メソッド	説明
XMLText()（2-164 ページ）	XMLText のインスタンスを作成します。
addText()（2-164 ページ）	Text ノードのデータにテキストを追加します。
getData()（2-165 ページ）	インタフェースを実装するノードの文字データを戻します。
getNodeName()（2-165 ページ）	XMLText ノードの名前を戻します。
getNodeType()（2-165 ページ）	実際の実装オブジェクトの型を表すノードのタイプを戻します。
getNodeValue()（2-166 ページ）	Text ノードの文字列値を戻します。
isWhiteSpaceNode()（2-166 ページ）	Text ノードが空白ノードかどうかを確認します。
readExternal()（2-166 ページ）	writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。

表 2-26 XMLText のメソッドの概要（続き）

メソッド	説明
reportSAXEvents()（2-167 ページ）	DOM ツリーの SAX イベントをレポートします。
splitText()（2-167 ページ）	Text ノードを、指定されたオフセットで 2 つの Text ノードに分割します。
writeExternal()（2-168 ページ）	オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。

XMLText()

説明

デフォルトのコンストラクタです。このコンストラクタは DOM ノードの非シリアル化 / 解凍中にのみ使用されることに注意してください。ノードを非シリアル化して、シリアル化 / 圧縮したストリームから DOM ノードを作成するには、オブジェクトのハンドルを作成する必要があります。

構文

```
public  XMLText();
```

addText()

説明

appendData と同様に、Text ノードのデータにテキストを追加します。

構文

```
public void addText( char[] ch,
                     int start,
                     int length);
```

パラメータ	説明
ch	追加する文字配列。
start	開始インデックス。
length	文字配列の長さ。

getData()

説明

インタフェースを実装するノードの文字データを戻します。DOM インプリメンテーションでは、Text ノードに格納できるデータ量が制限されない場合があります。ただし、実装による制限のため、ノードのデータ全体が単一の DOMString に収まらない場合もあります。この場合、substringData をコールすると、データを適切なサイズのピースに分割して取得することができます。

次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）
- DOMSTRING_SIZE_ERR（実装プラットフォーム上で DOMString 変数に収まらない文字が戻された場合）

構文

```
public String getData();
```

getNodeName()

説明

XMLText ノードの名前を戻します。

構文

```
public String getNodeName();
```

getNodeType()

説明

実際の実装オブジェクトの型を表すノードのタイプを戻します。

構文

```
public short getNodeType();
```

getNodeValue()

説明

Text ノードの文字列値を戻します。値の取得時にエラーが発生した場合は、DOMException が発生します。

構文

```
public String getNodeValue();
```

isWhiteSpaceNode()

説明

Text ノードが空白ノードかどうかを確認します。

構文

```
public boolean isWhiteSpaceNode();
```

readExternal()

説明

writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに応じてオブジェクトをリストアします。このメソッドは、XMLText オブジェクトが独立したノードとして非シリアル化（または読込み）される場合にコールされ、他の DOM ノードからはコールされません。次の例外が発生します。

- IOException（入力ストリームの読込み中にエラーが発生した場合）
- ClassNotFoundException（クラスが検出されない場合）

構文

```
public void readExternal( java.io.ObjectInput in);
```

パラメータ	説明
in	圧縮ストリームの読込みに使用する ObjectInput ストリーム。

reportSAXEvents()

説明

DOM ツリーの SAX イベントをレポートします。SAXException が発生します。

構文

```
public void reportSAXEvents(org.xml.sax.ContentHandler cntHandler);
```

パラメータ	説明
cntHandler	コンテンツ・ハンドラ。

splitText()

説明

Text ノードを、指定されたオフセットで 2 つの Text ノードに分割します。このため、この 2 つの Text ノードは兄弟関係となります。元のノードには、オフセット点までのすべての内容が含まれます。また、元のノードの直後に兄弟関係として挿入される新しいノードには、オフセット点以降のすべての内容が含まれます。

次の DOMException が発生します。

- INDEX_SIZE_ERR（指定されたオフセットが負か、または data 内の文字数を超える場合）
- NO_MODIFICATION_ALLOWED_ERR（ノードが読込み専用の場合）

構文

```
public org.w3c.dom.Text splitText( int offset);
```

パラメータ	説明
offset	分割点とするオフセット（0（ゼロ）から開始）。

writeExternal()

説明

オブジェクトの情報を含むバイナリ圧縮ストリームを作成することによって、オブジェクトの状態を保存します。圧縮ストリームの書込み中に例外が発生した場合、IOException が発生します。

構文

```
public void writeExternal( java.io.ObjectOutput out);
```

パラメータ	説明
out	圧縮ストリームの書込みに使用する ObjectOutput ストリーム。

3

Java での XML 処理（JAXP）

この章では、oracle.xml.parser.v2 パッケージに含まれる JAXP API について説明します。

この章の内容は次のとおりです。

- JXDocumentBuilder クラス
- JXDocumentBuilderFactory クラス
- JXSAXParser クラス
- JXSAXParserFactory クラス
- JXSAXTransformerFactory クラス
- JXTransformer クラス

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

JXDocumentBuilder クラス

JXDocumentBuilder の説明

このクラスは、XML 文書から DOM 文書インスタンスを取得する API を定義します。このクラスを使用すると、アプリケーション・プログラマは、XML から `org.w3c.dom.Document` を取得できます。

このクラスのインスタンスは、`DocumentBuilderFactory.newDocumentBuilder` メソッドから取得できます。このクラスのインスタンスを取得すると、様々な入力ソースから XML を解析できます。これらの入力ソースは、入力ストリーム、ファイル、URL および SAX 入力ソースです。

このクラスは、SAX API からいくつかのクラスを再利用することに注意してください。この場合、基礎となる DOM インプリメンテーションの実装者は、SAX パーサーを使用して XML 文書を `Document` に解析する必要はありません。既存の API を使用して、実装からアプリケーションに通信する必要があるのみです。

JXDocumentBuilder の構文

```
public class JXDocumentBuilder
```

`oracle.xml.jaxp.JXDocumentBuilder`

JXDocumentBuilder のメソッド

表 3-1 JXDocumentBuilder のメソッドの概要

メソッド	説明
getDOMImplementation() (3-3 ページ)	対応する DOM インプリメンテーション・オブジェクトを戻します。
isNamespaceAware() (3-3 ページ)	パーサーが名前空間を認識するかどうかを示します。
isValidating() (3-3 ページ)	パーサーが XML 文書の妥当性チェックを行うかどうかを示します。
newDocument() (3-4 ページ)	DOM ツリーを構築するための DOM Document オブジェクトの新しいインスタンスを取得します。
parse() (3-4 ページ)	指定された入力ソースのコンテンツを XML 文書として解析し、新しい DOM Document オブジェクトを戻します。

表 3-1 JXDocumentBuilder のメソッドの概要（続き）

メソッド	説明
setEntityResolver()（3-4 ページ）	エンティティ・リゾルバを指定して、解析される XML 文書に存在するエンティティを解決します。
setErrorHandler()（3-5 ページ）	解析される XML 文書に存在するエンティティの解決に使用するエラー・ハンドラを指定します。

getDOMImplementation()

説明

対応する DOM インプリメンテーション・オブジェクト（XML 文書を処理するためのオブジェクト）を戻します。DOM アプリケーションは、様々な実装から生成されるオブジェクトを使用できます。

構文

```
public org.w3c.dom.DOMImplementation getDOMImplementation();
```

isNamespaceAware()

説明

パーサーが名前空間を認識するように構成されているかどうかを示します。

構文

```
public boolean isNamespaceAware();
```

isValidating()

説明

パーサーが XML 文書の妥当性チェックを行うように構成されているかどうかを示します。

構文

```
public boolean isValidating();
```

newDocument()

説明

DOM ツリーを構築するための DOM Document オブジェクトの新しいインスタンスを取得します。

構文

```
public org.w3c.dom.Document newDocument();
```

parse()

説明

指定された入力ソースのコンテンツを XML 文書として解析し、新しい DOM Document オブジェクトを戻します。次の例外が発生します。

- `IOException`（I/O エラーが発生した場合）
- `SAXException`（解析エラーが発生した場合）
- `IllegalArgumentException`（入力ソースが `null` の場合）

構文

```
public org.w3c.dom.Document parse( org.xml.sax.InputSource is);
```

パラメータ	説明
is	解析されるコンテンツを含む InputSource。

setEntityResolver()

説明

`EntityResolver` を指定して、解析される XML 文書に存在するエンティティを解決します。これを `null` に設定すると、実際の実装では、独自の実装および動作が使用されます。

構文

```
public void setEntityResolver( org.xml.sax.EntityResolver er);
```

パラメータ	説明
er	エンティティ・リゾルバ。

setErrorHandler()

説明

解析される XML 文書に存在するエンティティの解決に使用するエラー・ハンドラを指定します。これを null に設定すると、実際の実装では、デフォルトの独自の実装および動作が使用されます。

構文

```
public void setErrorHandler( org.xml.sax.ErrorHandler eh);
```

パラメータ	説明
eh	エラー・ハンドラ。

JXDocumentBuilderFactory クラス

JXDocumentBuilderFactory の説明

このクラスは、ファクトリ API を定義します。このファクトリ API を使用すると、アプリケーションで XML 文書から DOM オブジェクト・ツリーを生成するパーサーを取得できます。

JXDocumentBuilderFactory の構文

```
public class JXDocumentBuilderFactory
```

oracle.xml.jaxp.JXDocumentBuilderFactory

JXDocumentBuilderFactory のフィールド

表 3-2 JXDocumentBuilderFactory のフィールド

フィールド	構文	説明
BASE_URL	public static final java.lang.String BASE_URL	エンティティの解析に使用されるベース URL。
DEBUG_MODE	public static final java.lang.String DEBUG_MODE	デバッグ・モード（Boolean.TRUE または Boolean.FALSE）。
DTD_OBJECT	public static final java.lang.String DTD_OBJECT	妥当性チェックに使用される DTD オブジェクト。
ERROR_ ENCODING	public static final java.lang.String ERROR_ENCODING	エラー・ストリームを介したエラー・レポートのエンコーディング（ERROR_STREAM が設定されている場合のみ）。
ERROR_STREAM	public static final java.lang.String ERROR_STREAM	エラーをレポートするためのエラー・ストリーム。オブジェクトは、OutputStream または PrintWriter です。エラー・ハンドラが設定されている場合、この属性は無視されます。
NODE_FACTORY	public static final java.lang.String NODE_FACTORY	カスタム・ノードを作成するための NodeFactory。

表 3-2 JXDocumentBuilderFactory のフィールド （続き）

フィールド	構文	説明
SCHEMA_OBJECT	public static final java.lang.String SCHEMA_OBJECT	妥当性チェックに使用されるスキーマ・オブジェクト。
SHOW_WARNINGS	public static final java.lang.String SHOW_WARNINGS	警告を無視するブール値（Boolean.TRUE または Boolean.FALSE）。
USE_DTD_ONLY_FOR_VALIDATION	public static final java.lang.String USE_DTD_ONLY_FOR_VALIDATION	`true` の場合、DTD オブジェクトは、妥当性チェックにのみ使用され、パーサー・ドキュメントには追加されません。

JXDocumentBuilderFactory のメソッド

表 3-3 JXDocumentBuilderFactory のメソッドの概要

メソッド	説明
JXDocumentBuilderFactory()（3-8 ページ）	デフォルトのコンストラクタです。
getAttribute()（3-8 ページ）	ユーザーが実際の実装の特定の属性を取得することができるようにします。
isExpandEntityReferences()（3-8 ページ）	ファクトリが実体参照ノードを拡張するパーサーを生成するように構成されているかどうかを示します。
isIgnoringComments()（3-9 ページ）	ファクトリがコメントを無視するパーサーを生成するように構成されているかどうかを示します。
isNamespaceAware()（3-9 ページ）	ファクトリが名前空間を認識するパーサーを生成するように構成されているかどうかを示します。
newDocumentBuilder()（3-9 ページ）	現在構成されているパラメータを使用して、DocumentBuilder の新しいインスタンスを作成します。
setAttribute()（3-10 ページ）	実際の実装に特定の属性を設定します。

JXDocumentBuilderFactory()

説明

デフォルトのコンストラクタです。

構文

```
public  JXDocumentBuilderFactory();
```

getAttribute()

説明

ユーザーが実際の実装の特定の属性を取得することを許可します。属性値を戻します。実際の実装が属性を認識しない場合、IllegalArgumentException が発生します。

構文

```
public Object getAttribute( String name);
```

パラメータ	説明
name	属性名。

isExpandEntityReferences()

説明

ファクトリが実体参照ノードを拡張するパーサーを生成するように構成されているかどうかを示します。常に、true を戻します、現在、実体参照の拡張を回避する方法はありません。

構文

```
public boolean isExpandEntityReferences();
```

isIgnoringComments()

説明

ファクトリがコメントを無視するパーサーを生成するように構成されているかどうかを示します。常に、false を戻します。現在、コメントを無視するようには構成できません。

構文

```
public boolean isIgnoringComments();
```

isNamespaceAware()

説明

ファクトリが名前空間を認識するパーサーを生成するように構成されているかどうかを示します。常に、true を戻します。現在、名前空間を無視する方法はありません。

構文

```
public boolean isNamespaceAware();
```

newDocumentBuilder()

説明

現在構成されているパラメータを使用して、DocumentBuilder の新しいインスタンスを作成します。リクエストされた構成を満たす DocumentBuilder を作成できない場合、ParserConfigurationException が発生します。

構文

```
public DocumentBuilder newDocumentBuilder();
```

setAttribute()

説明

実際の実装に特定の属性を設定します。実際の実装が属性を認識しない場合、`IllegalArgumentException` が発生します。

構文

```
public void setAttribute( String name,
                          Object value);
```

パラメータ	説明
name	属性名。
value	属性値。

JXSAXParser クラス

JXSAXParser の説明

このクラスは、org.xml.sax.XMLReader の実装クラスをラップする API を定義します。JAXP 1.0 では、このクラスは org.xml.sax.Parser インタフェースをラップしましたが、このインタフェースは、XMLReader に置き換えられました。

移行を簡略化するために、このクラスは、新しいメソッドのサポートのみでなく、同じ名前およびインタフェースも継続してサポートします。このクラスのインスタンスは、SAXParserFactory.newSAXParser メソッドから取得できます。このクラスのインスタンスを取得すると、様々な入力ソースから XML を解析できます。これらの入力ソースは、入力ストリーム、ファイル、URL および SAX 入力ストリームです。

static メソッドは、システム・プロパティの設定に基づいて新しいファクトリ・インスタンスを作成します。プロパティが定義されていない場合は、プラットフォームのデフォルトを使用します。

作成するファクトリ実装を制御するシステム・プロパティを、「javax.xml.style.TransformFactory」といいます。このプロパティは、抽象クラスの具体的なサブクラスのクラス名を指定します。プロパティが定義されていない場合は、プラットフォームのデフォルトを使用します。

コンテンツが、基礎となるパーサーによって解析されると、指定された HandlerBase のメソッドがコールされます。

JXSAXParser の構文

```
public class JXSAXParser
```

oracle.xml.jaxp.JXSAXParser

JXSAXParser のメソッド

表 3-4 JXSAXParser のメソッドの概要

メソッド	説明
getProperty()（3-12 ページ）	XMLReader の実際の実装にリクエストされたプロパティの値を戻します。
getXMLReader()（3-12 ページ）	クラスの実装によってカプセル化された XMLReader を戻します。
isNamespaceAware()（3-13 ページ）	パーサーが名前空間を認識するように構成されているかどうかを示します。

表 3-4 JXSAXParser のメソッドの概要（続き）

メソッド	説明
isValidating()（3-13 ページ）	パーサーが XML 文書の妥当性チェックを行うように構成されているかどうかを示します。
setProperty()（3-13 ページ）	基礎となる XMLReader の実装に特定のプロパティを設定します。

getProperty()

説明

org.xml.sax.XMLReader の実際の実装にリクエストされたプロパティの値を戻します。`org.xml.sax.XMLReader#getProperty` も参照してください。次の例外が発生します。

- `SAXNotRecognizedException`（基礎となる XMLReader がプロパティ名を認識しない場合）
- `SAXNotSupportedException`（基礎となる XMLReader がプロパティ名を認識するが、プロパティをサポートしない場合）

構文

```
public java.lang.Object getProperty( String name);
```

パラメータ	説明
name	取得するプロパティの名前。

getXMLReader()

説明

クラスの実装によってカプセル化された XMLReader を戻します。

構文

```
public XMLReader getXMLReader();
```

isNamespaceAware()

説明

パーサーが名前空間を認識するように構成されているかどうかを示します。パーサーが名前空間を認識する場合は `true`、認識しない場合は `false` を戻します。

構文

```
public boolean isNamespaceAware();
```

isValidating()

説明

パーサーが XML 文書の妥当性チェックを行うように構成されているかどうかを示します。

構文

```
public boolean isValidating();
```

setProperty()

説明

基礎となる XMLReader の実装に特定のプロパティを設定します。
`org.xml.sax.XMLReader#setProperty` も参照してください。次の例外が発生します。

- `SAXNotRecognizedException`（基礎となる XMLReader がプロパティ名を認識しない場合）
- `SAXNotSupportedException`（基礎となる XMLReader がプロパティ名を認識するが、プロパティをサポートしない場合）

構文

```
public void setProperty( String name,
                         Object value);
```

パラメータ	説明
name	設定するプロパティの名前。
value	設定するプロパティの値。

JXSAXParserFactory クラス

JXSAXParserFactory の説明

このクラスは、ファクトリ API を定義します。このファクトリ API を使用すると、アプリケーションで XML 文書から SAX ベースのパーサーを構成および取得して、XML 文書を解析できます。

構文

```
public class JXSAXParserFactory
```

oracle.xml.jaxp.JXSAXParserFactory

JXSAXParserFactory のメソッド

表 3-5 JXSAXParserFactory のメソッドの概要

メソッド	説明
JXSAXParserFactory() (3-14 ページ)	デフォルトのコンストラクタです。
getFeature() (3-15 ページ)	XMLReader の実際の実装にリクエストされたプロパティの値を戻します。
isNamespaceAware() (3-15 ページ)	ファクトリが、名前空間を認識するパーサーを生成するように構成されているかどうかを示します。
newSAXParser() (3-15 ページ)	現在構成されているパラメータを使用して、SAXParser の新しいインスタンスを作成します。
setFeature() (3-16 ページ)	基礎となる XMLReader の実装に特定の機能を設定します。

JXSAXParserFactory()

説明

デフォルトのコンストラクタです。

構文

```
public  JXSAXParserFactory();
```

getFeature()

説明

XMLReader の実際の実装にリクエストされたプロパティの値を戻します。`org.xml.sax.XMLReader#getProperty` も参照してください。次の例外が発生します。

- `SAXNotRecognizedException`（基礎となる XMLReader がプロパティ名を認識しない場合）
- `SAXNotSupportedException`（基礎となる XMLReader がプロパティ名を認識するが、プロパティをサポートしない場合）

構文

```
public boolean getFeature( String name);
```

パラメータ	説明
name	取得するプロパティの名前。

isNamespaceAware()

説明

ファクトリが、名前空間を認識するパーサーを生成するように構成されているかどうかを示します。ファクトリが、名前空間を認識するパーサー用に構成されている場合は `true`、構成されていない場合は `false` を戻します。

構文

```
public boolean isNamespaceAware();
```

newSAXParser()

説明

現在構成されているパラメータを使用して、SAXParser の新しいインスタンスを作成します。リクエストされた構成を満たすパーサーを作成できない場合は、`ParserConfigurationException` が発生します。

構文

```
public SAXParser newSAXParser();
```

setFeature()

説明

基礎となる XMLReader の実装に特定の機能を設定します。
`org.xml.sax.XMLReader#setFeature` も参照してください。次の例外が発生します。

- `SAXNotRecognizedException`（基礎となる XMLReader がプロパティ名を認識しない場合）
- `SAXNotSupportedException`（基礎となる XMLReader がプロパティ名を認識するが、プロパティをサポートしない場合）

構文

```
public void setFeature( String name, boolean value);
```

パラメータ	説明
name	設定する機能の名前。
value	設定する機能の値。

JXSAXTransformerFactory クラス

JXSAXTransformerFactory の説明

このクラスは、JXSAXTransformerFactory インスタンスを使用して、Transformer オブジェクトおよび Template オブジェクトを作成できます。

作成するファクトリ実装を判別するシステム・プロパティは、「javax.xml.transform.TransformerFactory」といいます。このプロパティは、TransformerFactory 抽象クラスの具体的なサブクラスを指定します（オラクル社の場合は、JXSAXTransformerFactory）。プロパティが定義されていない場合は、プラットフォームのデフォルトを使用します。

このクラスは、SAX 固有のファクトリ・メソッドも提供します。Transformer オブジェクト作成用および Template オブジェクト作成用の 2 種類のコンテンツ・ハンドラを提供します。

アプリケーションでは、変換中に使用される XMLReader にエラー・ハンドラまたはエンティティ・リゾルバを設定する必要がある場合、URIResolver を使用して、（getXMLReader とともに）XMLReader への参照を提供する SAXSource を戻す必要があります。

JXSAXTransformerFactory の構文

```
public class JXSAXTransformerFactory
```

`oracle.xml.jaxp.JXSAXTransformerFactory`

JXSAXTransformerFactory のメソッド

表 3-6 JXSAXTransformerFactory のメソッドの概要

メソッド	説明
JXSAXTransformerFactory()（3-18 ページ）	デフォルトのコンストラクタです。
getAssociatedStylesheet()（3-19 ページ）	xml-stylesheet 処理命令を介して対応付けられたスタイルシート仕様を取得します。
getAttribute()（3-19 ページ）	ユーザーが実際の実装の特定の属性を取得することができるようにします。
getErrorListener()（3-20 ページ）	TransformerFactory に対する現行のエラー・イベント・ハンドラ（`null` にはなりません）を戻します。
getFeature()（3-20 ページ）	機能の値を検索します。

表 3-6 JXSAXTransformerFactory のメソッドの概要（続き）

メソッド	説明
getURIResolver()（3-20 ページ）	URI 解決のための変換中にデフォルトで使用されるオブジェクトを取得します。
newTemplates()（3-21 ページ）	ソースを処理し、ソースのコンパイル済表現である Template オブジェクトに変換します。
newTemplatesHandler()（3-21 ページ）	SAX ContentHandler イベントを処理して Template オブジェクトに変換が可能な TemplatesHandler オブジェクトを取得します。
newTransformer()（3-22 ページ）	新しい Transformer オブジェクトを生成します。
newTransformerHandler()（3-22 ページ）	TransformerHandler オブジェクトを生成します。
newXMLFilter()（3-23 ページ）	XMLFilter を作成します。
setAttribute()（3-24 ページ）	実際の実装に特定の属性を設定します。
setErrorListener()（3-24 ページ）	TransformerFactory に対するエラー・イベント・リスナーを設定します。これは、変換命令の処理に使用され、変換自体には使用されません。
setURIResolver()（3-25 ページ）	xsl:import または xsl:include で使用される URI 解決のための変換中にデフォルトで使用されるオブジェクトを設定します。

JXSAXTransformerFactory()

説明

デフォルトのコンストラクタです。

構文

```
public  JXSAXTransformerFactory();
```

getAssociatedStylesheet()

説明

source パラメータおよび他のパラメータに指定されたドキュメントに一致する、xml-stylesheet 処理命令（http://www.w3.org/TR/xml-stylesheet/ を参照）を介して対応付けられたスタイルシート仕様を取得します。TransformerFactory へ渡すのに適した Source オブジェクトを戻します。複数のスタイルシートを戻すことができることに注意してください。この場合、これらのスタイルシートは、単一のスタイルシートのインポートまたはカスケードのリストと同様に適用されます。

構文

```
public Source getAssociatedStylesheet( Source source,
                                       String media,
                                       String title,
                                       String charset);
```

パラメータ	説明
source	XML ソース文書。
media	一致させるメディア属性。null の場合もあります。その場合は、優先テンプレートが使用されます（代替なし）。
title	一致させるタイトル属性の値。null の場合もあります。
charset	一致させるキャラクタ・セット属性の値。null の場合もあります。

getAttribute()

説明

ユーザーが実際の実装の特定の属性を取得することができるようにします。属性値を戻します。実際の実装が属性を認識しない場合は、IllegalArgumentException が発生します。

構文

```
public java.lang.Object getAttribute( String name);
```

パラメータ	説明
name	属性名。

getErrorListener()

説明

TransformerFactory に対する現行のエラー・イベント・ハンドラ（null にはなりません）を戻します。

構文

```
public ErrorListener getErrorListener();
```

getFeature()

説明

機能の値を検索します。機能の現在の状態（true または false）を戻します。機能名は任意の絶対 URI です。

構文

```
public boolean getFeature( String name);
```

パラメータ	説明
name	機能名（絶対 URI）。

getURIResolver()

説明

document()、xsl:import() または xsl:include() で使用される URI 解決のための変換中に、デフォルトで使用されるオブジェクトを取得します。setURIResolver を使用して設定された URIResolver を戻します。

構文

```
public URIResolver getURIResolver();
```

newTemplates()

説明

ソースを処理し、ソースのコンパイル済表現である Template オブジェクトに変換します。変換に使用可能な Template オブジェクト（null にはなりません）を戻します。この Template オブジェクトは、同時に複数のスレッドで使用できます。Template オブジェクトを作成すると、実行時の変換のパフォーマンスを低下させることなく、TransformerFactory によって変換命令のパフォーマンスの最適化を詳細に行うことができます。解析中にメソッドが Template オブジェクトの作成に失敗した場合は、TransformerConfigurationException が発生します。

構文

```
public Templates newTemplates( Source source);
```

パラメータ	説明
source	URL、入力ストリームなどを保持するオブジェクト。

newTemplatesHandler()

説明

SAX ContentHandler イベントを処理して Template オブジェクトに変換が可能な TemplatesHandler オブジェクトを取得します。TransformerHandler（SAX 解析イベントに対するコンテンツ・ハンドラとして使用可能）に対する非 null の参照を戻します。TemplatesHandler を作成できない場合は、TransformerConfigurationException が発生します。

構文

```
public TemplatesHandler newTemplatesHandler();
```

newTransformer()

説明

新しい Transformer オブジェクトを生成します。シングル・スレッドでの変換の実行に使用可能な Transformer オブジェクト（null にはなりません）を戻します。同時に実行している複数のスレッドでこのオブジェクトを使用しないように注意する必要があります。異なるスレッドでは、異なる TransformerFactory を同時に使用する必要があります。解析中に Template オブジェクトの作成に失敗した場合は、TransformerConfigurationException が発生します。次の表に、オプションを示します。

構文	説明
public Transformer newTransformer();	結果に対するソースのコピーを実行する、新しい Transformer オブジェクトを作成します。
public Transformer newTransformer(Source source);	ソースを処理して Transformer オブジェクトに変換します。

パラメータ	説明
source	URL、入力ストリームなどを保持するオブジェクト。

newTransformerHandler()

説明

TransformerHandler オブジェクトを生成します。TransformerHandler（SAX 解析イベントを変換可能）に対する非 null の参照を戻します。変換は、恒等（またはコピー）変換として定義されます（一連の SAX 解析イベントを DOM ツリーにコピーするなど）。TransformerHandler を作成できない場合は、TransformerConfigurationException が発生します。次の表に、オプションを示します。

構文	説明
public TransformerHandler newTransformerHandler();	SAX ContentHandler イベントを処理して結果を戻すことが可能な TransformerHandler オブジェクトを生成します。

構文	説明
public TransformerHandler newTransformerHandler(Source source);	SAX ContentHandler イベントを処理して、source 引数で指定された変換命令に基づいた結果を戻すことが可能な TransformerHandler オブジェクトを生成します。
public TransformerHandler newTransformerHandler(Templates templates);	SAX ContentHandler イベントを処理して、templates 引数で指定された変換命令に基づいた結果を戻すことが可能な TransformerHandler オブジェクトを生成します。

パラメータ	説明
source	変換命令のソース。
templates	コンパイル済の変換命令。

newXMLFilter()

説明

XMLFilter を作成します。XMLFilter オブジェクトまたは null（この機能がサポートされていない場合）を戻します。XMLFilter を作成できない場合は、TransformerConfigurationException が発生します。次の表に、オプションを示します。

構文	説明
public XMLFilter newXMLFilter(Source source);	指定されたソースを変換命令として使用する XMLFilter を作成します。
public XMLFilter newXMLFilter(Templates templates);	指定されたテンプレートを変換命令として使用する XMLFilter を作成します。

パラメータ	説明
source	変換命令のソース。
templates	コンパイル済の変換命令。

setAttribute()

説明

実際の実装に特定の属性を設定します。コンテキストの属性は、実装で提供されるオプションとして定義されます。

構文

```
public void setAttribute( String name,
                          Object value);
```

パラメータ	説明
name	属性名。
value	属性値。

setErrorListener()

説明

TransformerFactory に対するエラー・イベント・リスナーを設定します。これは、変換命令の処理に使用され、変換自体には使用されません。リスナーが null の場合は、IllegalArgumentException が発生します。

構文

```
public void setErrorListener(
               javax.xml.transform.ErrorListener listener);
```

パラメータ	説明
listener	新しいエラー・リスナー。

setURIResolver()

説明

xsl:import または xsl:include で使用される URI 解決のための変換中にデフォルトで使用されるオブジェクトを設定します。

構文

```
public void setURIResolver( javax.xml.transform.URIResolver resolver);
```

パラメータ	説明
name	URIResolver インタフェースを実装するオブジェクトまたは `null`。

JXTransformer クラス

JXTransformer の説明

このクラスのインスタンスは、ソース・ツリーを結果ツリーに変換できます。

このクラスのインスタンスは、TransformerFactory.newTransformer メソッドから取得できます。このインスタンスは、様々なソースから XML を処理し、変換出力を様々なシンクに書き込むために使用できます。

このクラスのオブジェクトは、同時に実行している複数のスレッドでは使用できません。異なる Transformer は、異なるスレッドで同時に使用できます。

Transformer は、複数回使用できます。パラメータおよび出力のプロパティは、変換にまたがって保持されます。

JXTransformer の構文

```
public class JXTransformer
```

oracle.xml.jaxp.JXTransformer

JXTransformer のメソッド

表 3-7 JXTransformer のメソッドの概要

メソッド	説明
JXTransformer()（3-27 ページ）	JXTransformer オブジェクトを作成します。
clearParameters()（3-27 ページ）	setParameter を使用して設定されたすべてのパラメータを消去します。
getErrorListener()（3-28 ページ）	実際の変換でのエラー・イベント・リスナーを取得します。
getOutputProperties()（3-28 ページ）	実際の変換での出力プロパティのコピーを取得します。
getOutputProperty()（3-29 ページ）	実際の変換での出力プロパティの文字列値を戻します。
getParameter()（3-29 ページ）	明示的に設定されたパラメータを戻します。
getURIResolver()（3-30 ページ）	URI の解決に使用されるオブジェクトを戻します。
setErrorListener()（3-30 ページ）	実際の変換でのエラー・イベント・リスナーを設定します。
setOutputProperties()（3-30 ページ）	実際の変換での出力プロパティを設定します。

表 3-7 JXTransformer のメソッドの概要（続き）

メソッド	説明
setOutputProperty()（3-31 ページ）	実際の変換での出力プロパティを設定します。
setParameter()（3-32 ページ）	実際の変換でのパラメータを追加します。
setURIResolver()（3-32 ページ）	document() で使用された URI の解決に使用されるオブジェクトを設定します。
transform()（3-33 ページ）	ソース・ツリーを処理して出力結果を戻します。

JXTransformer()

説明

JXTransformer オブジェクトを作成します。次の表に、オプションを示します。

構文	説明
public JXTransformer();	デフォルトのコンストラクタです。
public JXTransformer(oracle.xml.parser.v2.XSLStylesheet templates);	XSL スタイルシートを使用してソースを変換する JXTransformer オブジェクトを作成します。

パラメータ	説明
templates	XSL スタイルシートまたはテンプレート。

clearParameters()

説明

setParameter を使用して設定されたすべてのパラメータを消去します。

構文

```
public void clearParameters();
```

getErrorListener()

説明

実際の変換でのエラー・イベント・リスナーを取得します。このエラー・リスナーは null にはなりません。

構文

```
public javax.xml.transform.ErrorListener getErrorListener();
```

getOutputProperties()

説明

実際の変換での出力プロパティのコピーを取得します。

戻されるプロパティには、ユーザーおよびスタイルシートによって設定されたプロパティが含まれている必要があります。これらのプロパティは、W3C の XSL Transformations（XSLT）勧告のセクション 16 で指定されたデフォルトのプロパティによって「デフォルト」に指定されています。ユーザーまたはスタイルシートによって固有に設定されたプロパティは、ベースの Properties リストに含まれる必要があります。また、固有に設定されていない XSLT のデフォルトのプロパティは、デフォルトの Properties リストに含まれる必要があります。そのため、getOutputProperties().getProperty() は、スタイルシートまたはデフォルトのプロパティの setOutputProperty()、setOutputProperties() によって設定されたすべてのプロパティを取得しますが、getOutputProperties().get() は、setOutputProperty()、setOutputProperties() によって明示的に設定されたプロパティまたはスタイルシートのプロパティのみを取得します。戻される Properties オブジェクトの変更は、変換に含まれるプロパティには影響しません。

引数キーが認識されず、名前空間で修飾されていない場合、プロパティは無視され、その動作は setOutputProperties() に影響しません。

構文

```
public java.util.Properties getOutputProperties();
```

getOutputProperty()

説明

実際の変換での出力プロパティの文字列値を戻します。プロパティが検索されない場合は、null を戻します。指定されたプロパティは、setOutputProperty を使用して設定されたプロパティまたはスタイルシートに指定されたプロパティです。プロパティがサポートされていない場合は、IllegalArgumentException が発生します。

構文

```
public String getOutputProperty( String name);
```

パラメータ	説明
name	出力プロパティ名を指定する非 null の文字列。名前空間で修飾されている場合があります。

getParameter()

説明

setParameter() または setParameters() を使用して明示的に設定されたパラメータを戻します。指定された名前を持つパラメータが検索されない場合は null を戻します。このメソッドは、デフォルトのパラメータ値を戻しません。デフォルトのパラメータ値は、変換処理中にノードのコンテキストが評価されるまで判別できません。

構文

```
public Object getParameter( String name);
```

パラメータ	説明
name	パラメータ名。

getURIResolver()

説明

document() などで使用された URI の解決に使用されるオブジェクトを戻します。URIResolver インタフェースを実装するオブジェクトまたは null を取得します。リスナーが null の場合は、IllegalArgumentException が発生します。

構文

```
public javax.xml.transform.URIResolver getURIResolver();
```

setErrorListener()

説明

実際の変換でのエラー・イベント・リスナーを設定します。

構文

```
public void setErrorListener(javax.xml.transform.ErrorListener listener)
```

パラメータ	説明
listener	新しいエラー・リスナー。

setOutputProperties()

説明

実際の変換での出力プロパティを設定します。これらのプロパティは、xsl:output を使用してテンプレートに設定されたプロパティをオーバーライドします。引数キーが認識されず、名前空間で修飾されていない場合は、IllegalArgumentException が発生します。

このファンクションに対する引数が null の場合は、事前に設定されているすべてのプロパティが削除され、Template オブジェクトに定義された値に戻ります。

修飾されたプロパティのキー名を、2 つの部分の文字列（中カッコ（{}）で囲まれた名前空間 URI とその後に続くローカル名）として渡します。名前に null の URL が含まれている場合、文字列にはローカル名のみが含まれます。アプリケーションは、名前の最初の文字が「{」文字であるかどうかをテストすることによって、非 null の URI であるかどうかを確認できます。

たとえば、URI およびローカル名が、
<xyz:foo xmlns:xyz="http://xyz.foo.com/yada/baz.html"/> で定義された要素から取得された場合、修飾名は「{http://xyz.foo.com/yada/baz.html}foo」になります。接頭辞は使用されないことに注意してください。

構文

```
public void setOutputProperties( java.util.Properties oformat);
```

パラメータ	説明
oformat	実際の変換での同じプロパティをオーバーライドするために使用される、一連の出力プロパティ。

setOutputProperty()

説明

実際の変換での出力プロパティを設定します。修飾されたプロパティ名を、2 つの部分の文字列（中カッコ（{}）で囲まれた名前空間 URI とその後に続くローカル名）として渡します。名前に `null` の URL が含まれている場合、文字列にはローカル名のみが含まれます。アプリケーションは、名前の最初の文字が「{」文字であるかどうかをテストすることによって、非 `null` の URI であるかどうかを確認できます。

たとえば、URI およびローカル名が、
<xyz:foo xmlns:xyz="http://xyz.foo.com/yada/baz.html"/> で定義された要素から取得された場合、修飾名は「{http://xyz.foo.com/yada/baz.html}foo」になります。接頭辞は使用されないことに注意してください。

`setOutputProperties(Properties)` に渡された Properties オブジェクトは、このメソッドのコールによっては影響されません。

プロパティがサポートされておらず、名前空間で修飾されていない場合は、`IllegalArgumentException` が発生します。

構文

```
public void setOutputProperty(java.lang.String name,
                              java.lang.String value)
```

パラメータ	説明
name	出力プロパティ名を指定する非 `null` の文字列。名前空間で修飾されている場合があります。
value	出力プロパティの非 `null` の文字列値。

setParameter()

説明

実際の変換でのパラメータを追加します。修飾された名前を、2 つの部分の文字列（中カッコ（{}）で囲まれた名前空間 URI とその後に続くローカル名）として渡します。名前に null の URL が含まれている場合、文字列にはローカル名のみが含まれます。アプリケーションは、名前の最初の文字が「{」文字であるかどうかをテストすることによって、非 null の URI であるかどうかを確認できます。

たとえば、URI およびローカル名が、<xyz:foo xmlns:xyz="http://xyz.foo.com/yada/baz.html"/> で定義された要素から取得された場合、修飾名は「{http://xyz.foo.com/yada/baz.html}foo」になります。接頭辞は使用されないことに注意してください。

構文

```
public void setParameter( String name,
                          Object value);
```

パラメータ	説明
name	パラメータ名。中カッコ（{}）内の名前空間 URI で始まる場合があります。
value	値オブジェクト。すべての有効な Java オブジェクトを指定できます。適切なオブジェクト強制を提供するか、または拡張機能で使用するために単純にオブジェクトを渡すかどうかは、プロセッサによって異なります。

setURIResolver()

説明

document() で使用された URI の解決に使用するオブジェクトを設定します。resolver 引数が null の場合、URIResolver の値が消去され、デフォルトの動作が使用されます。現在、document() ファンクションの URIResolver はサポートされていません。

構文

```
public void setURIResolver( javax.xml.transform.URIResolver resolver);
```

パラメータ	説明
resolver	URIResolver インタフェースを実装するオブジェクトまたは null。

transform()

説明

ソース・ツリーを処理して出力結果を戻します。変換中にリカバリ不能なエラーが発生した場合は、TransformerException が発生します。

構文

```
public void transform( javax.xml.transform.Source xmlSource,
                       javax.xml.transform.Result outputTarget);
```

パラメータ	説明
xmlSource	ソース・ツリーの入力。
outputTarget	出力ターゲット。

4

Java での XSLT 処理

XSLT Processors は、XSLT スタイルシートによって指定された変換を使用して、XML 文書をその他のテキスト・フォーマットに変換できます。

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

oraxsl クラス

oraxsl の説明

このクラスは、複数の XML 文書で適用されるスタイルシートのコマンドライン・インタフェースを提供します。このインタフェースには、動作を指定する多くのコマンドライン・オプションがあります。

oraxsl の構文

```
public class oraxsl extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.parser.v2.oraxsl
```

oraxsl の使用方法

```
java oraxsl options* source? stylesheet? result?
```

表 4-1 oraxsl のコマンドライン・オプション

コマンド	説明
-w	警告の表示。
-e <error log>	エラーを書き込むファイル。
-l <xml file list>	変換するファイルのリスト。
-d <directory>	変換するファイルを含むディレクトリ。
-x <source extension>	対象外とする拡張子。
-i <source extension>	対象とする拡張子。
-s <stylesheet>	使用するスタイルシート。
-r <result extension>	結果ファイルの拡張子。
-o <result extension>	結果ファイルを保存するディレクトリ。
-p <param list>	パラメータのリスト。
-t <# of threads>	使用するスレッド数。
-v	冗長モード。

oraxsl のメソッド

表 4-2 oraxsl のメソッドの概要

メソッド	説明
oraxsl()（4-3 ページ）	クラス・コンストラクタです。
main()（4-3 ページ）	oraxsl ドライバを起動します。

oraxsl()

説明

クラス・コンストラクタです。

構文

```
public  oraxsl();
```

main()

説明

oraxsl ドライバを起動します。

構文

```
public static void main( String[] args);
```

パラメータ	説明
args	コマンドライン引数。

XPathException クラス

XPathException の説明

このクラスは、XPath の処理中に例外が発生したことを示します。

XPathException の構文

```
public class XPathException extends oracle.xml.parser.v2.XSLException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--oracle.xml.util.XMLException
                    |
                    +--oracle.xml.parser.v2.XSLException
                          |
                          +--oracle.xml.parser.v2.XPathException
```

XPathException の実装済インタフェース

```
java.io.Serializable
```

XPathException のメソッド

表 4-3 XPathException のメソッドの概要

メソッド	説明
getErrorID()（4-4 ページ）	エラー ID を取得します。
getMessage()（4-5 ページ）	エラー・メッセージを取得します。

getErrorID()

説明

エラー ID を取得します。

構文

```
public int getErrorID();
```

getMessage()

説明

エラー・メッセージを取得します。次の表に、オプションを示します。

構文	説明
public String getMessage();	エラー ID およびエラー・パラメータからエラー・メッセージを作成します。`java.lang.Throwable` クラスの `getMessage()` をオーバーライドします。
public String getMessage(XMLError err);	パラメータとして送信される XMLError に基づいて、ローカライズされたメッセージを取得します。

パラメータ	説明
err	エラー・メッセージを取得するために使用される XMLError クラス。

XSLException クラス

XSLException の説明

このクラスは、XSL 変換中に例外が発生したことを示します。

XSLException の構文

```
public class XSLException extends oracle.xml.util.XMLException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--oracle.xml.util.XMLException
                    |
                    +--oracle.xml.parser.v2.XSLException
```

XSLException のダイレクト・サブクラス

```
XPathException
```

XSLException の実装済インタフェース

```
java.io.Serializable
```

XSLException のメソッド

XSLException()

説明

新しい XSL の例外を生成します。

構文

```
public  XSLException( String mesg);
```

XSLExtensionElement クラス

XSLExtensionElement の説明

このクラスは、拡張要素の基となる要素です。

XSLExtensionElement の構文

```
public class XSLExtensionElement
```

oracle.xml.parser.v2.XSLExtensionElement

XSLExtensionElement のメソッド

表 4-4 XSLExtensionElement のメソッドの概要

メソッド	説明
XSLExtensionElement()（4-7 ページ）	デフォルトのコンストラクタです。
getAttributeTemplateValue()（4-8 ページ）	属性値をテンプレートとして取得します。
getAttributeValue()（4-8 ページ）	属性値を取得します。
getChildNodes()（4-9 ページ）	拡張要素の子ノードのリストを取得します。
processAction()（4-9 ページ）	拡張要素の本体を実行します。
processContent()（4-9 ページ）	拡張要素の内容を処理します。

XSLExtensionElement()

説明

デフォルトのコンストラクタです。

構文

```
public  XSLExtensionElement();
```

getAttributeTemplateValue()

説明

属性値をテンプレートとして取得します。

構文

```
protected final String getAttributeTemplateValue(
          XSLTContext context,
          String namespace,
          String name);
```

パラメータ	説明
context	XSLT コンテキスト。
namespace	属性の名前空間。
name	属性名。

getAttributeValue()

説明

属性値を取得します。

構文

```
protected final String getAttributeValue( String namespace,
                                          String name);
```

パラメータ	説明
namespace	属性の名前空間。
name	属性名。

getChildNodes()

説明

拡張要素の子ノードを NodeList として取得します。

構文

```
protected final java.util.Vector getChildNodes();
```

processAction()

説明

拡張要素の本体を実行します。

構文

```
public void processAction( XSLTContext context);
```

パラメータ	説明
context	XSLT コンテキスト。

processContent()

説明

拡張要素の内容を処理します。

構文

```
protected final void processContent(XSLTContext context);
```

パラメータ	説明
context	XSLT コンテキスト。

XSLProcessor クラス

XSLProcessor の説明

このクラスは、事前に作成された XSLStylesheet を使用して、入力された XML 文書を変換するメソッドを提供します。影響される変換は、XSLT 1.0 仕様によって指定されています。

XSLProcessor の構文

```
public class XSLProcessor
```

oracle.xml.parser.v2.XSLProcessor

XSLProcessor のメソッド

表 4-5 XSLProcessor のメソッドの概要

メソッド	説明
XSLProcessor()（4-11 ページ）	デフォルトのコンストラクタです。
getParam()（4-11 ページ）	最上位のスタイルシートのパラメータ値を取得します。
newXSLStylesheet()（4-11 ページ）	XSLStylesheet を作成します。
processXSL()（4-12 ページ）	入力された XML 文書を変換します。
removeParam()（4-15 ページ）	最上位のスタイルシートのパラメータ値を削除します。
resetParams()（4-16 ページ）	すべてのパラメータ設定をリセットします。
setBaseURL()（4-16 ページ）	ベース URL を設定して、include/import の href 属性を解決します。
setEntityResolver()（4-16 ページ）	エンティティ・リゾルバを設定して、include/import の href 属性を解決します。
setErrorStream()（4-17 ページ）	警告を出力するための出力ストリームを設定します。
setLocale()（4-17 ページ）	エラー・レポートのロケールを設定します。
setParam()（4-17 ページ）	最上位のスタイルシートのパラメータ値を設定します。
showWarnings()（4-18 ページ）	警告を出力するかどうかを判別します。

XSLProcessor()

説明
デフォルトのコンストラクタです。

構文
```
public  XSLProcessor();
```

getParam()

説明
最上位のスタイルシートのパラメータ値を取得します。

構文
```
public Object getParam( String uri,
                        String name);
```

パラメータ	説明
uri	パラメータの名前空間 URI。
name	パラメータのローカル名。

newXSLStylesheet()

説明
新しい XSLStylesheet を作成し、戻します。XSLException が発生します。次の表に、オプションを示します。

構文	説明
public XSLStylesheet newXSLStylesheet(InputStream xsl);	指定された入力ストリームを使用して、XSLStylesheet を作成します。
public XSLStylesheet newXSLStylesheet(Reader xsl);	指定されたリーダーを使用して、XSLStylesheet を作成します。

構文	説明
public XSLStylesheet newXSLStylesheet(java.net.URL xsl);	指定された URL を使用して、XSLStylesheet を作成します。
public XSLStylesheet newXSLStylesheet(XMLDocument xsl);	指定された XML 文書を使用して、XSLStylesheet を作成します。

パラメータ	説明
xsl	XML 入力。

processXSL()

説明

入力された XML 文書を変換します。エラーが発生した場合は、`XSLException` が発生します。次の表に、オプションを示します。

構文	説明
public XMLDocumentFragment processXSL(XSLStylesheet xsl, InputStream xml, URL ref);	指定された入力ストリームおよびスタイルシートを使用して、入力された XML 文書を変換します。XMLDocumentFragment を戻します。
public XMLDocumentFragment processXSL(XSLStylesheet xsl, Reader xml, URL ref);	指定されたリーダーおよびスタイルシートを使用して、入力された XML 文書を変換します。XMLDocumentFragment を戻します。
public XMLDocumentFragment processXSL(XSLStylesheet xsl, URL xml, URL ref);	指定された URL およびスタイルシートを使用して、入力された XML 文書を変換します。XMLDocumentFragment を戻します。

構文	説明
public XMLDocumentFragment processXSL(XSLStylesheet xsl, XMLDocument xml);	指定された XMLDocument およびスタイルシートを使用して、入力された XML 文書を変換します。XMLDocumentFragment を戻します。
public void processXSL(XSLStylesheet xsl, XMLDocument xml, org.xml.sax.ContentHandler handler);	指定された XMLDocument、スタイルシートおよびコンテンツ・ハンドラを使用して、入力された XML 文書を変換します。
public XMLDocumentFragment processXSL(XSLStylesheet xsl, XMLDocumentFragment xml);	指定された XMLDocumentFragment およびスタイルシートを使用して、入力された XML 文書を変換します。XMLDocumentFragment を戻します。
public void processXSL(XSLStylesheet xsl, XMLDocumentFragment xml, OutputStream out);	指定された XMLDocumentFragment、スタイルシートおよび出力ストリームを使用して、入力された XML を変換します。
public void processXSL(XSLStylesheet xsl, XMLDocumentFragment xml, printWriter pw);	指定された XMLDocumentFragment、スタイルシートおよびプリント・ライターを使用して、入力された XML を変換します。
public void processXSL(XSLStylesheet xsl, XMLDocumentFragment xml, XMLDocumentHandler handlerXML);	指定された XMLDocumentFragment、スタイルシートおよび XML 文書ハンドラを使用して、入力された XML 文書を変換します。XSLT の結果がドキュメント・フラグメントであるため、XMLDocumentHandler のファンクション（setDocumentLocator、startDocument、endDocument、setDoctype、endDoctype、setXMLDecl および setTextDecl）はコールされません。
public void processXSL(XSLStylesheet xsl, XMLDocument xml, OutputStream out);	指定された XMLDocument、スタイルシートおよび出力ストリームを使用して、入力された XML 文書を変換します。

構文	説明
public void processXSL(XSLStylesheet xsl, XMLDocument xml, java.io.PrintWriter pw);	指定された XMLDocument、スタイルシートおよびプリント・ライターを使用して、入力された XML 文書を変換します。
public void processXSL(XSLStylesheet xsl, XMLDocument xml, XMLDocumentHandler handlerXML);	指定された XMLDocument、スタイルシートおよび XML 文書ハンドラを使用して、入力された XML 文書を変換します。変換の出力は、XMLDocumentHandler を介してレポートされます。XSLT の結果がドキュメント・フラグメントであるため、XMLDocumentHandler のファンクション（setDocumentLocator、startDocument、endDocument、setDoctype、endDoctype、setXMLDecl および setTextDecl）はコールされません。
public XMLDocumentFragment processXSL(XSLStylesheet xsl, XMLElement xml);	指定された XMLElement およびスタイルシートを使用して、入力された XML 文書を変換します。XML 文書のフラグメントを戻します。
public void processXSL(XSLStylesheet xsl, XMLElement inp, org.xml.sax.ContentHandler handler);	指定された XMLElement、スタイルシートおよびコンテンツ・ハンドラを使用して、入力された XML 文書を変換します。変換の出力は、コンテンツ・ハンドラを介してレポートされます。XSLT の結果がドキュメント・フラグメントであるため、ContentHandler のファンクション（setDocumentLocator、startDocument および endDocument）はコールされません。
public void processXSL(XSLStylesheet xsl, XMLElement xml, OutputStream out);	指定された XMLElement、スタイルシートおよび出力ストリームを使用して、入力された XML を変換します。
public void processXSL(XSLStylesheet xsl, XMLElement xml, PrintWriter pw);	指定された XMLElement、スタイルシートおよびプリント・ライターを使用して、入力された XML を変換します。

構文	説明
public void processXSL(XSLStylesheet xsl, XMLElement xml, XMLDocumentHandler handlerXML);	指定された XMLElement、スタイルシートおよびドキュメント・ハンドラを使用して、入力された XML 文書を変換します。XSLT の結果がドキュメント・フラグメントであるため、XMLDocumentHandler のファンクション（setDocumentLocator、startDocument、endDocument、setDoctype、endDoctype、setXMLDecl および setTextDecl）はコールされません。

パラメータ	説明
xsl	変換に使用される XSL スタイルシート。
xml	変換される XML 入力。
ref	入力された XML ファイルの外部エンティティを解決するための参照 URL。
handler	コンテンツ・ハンドラ。
out	結果が出力される出力ストリーム。
pw	結果が出力されるプリント・ライター。
handlerXML	XMLDocument ハンドラ。

removeParam()

説明

最上位のスタイルシートのパラメータ値を削除します。エラーが発生した場合は、`XSLException` が発生します。

構文

```
public void removeParam( String uri,
                         String name);
```

パラメータ	説明
uri	パラメータの URI。
name	パラメータ名。

resetParams()

説明

すべてのパラメータ設定をリセットします。エラーが発生した場合は、XSLException が発生します。

構文

```
public void resetParams();
```

setBaseURL()

説明

ベース URL を設定して、include/import の href 属性を解決します。ベース URL を使用する前に、エンティティ・リゾルバ（設定されている場合）が使用されます。setEntityResolver() も参照してください。

構文

```
public void setBaseURL(java.net.URL url);
```

パラメータ	説明
url	設定するベース URL。

setEntityResolver()

説明

エンティティ・リゾルバを設定して、include/import の href 属性を解決します。エンティティ・リゾルバが設定されていない場合、ベース URL（設定されている場合）が使用されます。

構文

```
public void setEntityResolver( org.xml.sax.EntityResolver eResolver);
```

パラメータ	説明
eResolver	エンティティ・リゾルバ。

setErrorStream()

説明

警告を出力するための出力ストリームを設定します。警告に対する出力ストリームが指定されていない場合、プロセッサは警告を出力しません。

構文

```
public final void setErrorStream(java.io.OutputStream out);
```

パラメータ	説明
out	エラーおよび警告に使用する出力ストリーム。

setLocale()

説明

アプリケーションは、これを使用して、エラー・レポート用のロケールを設定できます。

構文

```
public void setLocale( java.util.Locale locale);
```

パラメータ	説明
locale	設定するロケール。

setParam()

説明

最上位のスタイルシートのパラメータ値を設定します。パラメータ値は、有効な XPath 式が想定されます（したがって、文字列リテラル値を明示的に囲む必要があります）。param ファンクションは、XSLStylesheet の param ファンクションとともには使用できません。XSLProcessor の param ファンクションが使用される場合、XSLStylesheet ファンクションを使用して設定されたパラメータは無視されます。エラーが発生した場合は、`XSLException` が発生します。

構文

```
public void setParam( String uri,
                      String name,
                      Object value);
```

パラメータ	説明
uri	パラメータの URI。
name	パラメータ名。
value	パラメータ値。文字列は、下位互換性のために XPath 式として処理されます。

showWarnings()

説明

警告を出力するかどうかを判別します。

構文

```
public final void showWarnings( boolean flag);
```

パラメータ	説明
flag	警告を表示する必要があるかどうかを判別します。デフォルトでは、警告は出力されません。

XSLStylesheet クラス

XSLStylesheet の説明

このクラスは、テンプレート、キー、変数、属性セットなどの XSL スタイルシートに関する情報を保持します。同じスタイルシートの作成後、それを使用して、複数の XML 文書を変換できます。

XSLStylesheet の構文

```
public class XSLStylesheet
```

`oracle.xml.parser.v2.XSLStylesheet`

XSLStylesheet のフィールド

表 4-6 XSLStylesheet のフィールド

フィールド	構文	説明
output	public oracle.xml.parser.v2.XSLOutput output	出力。

XSLStylesheet のメソッド

表 4-7 XSLStylesheet のメソッドの概要

メソッド	説明
getDecimalFormat()（4-20 ページ）	スタイルシートに指定された 10 進フォーマット記号を戻します。
getOutputEncoding()（4-20 ページ）	`xsl:output` に指定されたエンコーディングの値を戻します。
getOutputMediaType()（4-20 ページ）	`xsl:output` に指定されたメディア・タイプの値を戻します。
getOutputProperties()（4-21 ページ）	`xsl:output` に指定された出力プロパティを java.util.Properties として戻します。
newTransformer()（4-21 ページ）	変換にスタイルシートを使用する JAXP Transformer オブジェクトを戻します。

表 4-7 XSLStylesheet のメソッドの概要（続き）

メソッド	説明
getContextNode()（4-22 ページ）	最上位のスタイルシートのパラメータ値を削除します。
getContextPosition()（4-23 ページ）	すべてのパラメータ設定をリセットします。
getContextPosition()（4-23 ページ）	最上位のスタイルシートのパラメータ値を設定します。

getDecimalFormat()

説明

スタイルシートに指定された、10 進フォーマット記号を戻します。

構文

```
public java.text.DecimalFormatSymbols getDecimalFormat( NSName nsname);
```

パラメータ	説明
nsname	XSL 10 進フォーマットからの修飾名。

getOutputEncoding()

説明

`xsl:output` に指定されたエンコーディングの値を戻します。

構文

```
public String getOutputEncoding();
```

getOutputMediaType()

説明

`xsl:output` に指定されたメディア・タイプの値を戻します。

構文

```
public jString getOutputMediaType();
```

getOutputProperties()

説明

xsl:output に指定された出力プロパティを java.util.Properties として戻します。

構文

```
public java.util.Properties getOutputProperties();
```

newTransformer()

説明

変換にスタイルシートを使用する JAXP Transformer オブジェクトを戻します。

構文

```
public javax.xml.transform.Transformer newTransformer();
```

XSLTContext クラス

XSLTContext の説明

このクラスは、XPath 処理のコンテキストです。

XSLTContext の構文

```
public class XSLTContext extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.parser.v2.XSLTContext
```

XSLTContext のメソッド

表 4-8 XSLTContext のメソッドの概要

メソッド	説明
getContextNode()（4-22 ページ）	現行のコンテキスト・ノードを戻します。
getContextPosition()（4-23 ページ）	現行のコンテキスト・ノードの位置を戻します。
getContextSize()（4-23 ページ）	現行のコンテキスト・ノードのサイズを戻します。
getError()（4-23 ページ）	エラーをレポートするための XMLError インスタンスを戻します。
getVariable()（4-23 ページ）	指定されたスタック・オフセットの変数を戻します。
reportCharacters()（4-24 ページ）	現行の出力ハンドラへ文字をレポートします。
reportNode()（4-24 ページ）	現行の出力ハンドラへ XMLNode をレポートします。
setError()（4-25 ページ）	XMLError を設定します。

getContextNode()

説明

現行のコンテキスト・ノードを戻します。

構文

```
public XMLNode getContextNode();
```

getContextPosition()

説明

現行のコンテキスト・ノードの位置を戻します。

構文

```
public int getContextPosition();
```

getContextSize()

説明

現行のコンテキスト・ノードのサイズを戻します。

構文

```
public int getContextSize();
```

getError()

説明

エラーをレポートするための XMLError インスタンスを戻します。

構文

```
public XMLError getError();
```

getVariable()

説明

指定されたスタック・オフセットの変数を戻します。

構文

```
public getVariable( NSName name,
                    int offset);
```

パラメータ	説明
name	変数名。
offset	変数のオフセット。

reportCharacters()

説明

現行の出力ハンドラへ文字をレポートします。

構文

```
public void reportCharacters( String data,
                              boolean disableoutesc);
```

パラメータ	説明
chars	出力される文字列。
disableoutesc	w3c.org の XML 1.0 仕様に定義されているとおり、文字のエスケープを無効または有効にするブール値。`true` は無効、`false` は有効を意味します。

reportNode()

説明

現行の出力ハンドラへ XMLNode をレポートします。

構文

```
public void reportNode( XMLNode node);
```

パラメータ	説明
node	出力されるノード。

setError()

説明

XMLError を設定します。

構文

```
public void setError(XMLError err);
```

パラメータ	説明
err	XMLError のインスタンス。

5

Java での圧縮

通常、XML ファイルではタグが繰り返されるため、XML 構造のストリームの圧縮および XML タグのトークン化が効果的になります。ドキュメントは、読込み完了前にリアルタイムで解析されるため、インターネット経由で 2 つのサブシステム間でドキュメントを交換する場合、圧縮および解凍のパフォーマンスが向上するというメリットが得られます。XML の圧縮では、XML データの構造情報および階層情報が圧縮フォーマットで保持されます。

この章では、圧縮を実行する `oracle.xml.parser.v2` パッケージ・クラスについて説明します。

この章の内容は次のとおりです。

- CXMLHandlerBase クラス
- CXMLParser クラス

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

CXMLHandlerBase クラス

CXMLHandlerBase の説明

SAX 圧縮は、SAX ハンドラを使用して実装されます。SAX ハンドラは、SAX イベントに基づいてデータを圧縮します。SAX 圧縮を使用するには、アプリケーションでインタフェースを実装し、`Parser.setDocumentHandler()` を介して SAX パーサーに登録する必要があります。

CXMLHandlerBase の構文

```
public class CXMLHandlerBase implements oracle.xml.parser.v2.XMLDocumentHandler
```

`oracle.xml.comp.CXMLHandlerBase`

CXMLHandlerBase の実装済インタフェース

`oracle.xml.parser.v2.XMLDocumentHandler`

CXMLHandlerBase のメソッド

表 5-1 CXMLHandlerBase のメソッドの概要

メソッド	説明
CXMLHandlerBase()（5-3 ページ）	新しい CXMLHandlerBase を作成します。
cDATASection()（5-4 ページ）	CDATA セクションの通知を受け取ります。
characters()（5-4 ページ）	要素内の文字データの通知を受け取ります。
comment()（5-5 ページ）	コメントの通知を受け取ります。
endDoctype()（5-5 ページ）	DTD の終わりの通知を受け取ります。
endDocument()（5-6 ページ）	ドキュメントの終わりの通知を受け取ります。
endElement()（5-6 ページ）	要素の終わりの通知を受け取ります。
endPrefixMapping()（5-6 ページ）	接頭辞マッピングの有効範囲を終了します。
getCXMLContext()（5-7 ページ）	圧縮に使用された CXML コンテキストを戻します。
getProperty()（5-7 ページ）	プロパティの値を検索し、戻します。
ignorableWhitespace()（5-7 ページ）	要素の内容にある無視可能な空白の通知を受け取ります。

表 5-1 CXMLHandlerBase のメソッドの概要 （続き）

メソッド	説明
processingInstruction()（5-8 ページ）	処理命令の通知を受け取ります。
setDoctype()（5-8 ページ）	DTD の通知を受け取るように、DTD を設定します。
setDocumentLocator()（5-8 ページ）	ドキュメント・イベント用の Locator オブジェクトを受け取るように、Locator を設定します。
setError()（5-9 ページ）	XMLError ハンドラの通知を受け取るように、XMLError ハンドラを設定します。
setProperty()（5-9 ページ）	プロパティの値を設定します。
setTextDecl()（5-10 ページ）	テキスト XML 宣言の通知を受け取るように、テキスト XML 宣言を設定します。
setXMLDecl()（5-10 ページ）	XML 宣言の通知を受け取るように、XML 宣言を設定します。
setXMLSchema()（5-11 ページ）	XMLSchema オブジェクトの通知を受け取るように、XMLSchema を設定します。
skippedEntity()（5-11 ページ）	スキップされたエンティティの通知を受け取ります。
startDocument()（5-12 ページ）	ドキュメントの始まりの通知を受け取ります。
startElement()（5-12 ページ）	要素の始まりの通知を受け取ります。
startPrefixMapping()（5-13 ページ）	URI の接頭辞マッピングの有効範囲を開始します。

CXMLHandlerBase()

説明

新しい CXMLHandlerBase を作成します。次の表に、オプションを示します。

構文	説明
public CXMLHandlerBase();	デフォルトのコンストラクタです。新しい CXMLHandlerBase を作成します。
public CXMLHandlerBase(ObjectOutput out);	ObjectOutput ストリームを使用して、CXMLHandlerBase を作成します。

構文	説明
public CXMLHandlerBase(oracle.xml.io.XMLObjectOutput out);	XMLObjectOutputStream を使用して、CXMLHandlerBase を作成します。

パラメータ	説明
out	出力ストリーム。

cDATASection()

説明

CDATA セクションの通知を受け取ります。パーサーは、CDATA セクションが検索されるたびに、このメソッドを 1 回コールします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void cDATASection( char[] cbuf,
                          int start,
                          int len);
```

パラメータ	説明
cbuf	CDATA セクション文字。
start	文字配列の開始位置。
len	文字配列のうちの使用する文字数。

characters()

説明

要素内の文字データの通知を受け取ります。

構文

```
public void characters( char[] cbuf,
                        int start,
                        int len);
```

パラメータ	説明
cbuf	文字。
start	文字配列の開始位置。
len	文字配列のうちの使用する文字数。

comment()

説明

コメントの通知を受け取ります。パーサーは、コメントが検索されるたびに、このメソッドを 1 回コールします。コメントは、主なドキュメント要素の前後で発生することに注意してください。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void comment( String text);
```

パラメータ	説明
text	コメント・データ（指定されていない場合は null）。

endDoctype()

説明

DTD の終わりの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void endDoctype();
```

endDocument()

説明

ドキュメントの終わりの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void endDocument();
```

endElement()

説明

要素の終わりの通知を受け取ります。

構文

```
public void endElement( oracle.xml.parser.v2.NSName elem);
```

パラメータ	説明
elem	要素。

endPrefixMapping()

説明

URI の接頭辞マッピングの有効範囲を終了します。

構文

```
public void endPrefixMapping( String prefix);
```

パラメータ	説明
prefix	マップされている接頭辞。

getCXMLContext()

説明
圧縮に使用された CXML コンテキストを戻します。

構文
```
public oracle.xml.comp.CXMLContext getCXMLContext();
```

getProperty()

説明
プロパティの値を検索し、戻します。プロパティ名は、任意の完全修飾 URI です。

構文
```
public java.lang.Object getProperty( String name);
```

パラメータ	説明
name	プロパティ名（完全修飾 URI）。

ignorableWhitespace()

説明
要素の内容にある無視可能な空白の通知を受け取ります。

構文
```
public void ignorableWhitespace( char[] cbuf,
                                 int start,
                                 int len);
```

パラメータ	説明
cbuf	XML 文書の文字。
start	配列の開始位置。
len	配列のうちの読み込む文字数。

processingInstruction()

説明

処理命令の通知を受け取ります。

構文

```
public void processingInstruction( String target,
                                   String data);
```

パラメータ	説明
target	処理命令のターゲット。
data	処理命令のデータ。

setDoctype()

説明

DTD の通知を受け取ることができるように、DTD を登録します。パーサーは、`startDocument()` をコールして、使用された DTD を登録した後、このメソッドをコールします。`org.xml.sax.SAXException` が発生します。この例外は、`SAXException` またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setDoctype( oracle.xml.parser.v2.DTD dtd);
```

パラメータ	説明
dtd	DTD ノード。

setDocumentLocator()

説明

ドキュメント・イベント用の Locator オブジェクトの通知を受け取ることができるように、Locator オブジェクトを登録します。デフォルトでは、何も実行されません。このメソッドは、サブクラス・イベントでオーバーライドできます。

構文

```
public void setDocumentLocator( org.xml.sax.Locator locator);
```

パラメータ	説明
locator	SAX ドキュメント・イベント用のロケータ。

setError()

説明

XMLError ハンドラの通知を受け取ることができるように、XMLError ハンドラを登録します。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setError( oracle.xml.parser.v2.XMLError he);
```

パラメータ	説明
he	XMLError オブジェクト。

setProperty()

説明

プロパティの値を設定します。プロパティ名は、任意の完全修飾 URI です。

構文

```
public void setProperty( String name,
                         Object value);
```

パラメータ	説明
name	プロパティ名（完全修飾 URI）。
value	リクエストされたプロパティの値。

setTextDecl()

説明

テキスト宣言の通知を受け取ることができるように、テキスト XML 宣言を登録します。パーサーは、テキスト XMLDecl ごとに、このメソッドをコールします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setTextDecl( String version,
                         String encoding);
```

パラメータ	説明
version	バージョン番号（指定されていない場合は null）。
encoding	エンコーディング名。

setXMLDecl()

説明

XML 宣言の通知を受け取ることができるように、XML 宣言を登録します。パーサーは、XMLDecl ごとに、このメソッドをコールします。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setXMLDecl( String version,
                        String standalone,
                        String encoding);
```

パラメータ	説明
version	バージョン番号。
standalone	スタンドアロン値（指定されていない場合は null）。
encoding	エンコーディング名（指定されていない場合は null）。

setXMLSchema()

説明

XMLSchema オブジェクトの通知を受け取ることができるように、XMLSchema を登録します。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setXMLSchema( Object s);
```

パラメータ	説明
s	XMLSchema オブジェクト。

skippedEntity()

説明

スキップされたエンティティの通知を受け取ります。パーサーは、エンティティがスキップされるたびに、このメソッドを 1 回コールします。妥当性を検証しないプロセッサは、宣言が検出されない場合（たとえば、エンティティが外部 DTD サブセットで宣言されたため）、エンティティをスキップします。external-general-entities プロパティおよび external-parameter-entities プロパティの値によっては、すべてのプロセッサが外部エンティティをスキップする場合があります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void skippedEntity( String name);
```

パラメータ	説明
name	スキップされたエンティティの名前。パラメータ・エンティティの場合は、「%」で始まり、外部 DTD サブセットの場合は、「[dtd]」という文字列になります。

startDocument()

説明

ドキュメントの始まりの通知を受け取ります。デフォルトでは、何も実行されません。このメソッドは、ドキュメントの始まりで特定のアクションを行うサブクラスでオーバーライドできます。

構文

```
public void startDocument();
```

startElement()

説明

要素の始まりの通知を受け取ります。

構文

```
public void startElement( oracle.xml.parser.v2.NSName elem,
                          oracle.xml.parser.v2.SAXAttrList attributes);
```

パラメータ	説明
elem	要素。
attributes	要素の属性。

startPrefixMapping()

説明

接頭辞 URI マッピングの有効範囲を開始します。

構文

```
public void startPrefixMapping( String prefix,
                                String uri);
```

パラメータ	説明
prefix	宣言される名前空間の接頭辞。
uri	接頭辞がマップされる名前空間の URI。

CXMLParser クラス

CXMLParser の説明

このクラスは、圧縮ストリームから SAX イベントを生成することによって、圧縮ストリームからの XML 文書の再生成を実装します。

CXMLParser の構文

```
public class CXMLParser
```

oracle.xml.comp.CXMLParser

CXMLParser のメソッド

表 5-2 CXMLHandlerBase のメソッドの概要

メソッド	説明
startPrefixMapping()（5-13 ページ）	圧縮ストリーム読込みのための新しい XML Parser オブジェクトを作成します。
CXMLParser()（5-14 ページ）	コンテンツ・ハンドラを取得します。
getContentHandler()（5-15 ページ）	現行のエラー・ハンドラを取得します。
getErrorHandler()（5-15 ページ）	圧縮ストリームを解析し、SAX イベントを生成します。
parse()（5-15 ページ）	圧縮ストリームを解析し、SAX イベントを生成します。
setContentHandler()（5-16 ページ）	コンテンツ・イベント・ハンドラを登録します。
setErrorHandler()（5-16 ページ）	エラー・イベント・ハンドラを登録します。

CXMLParser()

説明

圧縮ストリーム読込みのための新しい XML Parser オブジェクトを作成します。

構文

```
public  CXMLParser();
```

getContentHandler()

説明

コンテンツ・ハンドラを取得します。ハンドラが登録されていない場合は、null を戻します。

構文

```
public org.xml.sax.ContentHandler getContentHandler();
```

getErrorHandler()

説明

現行のエラー・ハンドラを取得します。ハンドラが登録されていない場合は、null を戻します。

構文

```
public org.xml.sax.ErrorHandler getErrorHandler();
```

parse()

説明

圧縮ストリームを解析し、SAX イベントを生成します。次の例外が発生します。

- SAXException（すべての SAX 例外）
- IOException（I/O 操作によるエラーが発生した場合）

構文

```
public void parse( String inFile);
```

パラメータ	説明
inFile	SAX イベントの再生成のために解析の必要がある入力ソース。

setContentHandler()

説明

コンテンツ・イベント・ハンドラを登録します。

構文

```
public void setContentHandler( org.xml.sax.ContentHandler handler);
```

パラメータ	説明
handler	コンテンツ・ハンドラ。

setErrorHandler()

説明

エラー・イベント・ハンドラを登録します。

構文

```
public void setErrorHandler( org.xml.sax.ErrorHandler handler);
```

パラメータ	説明
handler	エラー・ハンドラ。

6

XML Schema の処理

XML Schema Processor for Java のすべての機能は、`oracle.XML.parser.schema` パッケージに含まれています。

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

XMLSchema クラス

XMLSchema の説明

このクラスには、異なるターゲットの名前空間に対する一連のスキーマが含まれます。たとえば、XMLSchema オブジェクトは、XML 文書の検証のために XSDParser によって使用され、インポート済スキーマとして XSDBuilder によって使用されます。

XMLSchema の構文

```
public class XMLSchemaNode extends oracle.xml.parser.schema.XSDNode

oracle.xml.parser.schema.XSDNode
  |
  +--oracle.xml.parser.schema.XMLSchemaNode
```

XMLSchema のメソッド

表 6-1 XMLSchema のメソッドの概要

コンストラクタ	説明
XMLSchema() （6-3 ページ）	XMLSchema のコンストラクタです。
getAllTargetNS()（6-3 ページ）	スキーマに定義されたすべてのターゲットの名前空間を戻します。
getSchemaByTargetNS()（6-3 ページ）	指定された名前空間に対する schemaNode を戻します。
getSchemaTargetNS()（6-4 ページ）	最上位のスキーマのターゲットの名前空間を戻します。
getXMLSchemaNodeTable()（6-4 ページ）	XMLSchemaNode 表を戻します。
getXMLSchemaURLS()（6-4 ページ）	XMLSchema の URL を戻します。
printSchema()（6-5 ページ）	スキーマの情報を出力します。

XMLSchema()

説明

XMLSchema のコンストラクタです。XSDException が発生します。次の表に、オプションを示します。

構文	説明
public XMLSchema()	デフォルトのコンストラクタです。
public XMLSchema(int n);	schemaNode セットの初期サイズが指定されたスキーマを作成します。

パラメータ	説明
n	schemaNode セットの初期サイズ。

getAllTargetNS()

説明

スキーマに定義されたすべてのターゲットの名前空間を戻します。

構文

```
public java.lang.String[] getAllTargetNS();
```

getSchemaByTargetNS()

説明

指定された名前空間に対する schemaNode を戻します。

構文

```
public XMLSchemaNode getSchemaByTargetNS( String namespace);
```

パラメータ	説明
namespace	リクエストされたスキーマのターゲットの名前空間。

getSchemaTargetNS()

説明

最上位のスキーマのターゲットの名前空間を戻します。複数の最上位のスキーマが存在する場合、最後に作成されたスキーマを戻します。

構文

```
public String getSchemaTargetNS();
```

getXMLSchemaNodeTable()

説明

XMLSchemaNode 表をハッシュテーブルとして戻します。

構文

```
public java.util.Hashtable getXMLSchemaNodeTable();
```

getXMLSchemaURLS()

説明

XML Schema の URL を配列として戻します。

構文

```
public java.lang.String[] getXMLSchemaURLS();
```

printSchema()

説明

スキーマの情報を出力します。次の表に、オプションを示します。

構文	説明
public void printSchema();	スキーマの情報を出力します。
public void printSchema(boolean all);	組込み情報を含む、スキーマの情報を出力します。

パラメータ	説明
all	組込み情報を含むすべての情報を出力する必要があることを示すフラグ。

XMLSchemaNode クラス

XMLSchemaNode の説明

このクラスには、ターゲットの名前空間に存在する一連の最上位のスキーマ・コンポーネントが含まれます。

XMLSchemaNode の構文

```
public class XMLSchemaNode extends oracle.xml.parser.schema.XSDNode

oracle.xml.parser.schema.XSDNode
  |
  +--oracle.xml.parser.schema.XMLSchemaNode
```

XMLSchemaNode のメソッド

表 6-2 XMLSchemaNode のメソッドの概要

メソッド	説明
XMLSchemaNode()（6-7 ページ）	XMLSchema のコンストラクタです。
getAttributeDeclarations()（6-7 ページ）	スキーマに存在するすべての最上位属性を戻します。
getComplexTypeSet()（6-7 ページ）	スキーマに存在するすべての最上位の complexType 要素を戻します。
getComplexTypeTable()（6-7 ページ）	complexType の定義を戻します。
getElementSet()（6-8 ページ）	スキーマに存在するすべての最上位要素を戻します。
getSimpleTypeSet()（6-8 ページ）	スキーマに存在するすべての最上位の simpleType 要素を戻します。
getSimpleTypeTable()（6-8 ページ）	simpleType の定義を戻します。
getTargetNS()（6-8 ページ）	スキーマのターゲットの名前空間を戻します。
getTypeDefinitionTable()（6-9 ページ）	型定義を戻します。

XMLSchemaNode()

説明

XMLSchema のコンストラクタです。

構文

```
public  XMLSchemaNode();
```

getAttributeDeclarations()

説明

スキーマに存在するすべての最上位属性を配列として戻します。

構文

```
public XSDAttribute getAttributeDeclarations();
```

getComplexTypeSet()

説明

スキーマに存在するすべての最上位の complexType 要素を配列として戻します。

構文

```
public XSDNode getComplexTypeSet();
```

getComplexTypeTable()

説明

complexType の定義をハッシュテーブルとして戻します。

構文

```
public java.util.Hashtable getComplexTypeTable();
```

getElementSet()

説明

スキーマに存在するすべての最上位要素を配列として戻します。

構文

```
public XSDNode getElementSet();
```

getSimpleTypeSet()

説明

スキーマに存在するすべての最上位の simpleType 要素を配列として戻します。

構文

```
public XSDNode getSimpleTypeSet();
```

getSimpleTypeTable()

説明

simpleType の定義をハッシュテーブルとして戻します。

構文

```
public java.util.Hashtable getSimpleTypeTable();
```

getTargetNS()

説明

スキーマのターゲットの名前空間を戻します。XSDNode クラスの XSDNode.getTargetNS() をオーバーライドします。

構文

```
public String getTargetNS();
```

getTypeDefinitionTable()

説明

型定義をハッシュテーブルとして戻します。

構文

```
public java.util.Hashtable getTypeDefinitionTable();
```

XSDAttribute クラス

XSDAttribute の説明

このクラスは、スキーマ属性の宣言を表します。

XSDAttribute の構文

```
public class XSDAttribute extends oracle.xml.parser.schema.XSDNode

oracle.xml.parser.schema.XSDNode
  |
  +--oracle.xml.parser.schema.XSDAttribute
```

XSDAttribute のメソッド

表 6-3 XSDAttribute のメソッドの概要

メソッド	説明
getDefaultVal()（6-11 ページ）	要素の場合は、「default」属性の値、および「use」属性に基づく「value」属性の値を戻します。
getFixedVal()（6-11 ページ）	要素の場合は、「fixed」属性の値、および「use」属性に基づく「value」属性の値を戻します。
getName()（6-11 ページ）	ノード名を戻します。
getRefLocalname()（6-11 ページ）	解決済の「ref」属性のローカル名を戻します。
getRefNamespace()（6-12 ページ）	解決済の「ref」属性の名前空間を戻します。
getRefState()（6-12 ページ）	refState を戻します。
getTargetNS()（6-12 ページ）	ターゲットの名前空間を戻します。
getType()（6-12 ページ）	ノードのタイプを戻します。
isRequired()（6-13 ページ）	属性が必要であるかどうかを確認します。

getDefaultVal()

説明

要素の場合は、「default」属性の値、および「use」属性に基づく「value」属性の値を戻します。

構文

```
public String getDefaultVal();
```

getFixedVal()

説明

要素の場合は、「default」属性のデフォルト値、および「use」属性に基づく「value」属性の値を戻します。

構文

```
public java.lang.String getFixedVal();
```

getName()

説明

ノード名を戻します。XSDNode クラスの XSDNode.getName() をオーバーライドします。

構文

```
public String getName();
```

getRefLocalname()

説明

解決済「ref」属性の refLocal 名を戻します。

構文

```
public String getRefLocalname();
```

getRefNamespace()

説明

解決済「ref」属性の RefNamespace を戻します。

構文

```
public String getRefNamespace();
```

getRefState()

説明

refState 値を戻します。戻り値は、TYPE_UNRESOLVED、TYPE_RESOLVED、REF_UNRESOLVED または REF_RESOLVED のいずれかになります。

構文

```
public int getRefState();
```

getTargetNS()

説明

ターゲットの名前空間を戻します。XSDNode クラスの getTargetNS メソッドをオーバーライドします。

構文

```
public String getTargetNS();
```

getType()

説明

simpleType または complexType のいずれかのノードのタイプを戻します。

構文

```
public XSDNode getType();
```

isRequired()

説明

属性が必要であるかどうかを確認します。

構文

```
public boolean isRequired();
```

XSDBuilder クラス

XSDBuilder の説明

このクラスは、XMLSchema ドキュメントから XMLSchema オブジェクトを作成します。XMLSchema オブジェクトは、最上位のスキーマ宣言と定義に対応した、オブジェクトのセット（情報セットの項目）です。スキーマ・ドキュメントは、解析され、DOM ツリーに変換される「XML」です。このスキーマの DOM ツリーは、次の順序で解析される「スキーマ」です。最初に、（存在する場合）スキーマ・オブジェクトを作成し、参照可能にします。次に、（存在する場合）対応する DOM ツリーに置換します。次に、最上位の宣言と定義は、現行のスキーマ情報セットの項目として登録されます。最後に、ツリーの最上位要素（情報セットの項目）が、解析されたスキーマになります。XMLSchema 結果オブジェクトは、オブジェクト（最上位の入力要素）のセット（情報セット）です。オブジェクトの内容は、数の情報（min/maxOccurs）を含む、SNode 型のノード / オブジェクトの前の低レベル element/group decls/refs に対応するノードを持つツリーです。

XSDBuilder の構文

```
public class XSDBuilder
```

XSDBuilder のメソッド

表 6-4 XSDBuilder のメソッドの概要

メソッド	説明
XSDBuilder()（6-15 ページ）	クラス・コンストラクタです。
build()（6-15 ページ）	XMLSchema オブジェクトまたはドキュメントを作成します。
getObject()（6-17 ページ）	XMLSchema オブジェクトを戻します。
setEntityResolver()（6-17 ページ）	import/include を解決するためのエンティティ・リゾルバを設定します。
setError()（6-17 ページ）	XMLError オブジェクトを設定します。
setLocale()（6-18 ページ）	エラー・レポートのロケールを設定します。

XSDBuilder()

説明

XSDBuilder のコンストラクタです。

構文

```
public XSDBuilder() throws XSDException;
```

build()

説明

XMLSchema オブジェクトまたはドキュメントを作成して戻します。Builder による XMLSchema オブジェクトの作成に失敗した場合は、例外が発生します。次の表に、オプションを示します。

構文	説明
public Object build(InputStream in, URL baseurl);	入力ストリームおよび URL から XMLSchema オブジェクトを作成します。
public Object build(Reader r, URL baseurl);	リーダーおよび URL から XMLSchema オブジェクトを作成します。
public Object build(String sysId);	システム ID から XMLSchema オブジェクトを作成します。
public Object build(String ns, String sysId);	名前空間およびシステム ID から XMLSchema オブジェクトを作成します。
public Object build(String ns, URL sysId);	名前空間および URL から XMLSchema オブジェクトを作成します。
public Object build(URL schemaurl);	URL から XMLSchema オブジェクトを作成します。

構文	説明
public Object build(XMLDocument schemaDoc);	XML 文書から XMLSchema を作成します。
public Object build(XMLDocument[] schemaDocs, URL baseurl);	XML 文書の配列および URL から XMLSchema を作成します。
public Object build(XMLDocument doc, String fragment, String ns, URL sysId);	XML 文書、フラグメント、名前空間およびシステム ID から XMLSchema オブジェクトを作成します。
public Object build(XMLDocument schemaDoc, URL baseurl);	XML 文書および URL から XMLSchema を作成します。

パラメータ	説明
baseurl	相対参照の解決に使用される URL。文書内の import/include に使用されます。
doc	スキーマ要素を含む XML 文書。
fragment	スキーマ要素のフラグメント ID。
in	スキーマの入力ストリーム。
ns	targetNamespace の検証に使用されるスキーマのターゲットの名前空間。
r	スキーマのリーダー。
schemaDoc	XML 文書。
schemaDocs	XML 文書の配列。
sysId	スキーマの場所。
url	スキーマの URL。

getObject()

説明
XMLSchema オブジェクトを戻します。

構文
```
public Object getObject();
```

setEntityResolver()

説明
import/include を解決するためのエンティティ・リゾルバを設定します。
org.xml.sax.EintityResolver も参照してください。

構文
```
public void setEntityResolver( org.xml.sax.entityResolver entResolver);
```

パラメータ	説明
entResolver	エンティティ・リゾルバ。

setError()

説明
XMLError オブジェクトを設定します。

構文
public void setError(XMLError er)

パラメータ	説明
er	XMLError オブジェクト。

setLocale()

説明

エラー・レポートのロケールを設定します。

構文

```
public void setLocale(L ocale locale);
```

パラメータ	説明
locale	Locale オブジェクト。

XSDComplexType クラス

XSDComplexType の説明

このクラスは、スキーマの complexType の定義を表します。

XSDComplexType の構文

```
public class XSDComplexType extends oracle.xml.parser.schema.XSDNode

oracle.xml.parser.schema.XSDNode
  |
  +--oracle.xml.parser.schema.XSDComplexType
```

XSDComplexType のメソッド

表 6-5 XSDComplexType のメソッドの概要

メソッド	説明
getAttributeDeclarations()（6-20 ページ）	complexType の属性宣言を戻します。
getAttributeWildcard()（6-20 ページ）	complexType の属性ワイルド・カードを戻します。
getBaseType()（6-20 ページ）	complexType のベース型を戻します。
getContent()（6-20 ページ）	XSDComplexType のコンテンツを戻します。
getDerivationMethod()（6-21 ページ）	親の型から XSDComplexType を派生した方法に関する情報を戻します。
getElementSet()（6-21 ページ）	complexType 要素内に存在するすべてのローカル要素の配列を戻します。
getGroup()（6-21 ページ）	属性グループまたは子および属性のグループを戻します。
getRefLocalname()（6-21 ページ）	解決済「base」属性のローカル名を戻します。
getTypeGroup()（6-22 ページ）	complexType の型グループを戻します。
init()（6-22 ページ）	complexType を初期化します。
isAbstract()（6-22 ページ）	complexType が抽象型である場合、`true` を戻します。

getAttributeDeclarations()

説明
complexType の属性宣言を戻します。属性宣言のワイルド・カード配列は含みません。

構文
```
public XSDAttribute getAttributeDeclarations();
```

getAttributeWildcard()

説明
complexType の属性ワイルド・カードを戻します。

構文
```
public oracle.xml.parser.schema.XSDAny getAttributeWildcard();
```

戻り値
属性ワイルド・カード（存在する場合）。

getBaseType()

説明
complexType のベース型を戻します。

構文
```
public XSDNode getBaseType();
```

getContent()

説明
XSDComplexType のコンテンツを戻します。

構文
```
public int getContent();
```

getDerivationMethod()

説明

親の型から XSDComplexType を派生した方法に関する情報を戻します。
EXTENSION_DERIVATION または RESTRICTION_DERIVATION のいずれかになります。

構文

```
public short getDerivationMethod();
```

getElementSet()

説明

complexType 要素内に存在するすべてのローカル要素の配列を戻します。

構文

```
public XSDNode getElementSet();
```

getGroup()

説明

属性グループまたは子および属性のグループを戻します。

構文

```
public XSDGroup getGroup();
```

getRefLocalname()

説明

解決済「base」属性のローカル名を戻します。

構文

```
public java.lang.String getRefLocalname();
```

getTypeGroup()

説明

complexType の型グループを戻します。

構文

```
public XSDGroup getTypeGroup();
```

init()

説明

complexType を初期化します。

構文

```
public static void init();
```

isAbstract()

説明

complexType が抽象型の場合は、`true` を戻します。

構文

```
public boolean isAbstract();
```

XSDConstrainingFacet クラス

XSDConstrainingFacet の説明

このクラスは、データ型に対するスキーマの制約ファセットを表します。

XSDConstrainingFacet の構文

```
public class XSDConstrainingFacet extends java.lang.Object implements
oracle.xml.parser.schema.XSDTypeConstants

java.lang.Object
  |
  +--oracle.xml.parser.schema.XSDConstrainingFacet
```

XSDConstrainingFacet の実装済インタフェース

XSDTypeConstants

XSDConstrainingFacet のメソッド

表 6-6 XSDConstrainingFacet のメソッドの概要

メソッド	説明
getFacetId()（6-24 ページ）	ファセット ID を戻します。
getLexicalEnumeration()（6-24 ページ）	制約ファセットの字句列挙を戻します。
getLexicalValue()（6-24 ページ）	制約ファセットの字句値を戻します。
getName()（6-24 ページ）	制約ファセット名を戻します。
isFixed()（6-25 ページ）	ファセットが固定され、変更できないかどうかを確認します。
setFixed()（6-25 ページ）	ファセットが固定され、変更できないかどうかを設定します。
validateFacet()（6-25 ページ）	制約ファセットに対する値を検証します。

getFacetId()

説明
ファセット ID を戻します。

構文
```
public int getFacetId();
```

getLexicalEnumeration()

説明
制約ファセットの字句列挙を戻します。

構文
```
public java.util.Vector getLexicalEnumeration();
```

getLexicalValue()

説明
制約ファセットの字句値を戻します。

構文
```
public String getLexicalValue();
```

getName()

説明
制約ファセット名を戻します。

構文
```
public String getName();
```

isFixed()

説明

ファセットが固定され、変更できないかどうかを確認します。固定されている場合は true、固定されていない場合は false を戻します。

構文

```
public boolean isFixed();
```

setFixed()

説明

ファセットが固定され、変更できないかどうかを設定します。固定されている場合は true、固定されていない場合は false を戻します。

構文

```
public void setFixed( boolean fixed);
```

パラメータ	説明
value	検証する値。

validateFacet()

説明

制約ファセットに対する値を検証します。

構文

```
public void validateFacet( XSDDataValue value);
```

パラメータ	説明
value	検証する値。

XSDDataValue クラス

XSDDataValue の説明

このクラスは、スキーマの simpleType に対するデータ値を表します。

XSDDataValue の構文

```
public class XSDDataValue extends java.lang.Object implements
oracle.xml.parser.schema.XSDTypeConstants

java.lang.Object
  |
  +--oracle.xml.parser.schema.XSDDataValue
```

XSDDataValue の実装済インタフェース

XSDTypeConstants

XSDDataValue のメソッド

表 6-7 XSDDataValue のメソッドの概要

メソッド	説明
compareTo()（6-26 ページ）	2 つの値を比較して、小さい場合は -1、等しい場合は 0（ゼロ）、大きい場合は 1 を戻します。
getLength()（6-27 ページ）	文字列値またはバイナリ値の長さを戻します。
getLexicalValue()（6-27 ページ）	XSDDataValue クラスの字句値を戻します。
getPrecision()（6-27 ページ）	小数値の精度を戻します。
getScale()（6-27 ページ）	小数値のスケールを戻します。

compareTo()

説明

2 つの値を比較して、小さい場合は -1、等しい場合は 0（ゼロ）、大きい場合は 1 を戻します。データ値が比較不可能な場合は、XSDException が発生します。

構文

```
public int compareTo( XSDDataValue val);
```

getLength()

説明

文字列値またはバイナリ値の長さを取得します。データ値が文字列型またはバイナリ型でない場合は、XSDException が発生します。

構文

```
public int getLength();
```

getLexicalValue()

説明

XSDDataValue クラスの字句値を戻します。

構文

```
public String getLexicalValue();
```

getPrecision()

説明

小数値の精度を戻します。データ値が小数型でない場合は、XSDException が発生します。

構文

```
public int getPrecision();
```

getScale()

説明

小数スケールの int 値を戻します。データ値が小数型でない場合は、XSDException が発生します。

構文

```
public int getScale();
```

XSDElement クラス

XSDElement の説明

このクラスは、スキーマの要素宣言を表します。

XSDElement の構文

```
public class XSDElement
```

oracle.xml.parser.schema.XSDElement

XSDElement のメソッド

表 6-8 XSDElement のメソッドの概要

メソッド	説明
findEquivClass()（6-29 ページ）	XSDElement クラスと同等のクラスを検索します。
getDefaultVal()（6-29 ページ）	要素の場合は、「default」属性の値、および「use」属性に基づく「value」属性の値を戻します。
getEquivClassRef()（6-30 ページ）	解決済同等クラスのローカル名を戻します。
getFixedVal()（6-30 ページ）	要素の場合は、「fixed」属性の値、および「use」属性に基づく「value」属性の値を戻します。
getIdentities()（6-30 ページ）	一連の ID を戻します。
getMaxOccurs()（6-30 ページ）	maxOccurs を戻します。
getMinOccurs()（6-31 ページ）	minOccurs を戻します。
getName()（6-31 ページ）	名前を戻します。
getRefLocalname()（6-31 ページ）	解決済「ref」属性のローカル名を戻します。
getRefNamespace()（6-31 ページ）	解決済「ref」属性の名前空間を戻します。
getRefState()（6-32 ページ）	refState を戻します。
getSubstitutionGroup()（6-32 ページ）	置換グループを戻します。
getTargetNS()（6-32 ページ）	ターゲットの名前空間を設定します。
getType()（6-32 ページ）	ノードのタイプを設定します。
isAbstract()（6-33 ページ）	要素が抽象型の場合は、`true` を戻します。

表 6-8 XSDElement のメソッドの概要（続き）

メソッド	説明
isNullable()（6-33 ページ）	要素が null 値になる可能性がある場合は、true を戻します。
setMaxOccurs()（6-33 ページ）	maxOccurs を設定します。
setMinOccurs()（6-34 ページ）	minOccurs を設定します。

findEquivClass()

説明

XSDElement クラスと同等のクラスを検索します。

構文

```
public XSDElement findEquivClass( String ns,
                                  String nm);
```

パラメータ	説明
ns	名前空間。
nm	名前。

getDefaultVal()

説明

要素の場合は、「default」属性の値、および「use」属性に基づく「value」属性の値を戻します。

構文

```
public String getDefaultVal();
```

getEquivClassRef()

説明

解決済同等クラスのローカル名を戻します。

構文

```
public String getEquivClassRef();
```

getFixedVal()

説明

要素の場合は、「fixed」属性の値、および「use」属性に基づく「value」属性の値を戻します。

構文

```
public java.lang.String getFixedVal();
```

getIdentities()

説明

一連の ID を配列として戻します。

構文

```
public XSDIdentity getIdentities();
```

getMaxOccurs()

説明

maxOccurs を戻します。

構文

```
public int getMaxOccurs();
```

getMinOccurs()

説明
minOccurs を戻します。

構文
```
public int getMinOccurs();
```

getName()

説明
ノード名を戻します。

構文
```
public String getName();
```

getRefLocalname()

説明
解決済「ref」属性のローカル名を戻します。

構文
```
public String getRefLocalname();
```

getRefNamespace()

説明
解決済「ref」属性の名前空間を戻します。

構文
```
public String getRefNamespace();
```

getRefState()

説明

refState を戻します。戻り値は、TYPE_UNRESOLVED、TYPE_RESOLVED、REF_UNRESOLVED または REF_RESOLVED のいずれかになります。

構文

```
public int getRefState();
```

getSubstitutionGroup()

説明

置換グループを戻します。

構文

```
public java.util.Vector getSubstitutionGroup();
```

getTargetNS()

説明

ターゲットの名前空間を戻します。

構文

```
public java.lang.String getTargetNS();
```

getType()

説明

simpleType または complexType のいずれかのノードのタイプを戻します。

構文

```
public XSDNode getType();
```

isAbstract()

説明

要素が抽象型の場合は、true を戻します。

構文

```
public boolean isAbstract();
```

isNullable()

説明

要素が null 値になる可能性がある場合は、true を戻します。

構文

```
public boolean isNullable();
```

setMaxOccurs()

説明

maxOccurs を設定します。

構文

```
public void setMaxOccurs( int max);
```

パラメータ	説明
max	値。

setMinOccurs()

説明

minOccurs を設定します。

構文

```
public void setMinOccurs( int min);
```

パラメータ	説明
min	値。

XSDException クラス

XSDException の説明

このクラスは、XMLSchema 検証中に例外が発生したことを示します。

XSDException の構文

```
java.lang.Object
   |
   +---java.lang.Throwable
           |
           +---java.lang.Exception
                   |
                   +---oracle.xml.parser.schema.XSDException

public class XSDException extends Exception
```

getMessage()

説明

Throwable クラスの getMessage() をオーバーライドして、エラー ID およびエラー・パラメータからエラー・メッセージを作成します。次の表に、オプションを示します。

構文	説明
public String getMessage()	エラー ID およびエラー・パラメータからエラー・メッセージを作成します。
public String getMessage(XMLError err)	パラメータとして送信される XMLError に基づいて、ローカライズされたエラー・メッセージを作成します。

パラメータ	説明
err	エラー・メッセージの取得に使用される XMLError クラス。

XSDGroup クラス

XSDGroup の説明

このクラスは、スキーマ・グループの定義を表します。

XSDGroup の構文

```
public class XSDGroup
```

oracle.xml.parser.schema.XSDGroup

XSDGroup のメソッド

表 6-9 XSDIdentity のメソッドの概要

メソッド	説明
getMaxOccurs()（6-36 ページ）	maxOccurs を戻します。
getMinOccurs()（6-37 ページ）	minOccurs を戻します。
getNodeVector()（6-37 ページ）	nodeVector に格納されたグループの小要素を戻します。
getOrder()（6-37 ページ）	ALL、SEQUENCE または CHOICE のいずれかのコンポジット型を戻します。
setMaxOccurs()（6-37 ページ）	maxOccurs を設定します。
setMinOccurs()（6-38 ページ）	minOccurs を設定します。

getMaxOccurs()

説明

maxOccurs を戻します。

構文

```
public int getMaxOccurs();
```

getMinOccurs()

説明
minOccurs を戻します。

構文
```
public int getMinOccurs();
```

getNodeVector()

説明
nodeVector に格納されたグループの小要素を戻します。

構文
```
public java.util.Vector getNodeVector();
```

getOrder()

説明
ALL、SEQUENCE または CHOICE のいずれかのコンポジット型を戻します。

構文
```
public int getOrder();
```

setMaxOccurs()

説明
maxOccurs を設定します。

構文
```
public void setMaxOccurs( int max);
```

パラメータ	説明
max	値。

setMinOccurs()

説明

minOccurs を設定します。

構文

```
public void setMinOccurs( int min);
```

パラメータ	説明
min	値。

XSDIdentity クラス

XSDIdentity の説明

このクラスは、スキーマの ID 制約定義 Unique、Key および Keyref を表します。

XSDIdentity の構文

```
public class XSDIdentity extends oracle.xml.parser.schema.XSDNode

oracle.xml.parser.schema.XSDNode
  |
  +--oracle.xml.parser.schema.XSDIdentity
```

XSDIdentity のメソッド

表 6-10 XSDIdentity のメソッドの概要

メソッド	説明
getFields()（6-39 ページ）	フィールドを戻します。
getNodeType()（6-40 ページ）	ノードのタイプを戻します。
getRefer()（6-40 ページ）	参照キーを戻します。
getSelector()（6-40 ページ）	セレクタを戻します。

getFields()

説明

フィールドを戻します。

構文

```
public java.lang.String[] getFields();
```

getNodeType()

説明

ノードのタイプを戻します。XSDNode クラスの XSDNode.getNodeType() をオーバーライドします。

構文

```
public int getNodeType();
```

getRefer()

説明

参照キーを戻します。

構文

```
public String getRefer();
```

getSelector()

説明

セレクタを戻します。

構文

```
public String getSelector();
```

XSDNode クラス

XSDNode の説明

このクラスは、ほとんどの XSD クラスのルート・クラスです。このクラスには、XMLSchema の定義の属性に対応するフィールドおよびメソッドが含まれます。

XSDNode の構文

```
public class XSDNode

oracle.xml.parser.schema.XSDNode
```

XSDNode のダイレクト・サブクラス

XMLSchema、XMLSchemaNode、XSDAttribute、XSDComplexType、XSDIdentity

XSDNode のメソッド

表 6-11 XSDNode のメソッドの概要

メソッド	説明
getName()（6-41 ページ）	ノード名を戻します。
getNamespaceURI()（6-42 ページ）	名前空間の URI を戻します。
getNodeType()（6-42 ページ）	XSDNode のタイプを戻します。
getTargetNS()（6-42 ページ）	ターゲットの名前空間を戻します。
isNodeType()（6-42 ページ）	ノードが指定されたタイプかどうかを確認します。

getName()

説明

ノード名を戻します。

構文

```
public java.lang.String getName();
```

getNamespaceURI()

説明
名前空間の URI を戻します。

構文
```
public String getNamespaceURI()
```

getNodeType()

説明
XSDNode のタイプを戻します。

構文
```
public int getNodeType();
```

getTargetNS()

説明
ターゲットの名前空間を戻します。

構文
```
public java.lang.String getTargetNS();
```

isNodeType()

説明
ノードが指定されたタイプかどうかを確認します。

構文
```
public boolean isNodeType( int type);
```

パラメータ	説明
type	確認するノードのタイプ。

XSDSimpleType クラス

XSDSimpleType の説明

このクラスは、スキーマの simpleType の定義を表します。

XSDSimpleType の構文

```
public class XSDSimpleType implements oracle.xml.parser.schema.XSDTypeConstants
```

oracle.xml.parser.schema.XSDSimpleType

XSDSimpleType の実装済インタフェース

XSDTypeConstants

XSDSimpleType のメソッド

表 6-12 XSDSimpleType のメソッドの概要

メソッド	説明
XSDSimpleType()（6-44 ページ）	クラス・コンストラクタです。
derivedFrom()（6-44 ページ）	指定されたベース型から型を派生させます。
getBase()（6-45 ページ）	XSDSimpleType のベース型を戻します。
getBasicType()（6-45 ページ）	XSDSimpleType を派生したベース型を取得します。
getBuiltInDatatypes()（6-45 ページ）	組込みデータ型を取得します。
getFacets()（6-46 ページ）	ファセットを取得します。
getMaxOccurs()（6-46 ページ）	maxOccurs の値を取得します。
getMinOccurs()（6-46 ページ）	minOccurs の値を取得します。
getVariety()（6-46 ページ）	型の種類を取得します。
isAbstract()（6-47 ページ）	XSDSimpleType が抽象型の場合は、`true` を戻します。
setFacet()（6-47 ページ）	データ型にファセットを設定します（内部プライベート API）。
setMaxOccurs()（6-47 ページ）	maxOccurs の値を設定します。

表 6-12 XSDSimpleType のメソッドの概要（続き）

メソッド	説明
setMinOccurs()（6-48 ページ）	minOccurs の値を設定します。
setSource()（6-48 ページ）	データ型のベース型を設定します。集計型の場合は、集計型のコンポーネントの型を設定します。
validateValue()（6-48 ページ）	XSDSimpleType に定義されたファセットを使用して文字列値を検証します。

XSDSimpleType()

説明

クラス・コンストラクタです。次の表に、オプションを示します。

構文	説明
public XSDSimpleType();	デフォルトのコンストラクタです。
public XSDSimpleType(int basic, String tnm);	W3C の XMLSchema データ型定義を実装します。

パラメータ	説明
basic	XSDSimpleType が派生した基本型の ID。
tnm	XSDSimpleType の名前。

derivedFrom()

説明

指定されたベース型から型を派生させます。新しい型を作成できない場合は、XSDException が発生します。

構文

```
public static XSDSimpleType derivedFrom( XSDSimpleType source,
                                         String nm,
                                         String var);
```

パラメータ	説明
source	XSDSimpleType（ベース型）。
nm	新しい型の名前。
var	派生のメソッド。

getBase()

説明
ベース型を取得します。

構文
```
public XSDSimpleType getBase();
```

getBasicType()

説明
XSDSimpleType を派生したベース型を取得します。

構文
```
public int getBasicType();
```

getBuiltInDatatypes()

説明
組込みデータ型を取得します。型の名前が有効でない場合は、XSDException が発生します。

構文
```
public static Hashtable getBuiltInDatatypes();
```

getFacets()

説明
ファセットを取得します。

構文
```
public XSDConstrainingFacet getFacets();
```

getMaxOccurs()

説明
maxOccurs の値を取得します。

構文
```
public int getMaxOccurs();
```

getMinOccurs()

説明
minOccurs の値を取得します。

構文
```
public int getMinOccurs();
```

getVariety()

説明
型の種類を取得します。

構文
```
public java.lang.String getVariety();
```

isAbstract()

説明

型が抽象型の場合は true、抽象型でない場合は false を戻します。

構文

```
public boolean isAbstract();
```

setFacet()

説明

データ型にファセットを設定します（内部プライベート API）。ファセットが有効でない場合は、XSDException が発生します。

構文

```
public void setFacet( String facetName,
                      String value);
```

パラメータ	説明
facetName	設定するファセットの名前。
value	ファセットの値。

setMaxOccurs()

説明

maxOccurs の値を設定します。

構文

```
public void setMaxOccurs(int max);
```

パラメータ	説明
max	最大出現回数。

setMinOccurs()

説明

minOccurs の値を設定します。

構文

```
public void setMinOccurs( int min);
```

パラメータ	説明
min	最小出現回数。

setSource()

説明

データ型のベース型を設定します。集計型の場合は、集計型のコンポーネントの型を設定します。src が有効な型でない場合は、SDException が発生します。

構文

```
public void setSource( XSDNode src);
```

パラメータ	説明
src	XSDNode のソース。

validateValue()

説明

XSDSimpleType に定義されたファセットを使用して文字列値を検証します。値が有効でない場合は、XSDException が発生します。

構文

```
public void validateValue( String val);
```

パラメータ	説明
val	検証する値。

XSDConstantValues インタフェース

XSDConstantValues の説明

このインタフェースは、W3C のスキーマ・プロセッサの定数を定義します。

XSDConstantValues の構文

```
public interface XSDConstantValues
```

XSDConstantValues で定義された定数

表 6-13 XSDConstantValues で定義された定数

定数	定義
_abstract	public static final String _abstract
_all	public static final String _all
_annotation	public static final String _annotation
_any	public static final String _any
_anyAttribute	public static final String _anyAttribute
_anySimpleType	public static final String _anySimpleType
_anyType	public static final String _anyType
_attrFormDefault	public static final String _attrFormDefault
_attribute	public static final String _attribute
_attributeGroup	public static final String _attributeGroup
_attrTag	public static final String _attrTag
_base	public static final String _base
_block	public static final String _block
_blockDefault	public static final String _blockDefault
_choice	public static final String _choice
_complexContent	public static final String _complexContent
_complexType	public static final String _complexType
_content	public static final String _content
_default	public static final String _default

表 6-13 XSDConstantValues で定義された定数（続き）

定数	定義
_derivedBy	public static final String _derivedBy
_element	public static final String _element
_elementOnly	public static final String _elementOnly
_elemFormDefault	public static final String _elemFormDefault
_empty	public static final String _empty
_enumeration	public static final String _enumeration
_equivClass	public static final String _equivClass
_extension	public static final String _extension
_false	public static final String _false
_field	public static final String _field
_final	public static final String _final
_finalDefault	public static final String _finalDefault
_fixed	public static final String _fixed
_form	public static final String _form
_group	public static final String _group
_id	public static final String _id
_import	public static final String _import
_include	public static final String _include
_itemType	public static final String _itemType
_key	public static final String _key
_keyref	public static final String _keyref
_lax	public static final String _lax
_list	public static final String _list
_maxOccurs	public static final String _maxOccurs
_memberTypes	public static final String _memberTypes
_minOccurs	public static final String _minOccurs
_mixed	public static final String _mixed
_nall	public static final String _nall
_name	public static final String _name

表 6-13 XSDConstantValues で定義された定数（続き）

定数	定義
_namespace	public static final String _namespace
_nil	public static final String _nil
_nillable	public static final String _nillable
_nnany	public static final String _nnany
_nnlist	public static final String _nnlist
_nnlocal	public static final String _nnlocal
_nnother	public static final String _nnother
_nntargetNS	public static final String _nntargetNS
_noNSSchemaLocation	public static final String _noNSSchemaLocation
_notation	public static final String _notation
_null	public static final String _null
_nullable	public static final String _nullable
_optional	public static final String _optional
_pattern	public static final String _pattern
_processContents	public static final String _processContents
_prohibited	public static final String _prohibited
_publicid	public static final String _publicid
_qualified	public static final String _qualified
_redefine	public static final String _redefine
_ref	public static final String _ref
_refer	public static final String _refer
_required	public static final String _required
_restriction	public static final String _restriction
_restrictions	public static final String _restrictions
_schema	public static final String _schema
_schemaLocation	public static final String _schemaLocation
_selector	public static final String _selector
_sequence	public static final String _sequence

表 6-13 XSDConstantValues で定義された定数（続き）

定数	定義
_simpleContent	public static final String _simpleContent
_simpleType	public static final String _simpleType
_skip	public static final String _skip
_strict	public static final String _strict
_substitution	public static final String _substitution
_systemid	public static final String _systemid
_targetNS	public static final String _targetNS
_textOnly	public static final String _textOnly
_this	public static final String _this
_true	public static final String _true
_type	public static final String _type
_undef	public static final String _undef
_union	public static final String _union
_unique	public static final String _unique
_unqualified	public static final String _unqualified
_use	public static final String _use
_value	public static final String _value
_version	public static final String _version
_xmlns	public static final String _xmlns
ABSENT_NS	public static final int ABSENT_NS
ACCEPTED	public static final int ACCEPTED
ALL	public static final int ALL
ANY	public static final int ANY
ANY_ATTRIBUTE	public static final int ANY_ATTRIBUTE
ANY_NODE	public static final String ANY_NODE
ATTRIBUTE	public static final int ATTRIBUTE
ATTRIBUTE_GROUP	public static final int ATTRIBUTE_GROUP
AUTO_VALIDATION	public static final String AUTO_VALIDATION
BASE_RESOLVED	public static final int BASE_RESOLVED

表 6-13 XSDConstantValues で定義された定数（続き）

定数	定義
BASE_UNRESOLVED	public static final int BASE_UNRESOLVED
BASE_URL	public static final String BASE_URL
CHOICE	public static final int CHOICE
constName	public static final String constName[]
cyclicChain	public static final int cyclicChain
DATATYPE	public static final int DATATYPE
DONE	public static final int DONE
ELEMENT	public static final int ELEMENT
ELEMENT_CHILD	public static final int ELEMENT_CHILD
ELEMENT_ONLY	public static final int ELEMENT_ONLY
elemNotNullable	public static final int elemNotNullable
EMPTY	public static final int EMPTY
EQUIV_RESOLVED	public static final int EQUIV_RESOLVED
EQUIV_UNRESOLVED	public static final int EQUIV_UNRESOLVED
ERROR	public static final int ERROR
EXTENTION	public static final int EXTENTION
FACET_CHILD	public static final int FACET_CHILD
FAKE_NODE	public static final XSDGroup FAKE_NODE
FIXED_SCHEMA	public static final String FIXED_SCHEMA
GROUP	public static final int GROUP
IDENTITY_KEY	public static final int IDENTITY_KEY
IDENTITY_KEYREF	public static final int IDENTITY_KEYREF
IDENTITY_UNIQUE	public static final int IDENTITY_UNIQUE
IMPORT	public static final int IMPORT
INCLUDE	public static final int INCLUDE
INFINITY	public static final int INFINITY
invalidAttr	public static final int invalidAttr
invalidAttrVal	public static final int invalidAttrVal

表 6-13 XSDConstantValues で定義された定数（続き）

定数	定義
invalidChars	public static final int invalidChars
invalidContent	public static final int invalidContent
invalidDerivation	public static final int invalidDerivation
invalidElem	public static final int invalidElem
invalidETag	public static final int invalidETag
invalidFacet	public static final int invalidFacet
invalidFixedChars	public static final int invalidFixedChars
invalidNameRef	public static final int invalidNameRef
invalidNS	public static final int invalidNS
invalidParsAttr	public static final int invalidParsAttr
invalidPrefix	public static final int invalidPrefix
invalidRef	public static final int invalidRef
invalidSTag	public static final int invalidSTag
invalidTargetNS	public static final int invalidTargetNS
LAX_VALIDATION	public static final String LAX_VALIDATION
missingAttr	public static final int missingAttr
MIXED	public static final int MIXED
needsSource	public static final int needsSource
NEW_STATE	public static final int NEW_STATE
NO_CHILD	public static final int NO_CHILD
noDefinition	public static final int noDefinition
NOT_DONE	public static final int NOT_DONE
NOTATION	public static final int NOTATION
notComplete	public static final int notComplete
notSubTypeOf	public static final int notSubTypeOf
NS_FRAME	public static final int NS_FRAME
NS_RESOLVER	public static final String NS_RESOLVER
REDEFINE	public static final int REDEFINE
REF_RESOLVED	public static final int REF_RESOLVED

表 6-13 XSDConstantValues で定義された定数（続き）

定数	定義
REF_UNRESOLVED	public static final int REF_UNRESOLVED
refToAbstractElem	public static final int refToAbstractElem
refToAbstractType	public static final int refToAbstractType
RESTRICTION	public static final int RESTRICTION
SCHEMA_NS	public static final int SCHEMA_NS
SEQ	public static final int SEQ
STRICT_VALIDATION	public static final String STRICT_VALIDATION
TEXT_ONLY	public static final int TEXT_ONLY
TOP_LEVEL	public static final int TOP_LEVEL
TYPE	public static final int TYPE
TYPE_RESOLVED	public static final int TYPE_RESOLVED
TYPE_UNRESOLVED	public static final int TYPE_UNRESOLVED
UNDEF	public static final int UNDEF
undefinedType	public static final int undefinedType
unexpectedAttr	public static final int unexpectedAttr
unexpectedElem	public static final int unexpectedElem
unnamedAttrDecl	public static final int unnamedAttrDecl
VALIDATION_MODE	public static final String VALIDATION_MODE
XSDAUG2000NS	public static final String XSDAUG2000NS
XSDDATATYPENS	public static final String XSDDATATYPENS
XSDNAMESPACE	public static final String XSDNAMESPACE
XSDRECNS	public static final String XSDRECNS
XSDRECTYPENS	public static final String XSDRECTYPENS
XSI2000NS	public static final String XSI2000NS
XSINAMESPACE	public static final String XSINAMESPACE
XSIRECNS	public static final String XSIRECNS

XSDValidator クラス

XSDValidator の説明

このクラスは、XMLSchema に対して XML 文書を検証します。

XSDValidator を登録すると、XMLParser と XMLDocument のイベント・ハンドラ（SAXHandler または DOMBuilder）間にパイプライン・ノードとして、XSDValidator オブジェクトが挿入されます。XSDValidator オブジェクトは、`startElement()`、`characters()` および `endElement()` の 3 つのイベントで動作します。デフォルトの要素およびデフォルトの属性値が定義されていると、これらは情報セットへの追加の XMLSchema としてイベントのコンテンツに追加され、上方に伝播されます。

XMLSchema オブジェクトは、要素宣言のセットまたはグループです。

```
[element(name)] -> [shode(min/maxOccurs)] ->
    [type(group/simpleType)]
```

XSDValidator は、スタック・ベースの状態マシンとして実装されます。各状態は、要素型（group または simpleType）を表します。

XMLSchema オブジェクト（グループ）が、最初の状態としてロードされます。現行の element が、現行の状態グループの要素と一致するものがあるかどうかを確認します。要素型が一致する場合は、要素名および snode 情報が新しい状態としてロードされます。

グループの場合は、counters のベクトルが並列スタックに割り当てられます。このベクトルは、要素の出現回数のカウントに使用されます。

状態ステータスは次のいずれかになります。

- NEW_STATE: ロードされるのみで、試行されません。
- ACCEPTED: minOccurs を満たします。要素の出現回数も受け入れます。
- DONE: maxOccurs を満たします。要素の出現回数は受け入れません。

要素のテキスト内容またはイベント文字は、`validateValue()` メソッドを介して simpleType と一致するかどうかを確認します。`endElement()` イベントを介して検出された要素の終了は、最後に指定された状態と一致します。

XMLSchema 属性は、特殊な要素（<_attrTag> attrType）の内容を構成するグループ（attrName -> attrType）として表されます。それに応じて、XMLParser によって、`startElement()` イベントを介して検出された属性が変換されます。

XSDAny オブジェクトは、名前空間フレーム記述子として使用されます。

エラーが発生した場合またはワイルド・カード（「any」）の内容がスキップされた場合は、不正な状態がロードされます。

XSDValidator の構文

```
public class XSDValidator
```

oracle.xml.parser.schema.XSDValidator

XSDValidator のメソッド

表 6-14 XSDValidator のメソッドの概要

メソッド	説明
XSDValidator()（6-57 ページ）	クラス・コンストラクタです。
characters()（6-58 ページ）	要素内の文字データの通知を伝播します。
endElement()（6-58 ページ）	要素の終わりの通知を受け取ります。
setDocumentLocator()（6-59 ページ）	ドキュメント・イベントに対する Locator オブジェクトを伝播します。
setError()（6-59 ページ）	XMLError オブジェクトを現在のエラーに設定します。
setXMLProperties()（6-60 ページ）	実行時プロパティに XML プロパティを設定します。
setXMLProperty()（6-60 ページ）	プロパティを設定します。
startElement()（6-61 ページ）	要素の始まりの通知を受け取ります。

XSDValidator()

説明

XSDValidator のコンストラクタです。

構文

```
public  XSDValidator();
```

characters()

説明

要素内の文字データの通知を伝播します。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。org.xml.sax.DocumentHandler も参照してください。

構文

```
public void characters( char[] ch,
                        int start,
                        int length);
```

パラメータ	説明
ch	文字。
start	文字配列の開始位置。
length	文字配列のうちの使用する文字数。

endElement()

説明

要素の終わりの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void endElement( String namespaceURI,
                        String localName,
                        String qName);
```

パラメータ	説明
namespaceURI	名前空間の URI（要素に名前空間 URI が設定されていない場合または名前空間処理が実行されていない場合は、空の文字列）。
localName	接頭辞なしのローカル名（名前空間処理が実行されていない場合は空の文字列）。
qName	XML 1.0 の接頭辞付き修飾名（修飾名が使用不可能な場合は空の文字列）。

setDocumentLocator()

説明

ドキュメント・イベント用の Locator オブジェクトを伝播します。
org.xml.sax.DocumentHandler および org.xml.sax.Locator も参照してください。

構文

```
public void setDocumentLocator( org.xml.sax.Locator locator);
```

パラメータ	説明
locator	すべての SAX ドキュメント・イベント用のロケータ。

setError()

説明

XMLError オブジェクトを現在のエラーに設定します。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。

構文

```
public void setError( oracle.xml.parser.v2.XMLError he);
```

パラメータ	説明
he	XMLError オブジェクト。

setXMLProperties()

説明

実行時プロパティに XML プロパティを設定します。

構文

```
public void setXMLProperties( XMLProperties xmlProp);
```

パラメータ	説明
xmlProp	XML プロパティ。

setXMLProperty()

説明

プロパティを設定して戻します。設定が正常に終了した場合は、設定されたプロパティの値が戻されます。プロパティが読込み専用で設定できない場合またはサポートされない場合は、null が戻されます。

構文

```
public Object setXMLProperty( java.lang.String name,
                              java.lang.Object value);
```

パラメータ	説明
name	プロパティ名。
value	プロパティの値。

startElement()

説明

要素の始まりの通知を受け取ります。org.xml.sax.SAXException が発生します。この例外は、SAXException またはそのサブクラスのインスタンスで、別の例外が隠されている可能性があります。endElement() および org.xml.sax.Attributes も参照してください。

構文

```
public void startElement( String namespaceURI,
                          String localName,
                          String qName,
                          Attributes atts);
```

パラメータ	説明
namespaceURI	名前空間の URI（要素に名前空間 URI が設定されていない場合または名前空間処理が実行されていない場合は、空の文字列）。
localName	接頭辞なしのローカル名（名前空間処理が実行されていない場合は空の文字列）。
qName	接頭辞付き修飾名（修飾名が使用不可能な場合は空の文字列）。
atts	要素に追加された属性（属性が存在しない場合は空の Attributes オブジェクト）。

7

XML Class Generator for Java

XML Class Generator for Java は、oracle.xml.classgen パッケージに含まれています。

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

CGDocument クラス

CGDocument の説明

このクラスは、Class Generator によって生成された DTD クラスのベース・ドキュメント・クラスとして機能します。

CGDocument の構文

```
public abstract class CGDocument extends oracle.xml.classgen.CGNode implements
java.io.Externalizable

oracle.xml.classgen.CGNode
  |
  +--oracle.xml.classgen.CGDocument
```

CGDocument の実装済インタフェース

```
java.io.Externalizable, java.io.Serializable
```

CGDocument のメソッド

表 7-1 CGDocument のメソッドの概要

メソッド	説明
CGDocument()（7-3 ページ）	DTD のルート要素に対するコンストラクタです。
print()（7-3 ページ）	作成した XML 文書を出力します。
readExternal()（7-4 ページ）	圧縮ストリームを読み込み、ルート要素に対応するオブジェクトを作成します。

CGDocument()

説明

DTD のルート要素に対するコンストラクタです。

構文

```
protected  CGDocument( String doctype,
                       oracle.xml.parser.v2.DTD dtd);
```

パラメータ	説明
doctype	DTD のルート要素名。
dtd	クラスの生成に使用される DTD。

print()

説明

作成した XML 文書を出力します。ドキュメントのコンテンツが DTD によって指定された文法と一致しない場合は、`InvalidContentException` が発生します（検証モードは、`true` に設定する必要があります）。DTDClassGenerator クラスの `setValidationMode()` も参照してください。次の表に、オプションを示します。

構文	説明
protected void print(java.io.OutputStream out);	作成した XML 文書を出力ストリームに出力します。
protected void print(java.io.OutputStream out, String enc);	作成した XML 文書をユーザー定義のエンコーディングで出力ストリームに出力します。

パラメータ	説明
out	ドキュメントが出力される出力ストリーム。
enc	出力ストリームのエンコーディング。

readExternal()

説明

圧縮ストリームを読み込み、ルート要素に対応するオブジェクトを作成します。XML インスタンス・ドキュメントを使用して生成されたクラスのインスタンス化に使用されます。

構文

```
protected void readExternal( java.io.ObjectInput inArg,
                             oracle.xml.comp.CXMLContext cxmlContext);
```

パラメータ	説明
inArg	圧縮ストリームを読み込むために渡される ObjectInput ストリーム。
cxmlContext	圧縮ストリームのコンテキスト。

CGNode クラス

CGNode の説明

このクラスは、DTD Class Generator によって生成された XML 文書のノードに対応するクラスのベース・クラスとして機能します。

CGNode の構文

```
public abstract class CGNode
```

oracle.xml.classgen.CGNode

CGNode のダイレクト・サブクラス

CGDocument

CGNode のフィールド

表 7-2 CGNode のフィールド

フィールド	構文	説明
isValidating	protected boolean isValidating	検証モードを示すブール値。

CGNode のメソッド

表 7-3 CGNode のメソッドの概要

メソッド	説明
CGNode()（7-6 ページ）	DOM ツリーの要素に対するコンストラクタです。
addCDATASection()（7-7 ページ）	要素に CDATA セクションを追加します。
addData()（7-7 ページ）	要素ノードに PCDATA を追加します。
addNode()（7-7 ページ）	要素にノードを子として追加します。
deleteData()（7-8 ページ）	要素ノードから PCDATA を削除します。
getAttribute()（7-8 ページ）	属性値を取得します。
getCGDocument()（7-9 ページ）	ベース・ドキュメントを取得します。
getData()（7-9 ページ）	要素の PCDATA を取得します。

表 7-3 CGNode のメソッドの概要（続き）

メソッド	説明
getDTDNode()（7-9 ページ）	ベース・ドキュメントから静的 DTD を取得します。
getElementNode()（7-9 ページ）	CGNode に対応する XMLElement ノードを取得します。
getNode()（7-10 ページ）	CGNode を取得します。CGNode は、入力文字列と一致する名前を持つノードに対応する要素の子の 1 つです。
readExternal()（7-10 ページ）	圧縮ストリームを読み込み、対応するノードをインスタンス化します。
setAttribute()（7-11 ページ）	属性値を設定します。
setDocument()（7-11 ページ）	ベース・ドキュメントを設定します。
setElementNode()（7-11 ページ）	CGNode に対応する XMLElement ノードを設定します。
storeID()（7-12 ページ）	ID 識別子の値を格納します。
storeIDREF()（7-12 ページ）	IDREF 識別子の値を格納します。
validateContent()（7-12 ページ）	DTD で指定されたコンテンツ・モデルのとおりに、要素の内容が有効かどうかを確認します。
validEntity()（7-13 ページ）	ENTITY 識別子が有効かどうかを確認します。
validID()（7-13 ページ）	ID 識別子が有効かどうかを確認します。
validNMTOKEN()（7-13 ページ）	NMTOKEN 識別子が有効かどうかを確認します。
writeExternal()（7-14 ページ）	CGNode に対応する圧縮ストリームを書き込みます。

CGNode()

説明

DOM ツリーの要素に対するコンストラクタです。

構文

```
protected  CGNode( String elementName);
```

パラメータ	説明
elementName	要素名。

addCDATASection()

説明

要素に CDATA セクションを追加します。theData に無効な文字が含まれている場合は、InvalidContentException が発生します（検証モードは true に設定する必要があります）。DTDClassGenerator クラスの setValidationMode() も参照してください。

構文

```
protected void addCDATASection( String theData);
```

パラメータ	説明
theData	要素に CDATA セクションとして追加されるテキスト。

addData()

説明

要素ノードに PCDATA を追加します。theData に無効な文字が含まれている場合は、InvalidContentException が発生します（検証モードは true に設定する必要があります）。DTDClassGenerator クラスの setValidationMode() も参照してください。

構文

```
protected void addData( String theData);
```

パラメータ	説明
theData	要素に追加するテキスト。

addNode()

説明

要素にノードを子として追加します。theData に無効な文字が含まれている場合は、InvalidContentException が発生します（検証モードは true に設定する必要があります）。DTDClassGenerator クラスの setValidationMode() も参照してください。

構文

```
protected void addNode( CGNode theNode);
```

パラメータ	説明
theNode	子として追加されるノード。

deleteData()

説明

要素ノードから PCDATA を削除します。theData に無効な文字が含まれている場合、InvalidContentException が発生します（検証モードは true に設定する必要があります）。DTDClassGenerator クラスの setValidationMode() も参照してください。

構文

```
protected void deleteData( String theData);
```

パラメータ	説明
theData	要素から削除するテキスト。

getAttribute()

説明

属性値を戻します。

構文

```
protected String getAttribute( String attName);
```

パラメータ	説明
attName	属性名。

getCGDocument()

説明

ベース・ドキュメント（ルート要素）を取得します。

構文

```
protected CGDocument getCGDocument();
```

getData()

説明

要素の PCDATA を取得します。データが存在しない場合は、InvalidContentException が発生します。

構文

```
protected String getData();
```

getDTDNode()

説明

ベースの CGDocument から静的 DTD を取得します。

構文

```
protected abstract oracle.xml.parser.v2.DTD getDTDNode();
```

getElementNode()

説明

CGNode に対応する XMLElement ノードを取得します。

構文

```
protected oracle.xml.parser.v2.XMLElement getElementNode();
```

getNode()

説明

CGNode を取得します。CGNode は、入力文字列と一致する名前を持つノードに対応する要素の子の 1 つです。

構文

```
protected java.lang.Object getNode(String theNode);
```

パラメータ	説明
theNode	戻された CGNode に対応する文字列の名前。

readExternal()

説明

圧縮ストリームを読み込み、対応するノードをインスタンス化します。次の例外が発生します。

- `IOException`（I/O エラーが発生した場合）
- `ClassNotFoundException`（クラスをインスタンス化できない場合）

構文

```
protected void readExternal( oracle.xml.io.XMLObjectInput in,
                             oracle.xml.comp.CXMLContext cxmlContext)
```

パラメータ	説明
in	圧縮ストリームの読込みに使用される `XMLObjectInput` ストリーム。
cxmlContext	圧縮ストリームのコンテキスト。

setAttribute()

説明
属性値を設定します。

構文
```
protected void setAttribute( String attName,
                             String value);
```

パラメータ	説明
attName	属性名。
value	属性値。

setDocument()

説明
ベース・ドキュメント（ルート要素）を設定します。

構文
```
public void setDocument( CGDocument d);
```

パラメータ	説明
d	ベースの CGDocument。

setElementNode()

説明
CGNode に対応する XMLElement ノードを設定します。

構文
```
protected void setElementNode( oracle.xml.parser.v2.XMLElement node);
```

パラメータ	説明
node	XMLElement。

storeID()

説明

IDREF 値によって検証できるように、ID 識別子の値を格納します。

構文

```
protected void storeID( String attName,
                        String id);
```

パラメータ	説明
attName	ID 属性の名前。
id	ID の値。

storeIDREF()

説明

対応する ID によって検証できるように、IDREF 識別子の値を格納します。

構文

```
protected void storeIDREF( String attName,
                           String idref);
```

パラメータ	説明
attName	IDREF 属性の名前。
idref	IDREF の値。

validateContent()

説明

DTD で指定されたコンテンツ・モデルのとおりに、要素の内容が有効かどうかを確認します。

構文

```
protected void validateContent();
```

validEntity()

説明

ENTITY 識別子が有効かどうかを確認します。ENTITY が有効な場合は `true`、有効でない場合は `false` を戻します。

構文

```
protected boolean validEntity( String entity);
```

パラメータ	説明
entity	ENTITY 属性の値。

validID()

説明

ID 識別子が有効かどうかを確認します。ID が有効な場合は `true`、有効でない場合は `false` を戻します。

構文

```
protected boolean validID( String name);
```

パラメータ	説明
name	ID 属性の値。

validNMTOKEN()

説明

NMTOKEN 識別子が有効かどうかを確認します。NMTOKEN が有効な場合は `true`、有効でない場合は `false` を戻します。

構文

```
protected boolean validNMTOKEN( String name);
```

パラメータ	説明
name	NMTOKEN 属性の値。

writeExternal()

説明

CGNode に対応する圧縮ストリームを書き込みます。

構文

```
protected void writeExternal( oracle.xml.io.XMLObjectOutput out,
                  oracle.xml.comp.CXMLContext cxmlContext);
```

パラメータ	説明
out	圧縮データを書き込む ObjectOutput ストリーム。
cxmlContext	圧縮ストリームのコンテキスト。

CGXSDElement クラス

CGXSDElement の説明

このクラスは、Schema Class Generator によって生成された XML Schema に対応する、すべての生成されたクラスのベース・クラスとして機能します。

CGXSDElement の構文

```
public abstract class CGXSDElement extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.classgen.CGXSDElement
```

CGXSDElement のフィールド

表 7-4 CGXSDElement のフィールド

フィールド	構文	説明
type	protected java.lang.Object type	ノードのタイプの情報。

CGXSDElement のメソッド

表 7-5 CGXSDElement のメソッドの概要

メソッド	説明
CGXSDElement()（7-16 ページ）	デフォルトのコンストラクタです。
addAttribute()（7-16 ページ）	指定されたノードの属性を、ハッシュテーブルに追加します。
addElement()（7-16 ページ）	要素ノードのローカル要素を、要素に対応するベクトルに追加します。
getAttributes()（7-17 ページ）	属性を、属性名および属性値のハッシュテーブルとして戻します。
getChildElements()（7-17 ページ）	すべてのローカル要素のベクトルを取得します。
getNodeValue()（7-17 ページ）	ノードの値を戻します。
print()（7-17 ページ）	要素ノードを出力します。

表 7-5 CGXSDElement のメソッドの概要（続き）

メソッド	説明
printAttributes()（7-18 ページ）	属性ノードを出力します。
setNodeValue()（7-18 ページ）	要素のノード値を設定します。

CGXSDElement()

説明
デフォルトのコンストラクタです。

構文
```
public  CGXSDElement();
```

addAttribute()

説明
指定されたノードの属性を、ハッシュテーブルに追加します。

構文
```
protected void addAttribute( String attName,
                             java.lang.Object attValue);
```

パラメータ	説明
attName	属性名。
attValue	属性値。

addElement()

説明
要素ノードのローカル要素を、要素に対応するベクトルに追加します。

構文
```
protected void addElement( java.lang.Object elem);
```

パラメータ	説明
elem	追加の必要があるオブジェクト。

getAttributes()

説明

属性を、属性名および属性値のハッシュテーブルとして戻します。

構文

```
public java.util.Hashtable getAttributes();
```

getChildElements()

説明

すべてのローカル要素のベクトルを取得します。

構文

```
public java.util.Vector getChildElements();
```

getNodeValue()

説明

ノードの値を戻します。

構文

```
public String getNodeValue();
```

print()

説明

要素ノードを出力します。出力ストリームに出力できない場合は、IOException が発生します。

構文

```
public void print( oracle.xml.parser.v2.XMLOutputStream out);
```

パラメータ	説明
out	出力される XMLObjectOutput ストリーム。

printAttributes()

説明

属性ノードを出力します。XMLObjectOutput ストリームに出力できない場合は、IOException が発生します。

構文

```
public void printAttributes( oracle.xml.parser.v2.XMLOutputStream out,
                             String name,
                              String namespace);
```

パラメータ	説明
out	出力される XMLObjectOutput ストリーム。
name	属性名。
namespace	名前空間。

setNodeValue()

説明

要素のノード値を設定します。

構文

```
protected void setNodeValue( String value);
```

パラメータ	説明
value	ノードの値。

DTDClassGenerator クラス

DTDClassGenerator の説明

このクラスは、DTD または DTD に基づく XML ファイルに対応するクラスをバインドするデータを生成します。

DTDClassGenerator の構文

```
public class DTDClassGenerator extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.classgen.DTDClassGenerator
```

DTDClassGenerator のメソッド

表 7-6 DTDClassGenerator のメソッドの概要

メソッド	説明
DTDClassGenerator()（7-20 ページ）	DTDClassGenerator のデフォルトのコンストラクタです。
generate()（7-20 ページ）	doctype 要素をルートとして DTD を全検索し、Java クラスを生成します。
setGenerateComments()（7-20 ページ）	生成されたクラスに Javadoc コメントを生成するかどうかを判別します。
setJavaPackage()（7-21 ページ）	生成されたクラスのパッケージを設定します。
setOutputDirectory()（7-21 ページ）	DTD の Java ソース・コードが生成される出力ディレクトリを設定します。
setSerializationMode()（7-21 ページ）	DTD をシリアル化されたオブジェクトとして保存するか、またはテキスト・ファイルとして保存するかを判別します。
setValidationMode()（7-22 ページ）	生成されたクラスによって XML 文書を検証する必要があるかどうかを判別します。

DTDClassGenerator()

説明

DTDClassGenerator のデフォルトのコンストラクタです。

構文

```
public  DTDClassGenerator();
```

generate()

説明

doctype 要素をルートとして DTD を全検索し、Java クラスを生成します。

構文

```
public void generate( oracle.xml.parser.v2.DTD dtd,
                      String doctype);
```

パラメータ	説明
dtd	クラスの生成に使用される DTD。
doctype	ルート要素名。

setGenerateComments()

説明

生成されたクラスに Javadoc コメントを生成するかどうかを判別します。デフォルト値は true です。

構文

```
public void setGenerateComments( boolean comments);
```

パラメータ	説明
comments	Javadoc コメントを生成するかどうかを判別するブール値フラグ。

setJavaPackage()

説明

生成されたクラスのパッケージを設定します。デフォルトはパッケージなしです。

構文

```
public void setJavaPackage( java.util.Vector packageName);
```

パラメータ	説明
packageName	パッケージ名。

setOutputDirectory()

説明

DTD の Java ソース・コードが生成される出力ディレクトリを設定します。デフォルトは現在のディレクトリです。

構文

```
public void setOutputDirectory( String dir);
```

パラメータ	説明
dir	出力ディレクトリ。

setSerializationMode()

説明

DTD を、シリアル化されたオブジェクトとして保存するか、またはテキスト・ファイルとして保存するかを判別します。生成されたクラスを使用して XML ファイルを作成する場合、DTD をシリアル化するとパフォーマンスが向上します。

構文

```
public void setSerializationMode( boolean yes);
```

パラメータ	説明
yes	DTD を、シリアル化されたオブジェクトとして保存（true）するかどうかを判別するブール値フラグ。デフォルトでは、テキスト・ファイルとして保存（false）されます。

setValidationMode()

説明

生成されたクラスによって作成した XML 文書を検証する必要があるかどうかを判別します。デフォルト値は true です。

構文

```
public void setValidationMode( boolean yes);
```

パラメータ	説明
yes	XML 文書を検証するかどうかを判別するブール値フラグ。デフォルトは true です。

InvalidContentException クラス

InvalidContentException の説明

このクラスは、DTD Class Generator および Schema Class Generator によって発生する例外を定義します。

InvalidContentException の構文

```
public class InvalidContentException extends java.lang.Exception

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--oracle.xml.classgen.InvalidContentException
```

InvalidContentException の実装済インタフェース

```
java.io.Serializable
```

InvalidContentException のメソッド

InvalidContentException()

説明

コンストラクタです。次の表に、オプションを示します。

構文	説明
public InvalidContentException();	デフォルトのコンストラクタです。
public InvalidContentException(String s);	このコンストラクタでは、例外についての情報を含む入力文字列が使用されます。

パラメータ	説明
s	例外についての情報を含む文字列。

oracg クラス

oracg の説明

このクラスは、DTD または XML に対応する Java クラスを生成するためのコマンドライン・インタフェースを提供します。

oracg の構文

```
public class oracg extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.classgen.oracg
```

oracg のコマンドライン・オプション

表 7-7 oracg のコマンドライン・オプション

オプション	説明
-help	ヘルプ・メッセージのテキストを出力します。
-version	バージョン番号を出力します。
-dtd [-root <rootName>]	入力ファイルは、DTD ファイルまたは DTD に基づく XML ファイルです。
-schema <Schema File>	入力ファイルは、スキーマ・ファイルまたはスキーマに基づく XML ファイルです。
-outputDir <Output Dir>	Java ソースが生成されるディレクトリの名前です。
-package <Package Name>	生成された Java クラスのパッケージ名です。
-comment	生成された Java ソース・コードのコメントを生成します。

SchemaClassGenerator クラス

SchemaClassGenerator の説明

このクラスは、XML Schema に対応するクラスを生成します。

SchemaClassGenerator の構文

```
public class SchemaClassGenerator extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.classgen.SchemaClassGenerator
```

SchemaClassGenerator のメソッド

表 7-8 SchemaClassGenerator のメソッドの概要

メソッド	説明
SchemaClassGenerator()（7-26 ページ）	コンストラクタです。
generate()（7-26 ページ）	最上位の要素、simpleType 要素および complexType 要素に対応する Schema クラスを生成します。
setGenerateComments()（7-27 ページ）	Javadoc コメントを生成するかどうかを判別します。
setJavaPackage()（7-27 ページ）	各名前空間にユーザー定義の Java パッケージ名を割り当てます。
setOutputDirectory()（7-28 ページ）	Schema クラスの Java ソース・コードが生成される出力ディレクトリを設定します。

SchemaClassGenerator()

説明

コンストラクタです。次の表に、オプションを示します。

構文	説明
public SchemaClassGenerator();	Schema Class Generator のデフォルトの空のコンストラクタです。
public SchemaClassGenerator(String fileName)	このコンストラクタでは、XML Schema の記述子を含む入力文字列が使用されます。

パラメータ	説明
fileName	入力 XML Schema。

generate()

説明

各ノード上で createSchemaClass() をコールして、最上位の要素、simpleType 要素および complexType 要素に対応する Schema クラスを生成します。

構文

```
public void generate( oracle.xml.parser.schema.XMLSchema schema);
```

パラメータ	説明
schema	スキーマ・オブジェクト。

setGenerateComments()

説明

Javadoc コメントを生成するかどうかを判別します。デフォルトは `true` です。

構文

```
public void setGenerateComments( boolean comments)
```

パラメータ	説明
comments	Javadoc コメントを生成するかどうかを判別します。デフォルトは `true` です。

setJavaPackage()

説明

各名前空間にユーザー定義の Java パッケージ名を割り当てます。スキーマに定義された名前空間に対して問合せを実行する場合は、これらの名前空間の数が、ユーザーが指定したパッケージ名の数と一致する必要があります。一致しない場合は、エラーが発生します。

構文

```
public void setJavaPackage( oracle.xml.parser.schema.XMLSchema schema,
                            java.util.Vector pkgName);
```

パラメータ	説明
schema	XML Schema。
pkgName	コマンドラインを介して指定されたユーザー定義のパッケージ名を含むベクトル。

setOutputDirectory()

説明

Schema クラスの Java ソース・コードが生成される出力ディレクトリを設定します。デフォルトは現在のディレクトリです。

構文

```
public void setOutputDirectory( String dir);
```

パラメータ	説明
dir	出力ディレクトリ。

8

XML SQL Utility for Java

XML SQL Utility for Java（XSU）は、SQL 問合せから XML を生成したり格納します。

この章の内容は次のとおりです。

- OracleXMLQuery クラス
- OracleXMLSave クラス
- OracleXMLSQLException クラス
- OracleXMLSQLNoRowsException クラス

参照：

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

OracleXMLQuery クラス

OracleXMLQuery の説明

このクラスは、SQL 問合せを指定して、XML を生成します。

OracleXMLQuery の構文

```
public class OracleXMLQuery extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.sql.query.OracleXMLQuery
```

OracleXMLQuery のフィールド

表 8-1 OracleXMLQuery のフィールドの概要

フィールド	構文	説明
DTD	public static final int DTD	DTD の生成を指定します。
ERROR_TAG	public static final String ERROR_TAG	ERROR ドキュメントのデフォルトのタグ名を指定します。
MAXROWS_ALL	public static final int MAXROWS_ALL	すべての行を結果に含めます。
NONE	public static final int NONE	DTD を生成しないことを指定します。
ROW_TAG	public static final String ROW_TAG	ROW 要素のデフォルトのタグ名を指定します。
ROWIDATTR_TAG	public static final String ROWIDATTR_TAG	ROW 要素のデフォルトのタグ名を指定します。
ROWSET_TAG	public static final String ROWSET_TAG	ドキュメントのデフォルトのタグ名を指定します。
SCHEMA	public static final int SCHEMA	XML Schema が生成されるかどうか指定します。
SKIPROWS_ALL	public static final int SKIPROWS_ALL	結果内ですべての行をスキップします。

OracleXMLQuery のメソッド

表 8-2 OracleXMLQuery のメソッドの概要

メソッド	説明
OracleXMLQuery()（8-5 ページ）	クラス・コンストラクタです。
close()（8-5 ページ）	Oracle XML エンジンで作成されたオープン・リソースをクローズします。
getNumRowsProcessed()（8-6 ページ）	処理された行の数を戻します。
getXMLDOM()（8-6 ページ）	コンストラクタで指定されたオブジェクト・リレーショナル・データを、XML 文書に変換します。
getXMLMetaData()（8-7 ページ）	XML 文書の DTD または XMLSchema を戻します。
getXMLSAX()（8-7 ページ）	コンストラクタで指定されたオブジェクト・リレーショナル・データを、XML 文書に変換します。
getXMLSchema()（8-8 ページ）	指定された問合せに対応する XMLSchema を生成します。
getXMLString()（8-8 ページ）	コンストラクタで指定されたオブジェクト・リレーショナル・データを、XML 文書に変換します。
keepObjectOpen()（8-9 ページ）	XML データが取得されるオブジェクトの永続性を有効または無効にします。
removeXSLTParam()（8-9 ページ）	最上位のスタイルシートのパラメータ値を削除します。
setCollIdAttrName()（8-10 ページ）	コレクション要素の区切りタグの ID 属性名を設定します。
setDataHeader()（8-10 ページ）	XML データ・ヘッダーを設定します。
setDateFormat()（8-11 ページ）	XML 文書に生成日付の書式を設定します。
setEncoding()（8-11 ページ）	XML 文書に処理命令のエンコーディングを設定します。
setErrorTag()（8-11 ページ）	XML のエラー・ドキュメントを囲むタグを設定します。
setException()（8-12 ページ）	ユーザーに例外の指定を許可し、XSU に対処させます。
setMaxRows()（8-12 ページ）	XML に変換する行の最大数を設定します。
setMetaHeader()（8-12 ページ）	XML メタ・ヘッダーを設定します。
setRaiseException()（8-13 ページ）	例外を発生させるかどうかを XSU に指示します。

表 8-2 OracleXMLQuery のメソッドの概要（続き）

メソッド	説明
setRaiseNoRowsException()（8-13 ページ）	生成された XML 文書が空の場合に `OracleXMLNoRowsException` を発生させるかどうかを XSU に指示します。
setRowIdAttrName()（8-14 ページ）	行の囲みタグの ID 属性名を設定します。
setRowIdAttrValue()（8-14 ページ）	列値が行の囲みタグの ID 属性に割り当てられるスカラー列を指定します。
setRowsetTag()（8-14 ページ）	XML データセットを囲むタグを設定します。
setRowTag()（8-15 ページ）	データベース・レコードに対する XML 要素を囲むタグを設定します。
setSkipRows()（8-15 ページ）	スキップする行数を設定します。
setSQLToXMLNameEscaping()（8-15 ページ）	マップされた SQL オブジェクト名によって有効な XML 識別子が作成されない場合に XML タグをエスケープするかどうかを指定します。
setStylesheetHeader()（8-16 ページ）	スタイルシート・ヘッダーを設定します。
setXSLT()（8-16 ページ）	生成された XML に適用する XSL 変換を登録します。
setXSLTParam()（8-17 ページ）	最上位のスタイルシートのパラメータ値を設定します。
useLowerCaseTagNames()（8-17 ページ）	タグ名を小文字に設定します。
useNullAttributeIndicator()（8-18 ページ）	`null` 値であることを示すために、特殊な XML 属性を使用するか、または XML 文書からエンティティを省略するかを指定します。
useTypeForCollElemTag()（8-18 ページ）	コレクション要素の型名をコレクション要素のタグ名として使用するように XSU に指示します。
useUpperCaseTagNames()（8-18 ページ）	タグ名を大文字に設定します。

OracleXMLQuery()

説明

OracleXMLQuery オブジェクトのクラス・コンストラクタです。次の表に、オプションを示します。

構文	説明
public OracleXMLQuery(java.sql.Connection conn, java.sql.ResultSet rset);	データベース接続および JDBC の結果セット・オブジェクトから OracleXMLQuery を作成します。
public OracleXMLQuery(java.sql.Connection conn, String query);	データベース接続および SQL 問合せ文字列から OracleXMLQuery を作成します。
public OracleXMLQuery(oracle.xml.sql.dataset.OracleXMLDataSet dset);	データセットから OracleXMLQuery を作成します。

パラメータ	説明
conn	データベース接続。
rset	JDBC 結果セット・オブジェクト。
query	SQL 問合せ文字列。
dset	データセット。

close()

説明

Oracle XML エンジンで作成された、すべてのオープン・リソースをクローズします。ただし、ユーザーが提供するインスタンスの結果セットはクローズしません。

構文

```
public void close();
```

getNumRowsProcessed()

説明

処理された行の数を戻します。

構文

```
public long getNumRowsProcessed();
```

getXMLDOM()

説明

コンストラクタで指定されたオブジェクト・リレーショナル・データを、XML 文書に変換します。XML 文書の DOM 表現を戻します。次の表に、オプションを示します。

構文	説明
public org.w3c.dom.Document getXMLDOM();	XML 文書の DOM 表現を戻します。
public org.w3c.dom.Document getXMLDOM(int metaType);	引数を使用して、XSU が XML とともに生成する XML メタデータの型を指定します。現在、この値は無視され、XML メタデータは生成されません。XML 文書の DOM 表現を戻します。
public org.w3c.dom.Document getXMLDOM(org.w3c.dom.Node root);	null でない場合、この引数は、XML 文書のルート要素とみなされます。XML 文書の DOM 表現を戻します。
public org.w3c.dom.Document getXMLDOM(org.w3c.dom.Node root, int metaType);	null でない場合、root 引数は、XML 文書のルート要素とみなされます。metaType 引数は、XSU が XML とともに生成する XML メタデータの型を指定するために使用されます。現在、この値は無視され、XML メタデータは生成されません。XML 文書の DOM 表現を戻します。

パラメータ	説明
metaType	XML メタデータの型（NONE および SCHEMA）。
root	新しい XML を追加するルート・ノード。

getXMLMetaData()

説明

このファンクションは、getXML*() コール（getXMLDOM()、getXMLSAX()、getXMLSchema()、getXMLString() など）によって生成された XML 文書の DTD または XMLSchema を戻します。

構文

```
public String getXMLMetaData( int metaType,
                              boolean withVer);
```

パラメータ	説明
metaType	生成する XML メタデータの型（NONE または DTD）を判別します。
withVer	バージョン処理命令を生成するかどうかを判別します。

getXMLSAX()

説明

コンストラクタで指定されたオブジェクト・リレーショナル・データを、XML 文書に変換します。

構文

```
public void getXMLSAX( org.xml.sax.ContentHandler sax);
```

パラメータ	説明
sax	登録する ContentHandler オブジェクト。

getXMLSchema()

説明

このメソッドは、指定された問合せに対応する XML Schema を生成して戻します。

構文

```
public org.w3c.dom.Document[] getXMLSchema();
```

getXMLString()

説明

コンストラクタで指定されたオブジェクト・リレーショナル・データを、XML 文書に変換します。XML 文書の文字列表現を戻します。次の表に、オプションを示します。

構文	説明
public String getXMLString();	引数は使用されません。
public String getXMLString(int metaType);	metaType 引数は、XSU が XML とともに生成する XML メタデータの型を指定するために使用されます。
public String getXMLString(org.w3c.dom.Node root);	null でない場合、root 引数は、XML 文書のルート要素とみなされます。
public String getXMLString(org.w3c.dom.Node root, int metaType);	null でない場合、root 引数は、XML 文書のルート要素とみなされます。metaType 引数は、XSU が XML とともに生成する XML メタデータの型を指定するために使用されます。root 引数が null でない場合、DTD はリクエストされても生成されません。

パラメータ	説明
metaType	XML メタデータの型（NONE、DTD、SCHEMA または OracleXMLQuery クラスの静的フィールド）。
root	新しい XML を追加するルート・ノード。

keepObjectOpen()

説明

ResultSet オブジェクトを使用しないすべての getXML*() ファンクション（getXMLDOM()、getXMLSAX()、getXMLSchema()、getXMLString() など）のデフォルトの動作では、ResultSet オブジェクトおよび Statement オブジェクトがコールの終わりにクローズされます。getXML() を繰り返しコールして次の行セットを取得する永続機能を使用する必要がある場合は、このファンクションを true に設定してコールし、デフォルトの動作を停止する必要があります。OracleXMLQuery は、getXML() のコール後、ResultSet オブジェクトおよび Statement オブジェクトをクローズしなくなります。カーソル状態をクローズするには、close() ファンクションを明示的にコールする必要があります。

構文

```
public void keepObjectOpen( boolean alive);
```

パラメータ	説明
alive	オブジェクトをオープンにしておく必要があるかどうかを指定します。

removeXSLTParam()

説明

最上位のスタイルシートのパラメータ値を削除します。スタイルシートが登録されてない場合、このメソッドは動作しません。

構文

```
public void removeXSLTParam( String name);
```

パラメータ	説明
name	パラメータ名。

setCollIdAttrName()

説明

コレクション要素の区切りタグの id 属性名を設定します。null または空の文字列を設定すると、ROWID 属性が省略されます。

構文

```
public void setCollIdAttrName( String attrName);
```

パラメータ	説明
attrName	属性名。

setDataHeader()

説明

XML データ・ヘッダーを指定します。このデータ・ヘッダーは、XML のエンティティで、問合せによって生成された XML エンティティ（行セット）の先頭に追加されます。2 つのエンティティは、docTag 引数で指定されたタグで囲まれます。最後に指定したデータ・ヘッダーが、使用されるデータ・ヘッダーです。header パラメータに null を設定すると、データ・ヘッダーの設定が解除されます。

構文

```
public void setDataHeader( java.io.Reader header,
                           String docTag);
```

パラメータ	説明
header	ヘッダー。
docTag	データ・ヘッダーおよび行セットを囲むタグ。

setDateFormat()

説明

XML 文書に生成日付の書式を設定します。日付書式パターン（日付マスク）の構文は、java.text.SimpleDateFormat クラスの要件を満たす必要があります。マスクに null または空の文字列を設定すると、日付マスクの設定が解除されます。

構文

```
public void setDateFormat( String mask);
```

パラメータ	説明
mask	日付マスク。

setEncoding()

説明

XML 文書に処理命令（PI）のエンコーディングを設定します。エンコーディングとして null または空の文字列が指定された場合、デフォルトのキャラクタ・セットが処理命令のエンコーディングに指定されます。

構文

```
public void setEncoding( String enc)
```

パラメータ	説明
enc	XML 文書のエンコーディング（エンコーディングの IANA 名）。

setErrorTag()

説明

XML のエラー・ドキュメントを囲むタグを設定します。

構文

```
public void setErrorTag( String tag);
```

パラメータ	説明
tag	タグ名。

setException()

説明

ユーザーに例外の指定を許可し、XSU に対処させます。

構文

```
public void setException( java.lang.Exception e);
```

パラメータ	説明
e	XSU が処理する例外。

setMaxRows()

説明

XML に変換する行の最大数を設定します。デフォルトでは、最大数は設定されていません。無制限の最大数を明示的に指定する方法は、MAXROWS_ALL フィールドを参照してください。

構文

```
public void setMaxRows( int rows);
```

パラメータ	説明
rows	生成する行の最大数。

setMetaHeader()

説明

XML メタ・ヘッダーを設定します。ヘッダーは、設定されると、オブジェクトが生成した各 XML 文書のメタデータ部（DTD または XML Schema）の先頭に挿入されます。最後に指定したメタ・ヘッダーが、使用されるメタ・ヘッダーです。header を null または空の文字列に設定すると、メタ・ヘッダーの設定が解除されます。

構文

```
public void setMetaHeader( java.io.Reader header);
```

パラメータ	説明
header	ヘッダー。

setRaiseException()

説明

例外を発生させるかどうかを XSU に指示します。このコールが行われなかった場合または flag 引数を false に設定した場合、XSU は SQL 例外をキャッチし、その例外のメッセージから XML 文書を生成します。

構文

```
public void setRaiseException( boolean flag);
```

パラメータ	説明
flag	例外を発生させる必要があるかどうかを判別します。

setRaiseNoRowsException()

説明

生成された XML 文書が空の場合に OracleXMLNoRowsException を発生させるかどうかを XSU に指示します。デフォルトでは、例外は発生しません。

構文

```
public void setRaiseNoRowsException( boolean flag);
```

パラメータ	説明
flag	データがない場合に OracleXMLNoRowsException を発生させるかどうかを判別します。

setRowIdAttrName()

説明

行の囲みタグの ID 属性名を設定します。null または空の文字列を設定すると、ROWID 属性が省略されます。

構文

```
public void setRowIdAttrName( String attrName);
```

パラメータ	説明
attrName	属性名。

setRowIdAttrValue()

説明

列値が行の囲みタグの ID 属性に割り当てられるスカラー列を指定します。null または空の文字列を設定すると、ROWID 属性に行カウント値（0、1、2 など）が割り当てられます。

構文

```
public void setRowIdAttrValue( String colName);
```

パラメータ	説明
colName	ROWID 属性に値が割り当てられる列。

setRowsetTag()

説明

XML データセットを囲むタグを設定します。

構文

```
public void setRowsetTag( String tag);
```

パラメータ	説明
tag	タグ名。

setRowTag()

説明

データベース・レコードに対応する XML 要素を囲むタグを設定します。

構文

```
public void setRowTag( String tag);
```

パラメータ	説明
tag	タグ名。

setSkipRows()

説明

スキップする行数を設定します。デフォルトでは、0（ゼロ）行がスキップされます。すべての行をスキップするには、SKIPROWS_ALL を使用します。

構文

```
public void setSkipRows( int rows);
```

パラメータ	説明
rows	スキップする行数。

setSQLToXMLNameEscaping()

説明

XML 識別子にマップされた SQL オブジェクト名が有効な XML 識別子でない場合に XML タグをエスケープするかどうかを指定します。

構文

```
public void setSQLToXMLNameEscaping( boolean flag);
```

パラメータ	説明
flag	SQL 識別子を XML 識別子にエスケープするかどうかを指定します。

setStylesheetHeader()

説明

生成された XML 文書にスタイルシート・ヘッダー（スタイルシート処理命令）を設定します。引数に null を設定すると、スタイルシート・ヘッダーおよびスタイルシートの型の設定が解除されます。次の表に、オプションを示します。

構文	説明
public void setStylesheetHeader(String uri);	スタイルシートの URI を使用して、スタイルシート・ヘッダーを設定します。
public void setStylesheetHeader(String uri, String type);	スタイルシートの URI およびスタイルシートの型を使用して、スタイルシート・ヘッダーを設定します。

パラメータ	説明
uri	スタイルシートの URI。
type	スタイルシートの型（デフォルトは「text/xsl」）。

setXSLT()

説明

生成された XML に適用する XSL 変換を登録します。スタイルシートがすでに登録されている場合は、新しいスタイルシートに置き換えられます。スタイルシートの登録を解除するには、引数に null を設定します。次の表に、オプションを示します。

構文	説明
public void setXSLT(java.io.Reader stylesheet, String ref);	スタイルシート・パラメータがデータとして設定されます。
public void setXSLT(java.lang.String stylesheet, String ref);	スタイルシート・パラメータがドキュメントの URI として設定されます。

パラメータ	説明
stylesheet	スタイルシート。
ref	挿入、インポートおよび外部エンティティの URL。

setXSLTParam()

説明

最上位のスタイルシートのパラメータ値を設定します。パラメータ値は、有効な XPath 式が想定されます。したがって、文字列リテラル値を明示的に囲む必要があります。スタイルシートが登録されてない場合、このメソッドは動作しません。

構文

```
public void setXSLTParam( String name,
                          String value);
```

パラメータ	説明
name	パラメータ名。
value	XPath 式としてのパラメータ値。

useLowerCaseTagNames()

説明

すべてのタグ名を小文字に設定します。すべての必要なタグを設定してから、このメソッドをコールします。

構文

```
public void useLowerCaseTagNames();
```

useNullAttributeIndicator()

説明

null 値であることを示すために、特殊な XML 属性を使用するか、または XML 文書からエンティティを省略するかを指定します。

構文

```
public void useNullAttributeIndicator( boolean flag);
```

パラメータ	説明
flag	属性を使用して null を示す必要があるかどうかを判別します。

useTypeForCollElemTag()

説明

デフォルトでは、コレクション要素のタグ名は、コレクションのタグ名の後に「_item」が続く名前になります。引数に true を指定してこのメソッドをコールすると、XSU はコレクション要素の型名をコレクション要素のタグ名として使用します。

構文

```
public void useTypeForCollElemTag( boolean flag);
```

パラメータ	説明
flag	列の要素型を使用してタグ名を示す必要があるかどうかを判別します。

useUpperCaseTagNames()

説明

すべてのタグ名を大文字に設定します。このメソッドは、すべての必要なタグを設定した後でのみ、コールしてください。

構文

```
public void useUpperCaseTagNames();
```

OracleXMLSave クラス

OracleXMLSave の説明

このクラスは、XML からオブジェクト・リレーショナル表またはオブジェクト・リレーショナル・ビューへの正規マッピングをサポートします。挿入、更新および削除がサポートされます。最初に、DML 操作を実行する必要がある表名を渡すことによって、クラスを作成します。作成後、ユーザーはこの表に対して自由に挿入、更新および削除を実行できます。

このクラスで提供されている有効なファンクションは、更新または削除のためのキー列を識別したり、更新中の列を制限するために有効です。

OracleXMLSave の構文

```
public class OracleXMLSave extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.sql.dml.OracleXMLSave
```

OracleXMLSave のフィールド

表 8-3 OracleXMLSave のフィールドの概要

フィールド	構文	説明
DATE_FORMAT	public static final String DATE_FORMAT	setDateFormat で使用される日付書式です。
DEFAULT_BATCH_SIZE	public static int DEFAULT_BATCH_SIZE	挿入時のデフォルトのバッチ・サイズは 17 です。
xDocIsEsc	public boolean xDocIsEsc	XML 文書に対して SQL から XML へのエスケープが行われているかどうかを示します。

OracleXMLSave のメソッド

表 8-4 OracleXMLSave のメソッドの概要

メソッド	説明
OracleXMLSave()（8-21 ページ）	Save クラスに対するパブリック・コンストラクタです。
close()（8-21 ページ）	オブジェクトに対応するすべてのコンテキストのクローズまたは割当て解除を行います。
deleteXML()（8-21 ページ）	XML 文書に基づいて表の行を削除します。
getURL()（8-22 ページ）	ファイル名または URL を指定して URL オブジェクトを戻します。
insertXML()（8-23 ページ）	指定された表に XML 文書を挿入します。
removeXSLTParam()（8-24 ページ）	最上位のスタイルシートのパラメータ値を削除します。
setBatchSize()（8-24 ページ）	DML 操作中に使用されるバッチ・サイズを変更します。
setCommitBatch()（8-25 ページ）	コミットのバッチ・サイズを設定します。
setDateFormat()（8-25 ページ）	XML 文書の日付書式を、XSU に通知します。
setIgnoreCase()（8-26 ページ）	XML 要素とデータベースの列または属性の間で大 / 小文字の区別をしない一致検索を実行するように、XSU に指示します。
setKeyColumnList()（8-26 ページ）	更新または削除中に、データベース表内の特定の行を識別するために使用する、列のリストを設定します。
setPreserveWhitespace()（8-27 ページ）	空白を保持するかどうかを XSU に指示します。
setRowTag()（8-27 ページ）	XML 文書で使用するタグを指定し、各行の値に対応する XML 要素を囲みます。
setSQLToXMLNameEscaping()（8-27 ページ）	SQL オブジェクト名によって有効な XML 識別子が作成されない場合、XML タグをエスケープするかどうかを指定します。
setUpdateColumnList()（8-28 ページ）	更新する列の値を設定します。
setXSLT()（8-28 ページ）	生成された XML に適用する XSL 変換を登録します。
setXSLTParam()（8-29 ページ）	最上位のスタイルシートのパラメータ値を設定します。
updateXML()（8-29 ページ）	XML 文書を指定して表を更新します。

OracleXMLSave()

説明

OracleXMLSave クラスに対するパブリック・コンストラクタです。

構文

```
public  OracleXMLSave( java.sql.Connection oconn,
                       String tabName;
```

パラメータ	説明
oconn	接続オブジェクト（データベースへの接続）。
tableName	更新する必要がある表の名前。

close()

説明

オブジェクトに対応するすべてのコンテキストのクローズまたは割当て解除を行います。

構文

```
public void close();
```

deleteXML()

説明

XML 文書に基づいて表の行を削除します。処理された XML の ROW 要素の数を戻します。この数は、XML 文書を介して選択された行が、表内の行を一意に識別するかどうかによって削除されたデータベースの行数と同等になる場合とならない場合があります。

デフォルトでは、削除処理によって、すべての要素値とそれに対応する列名が一致します。入力ドキュメントの各 ROW 要素は、表に対する個別の DELETE 文として受け入れられます。`setKeyColumnList()` を使用すると、一致する行を識別して削除し、他の要素は無視する列のリストが設定されます。これは、（DELETE 文がキャッシュされるため）一致の使用時、表内の複数行を削除する場合に有効なメソッドです。このメソッドを使用しない場合は、入力ドキュメントの各 ROW 要素に対して新しい DELETE 文を作成する必要があります。次の表に、オプションを示します。

構文	説明
public int deleteXML(org.w3c.dom.Document doc);	XML 文書は DOM 形式を取ります。
public int deleteXML(java.io.InputStream xmlStream);	XML 文書はストリーム形式を取ります。
public int deleteXML(java.io.Reader xmlReader);	XML 文書はリーダー形式を取ります。
public int deleteXML(String xmlDoc);	XML 文書は文字列形式を取ります。
public int deleteXML (java.net.URL url);	URL を介して XML 文書にアクセスします。

パラメータ	説明
doc	DOM 形式の XML 文書。
xmlStream	ストリーム形式の XML 文書。
xmlReader	リーダー形式の XML 文書。
xmlDoc	文字列形式の XML 文書。
url	表の行の削除に使用するドキュメントの URL。

getURL()

説明

ファイル名または URL を指定して、ターゲット・エンティティを識別する URL オブジェクトを戻します。設定された引数が有効な URL 形式（http:// や file:// など）でない場合、このメソッドは引数に「file://」を追加して、引数の修正を試みます。null または空の文字列が設定された場合は、null を戻します。

構文

```
public static java.net.URL getURL( String target);
```

パラメータ	説明
target	ファイル名または URL 文字列。

insertXML()

説明

指定された表に XML 文書を挿入します。挿入された行の数を戻します。

- 要素名と列名を一致させることによって表に値を挿入し、入力ドキュメント内で欠落しているすべての要素に null 値を挿入します。setUpdateColumnList() を使用すると、残りの列に null 値が挿入されず、かわりにデフォルト値が使用されます。
- すべてのキー列のリストを設定するには、setKeyColumnList() を使用します。
- 更新する列のリストを設定するには、setUpdateColumnList() を使用します。

次の表に、オプションを示します。

構文	説明
public int insertXML(org.w3c.dom.Document doc);	DOM から XML 文書を挿入します。
public int insertXML(java.io.InputStream xmlStream);	入力ストリームから XML 文書を挿入します。
public int insertXML(java.io.Reader xmlStream);	リーダーから XML 文書を挿入します。
public int insertXML(String xmlDoc);	文字列から XML 文書を挿入します。
public int insertXML(java.net.URL url);	URL から XML 文書を挿入します。

パラメータ	説明
doc	表に行を挿入するための DOM。
xmlStream	表への行の挿入に使用されるデータのストリーム。
xmlDoc	表への行の挿入に使用される文字列。

パラメータ	説明
url	表への行の挿入に使用されるドキュメントの URL。

removeXSLTParam()

説明

最上位のスタイルシートのパラメータ値を削除します。スタイルシートが登録されてない場合、このメソッドは動作しません。

構文

```
public void removeXSLTParam( String name);
```

パラメータ	説明
name	パラメータ名。

setBatchSize()

説明

DML 操作中に使用されるバッチ・サイズを変更します。挿入、更新または削除を実行する場合、バッチ処理を行って I/O 循環を最小化することをお薦めします。ただし、この操作の実行中は、バインド値を格納するために、より多くのキャッシュが必要になります。バッチ処理を行う場合、コミットは、バッチの側からのみ発生します。バッチ内の 1 文が正常に実行されなかった場合、バッチ処理全体がロールバックされます。この動作を防止するには、バッチ・サイズを 1 に設定します。デフォルトのバッチ・サイズは DEFAULT_BATCH_SIZE です。

構文

```
public void setBatchSize( int size);
```

パラメータ	説明
size	すべての DML に使用するバッチ・サイズ。

setCommitBatch()

説明

コミットのバッチ・サイズを設定します。これは、実行後にコミットする必要がある記録の挿入の回数を意味します。`size < 1` の場合またはセッションが自動コミット・モードの場合、XSU は明示的なコミットを行いません。コミットのデフォルトのバッチ・サイズは `0`（ゼロ）です。

構文

```
public void setCommitBatch( int size);
```

パラメータ	説明
size	コミットのバッチ・サイズ。

setDateFormat()

説明

XML 文書の日付書式を、XSU に通知します。デフォルトでは、OracleXMLSave は、日付書式を 'MM/dd/yyyy HH:mm:ss' であると想定しています。このファンクションをコールすると、このデフォルトの書式をオーバーライドできます。日付書式パターン（日付マスク）の構文は、java.text.SimpleDateFormat クラスの要件を満たす必要があります。マスクに `null` または空の文字列を設定すると、デフォルトのマスク `OracleXMLSave.DATE_FORMAT` が使用されます。

構文

```
public void setDateFormat( String mask);
```

パラメータ	説明
mask	日付マスク。

setIgnoreCase()

説明

XSU は、要素名（XML タグ）に基づいて XML 要素をデータベースの列または属性にマップします。このファンクションは、大 / 小文字を区別しない一致検索を実行するように、XSU に指示します。これは、Save オブジェクトの作成時に実行されるメタデータのキャッシュに影響を与える場合があります。

構文

```
public void setIgnoreCase( boolean ignore);
```

パラメータ	説明
ignore	XML 文書内のタグで大 / 小文字を無視する必要があるかどうかを指定します。

setKeyColumnList()

説明

更新または削除中に、データベース表内の特定の行を識別するために使用する、列のリストを設定します。このコールは、挿入の場合は無視されます。更新を実行する前に、キー列を指定する必要があります。削除用のオプションです。このキー列を指定すると、XML 文書内のこれらのタグの値は、更新または削除するデータベースの行を識別するために使用されます。現在、XML 文書内でキー列の大 / 小文字を指定できないため、キー列自体の値を更新する方法はありません。

構文

```
public void setKeyColumnList( String[] keyColNames);
```

パラメータ	説明
keyColNames	キーとして使用される列のリスト名。

setPreserveWhitespace()

説明

空白を保持するかどうかを XSU に指示します。

構文

```
public void setPreserveWhitespace( boolean flag);
```

パラメータ	説明
flag	空白を保持する必要があるかどうかを指定します。

setRowTag()

説明

XML 文書で使用するタグを指定し、各行の値に対応する XML 要素を囲みます。この値を null に指定すると、ROW タグが存在せず、ドキュメントの最上位の要素が行自体に対応することを表します。

構文

```
public void setRowTag( String rowTag);
```

パラメータ	説明
rowTag	タグ名。

setSQLToXMLNameEscaping()

説明

XML 識別子にマップされた SQL オブジェクト名が有効な XML 識別子でない場合、XML タグをエスケープするかどうかを指定します。

構文

```
public void setSQLToXMLNameEscaping( boolean flag);
```

パラメータ	説明
flag	SQL を XML にエスケープする必要があるかどうかを指定します。

setUpdateColumnList()

説明

更新する列の値を設定します。挿入および更新には適用されますが、削除には適用されません。

- 挿入の場合、デフォルトでは、表のすべての列に値が挿入されます。
- 更新の場合、デフォルトでは XML 文書の ROW 要素にあるタグに対応する列のみが更新されます。列を指定すると指定した列のみが、UPDATE 文または INSERT 文で更新されます。ドキュメント内の他のすべての要素は無視されます。

構文

```
public void setUpdateColumnList( String[] updColNames);
```

パラメータ	説明
updColNames	更新する列の文字列リスト。

setXSLT()

説明

生成された XML に適用する XSL 変換を登録します。スタイルシートがすでに登録されている場合は、新しいスタイルシートに置き換えられます。スタイルシートの登録を解除するには、`stylesheet` 引数に `null` を設定します。次の表に、オプションを示します。

構文	説明
public void setXSLT(java.io.Reader stylesheet, String ref);	スタイルシート・パラメータがデータとして設定されます。
public void setXSLT(String stylesheet, String ref);	スタイルシート・パラメータがドキュメントの URI として設定されます。

パラメータ	説明
stylesheet	スタイルシートの URI。
ref	挿入、インポートおよび外部エンティティの URL。

setXSLTParam()

説明

最上位のスタイルシートのパラメータ値を設定します。パラメータ値は、有効な XPath 式が想定されます（したがって、文字列リテラル値を明示的に囲む必要があります)。スタイルシートが登録されてない場合、このメソッドは動作しません。

構文

```
public void setXSLTParam( String name,
                          String value);
```

パラメータ	説明
name	パラメータ名。
value	XPath 式としてのパラメータ値。

updateXML()

説明

XML 文書を指定して表を更新します。処理された XML 要素の数を戻します。この数は、XML 文書を介して選択された行が、表内の行を一意に識別するかどうかによって変更されたデータベースの行数と同等になる場合とならない場合があります。

- 更新には、指定された表内で更新する行を一意に識別するために使用する、キー列のリストが必要です。デフォルトでは、更新はキー列のリストを使用し、XML 文書内の対応する値と一致させて特定の行を識別した後、その XML 文書内に同等の要素が存在する表のすべての列を更新します。入力ドキュメントの各 ROW 要素は、表に対する個別の UPDATE 文として処理されます。
- 更新する列のリストを指定して、必要な列のみを更新し、XML 文書に存在する他の要素を無視することができます。このメソッドは、入力 XML 文書に複数の行が存在する場合、UPDATE 文自体がキャッシュおよびバッチ処理されるため、非常に有効です。

- すべてのキー列のリストを設定するには、`setKeyColumnList()` を使用します。
- 更新する列のリストを設定するには、`setUpdateColumnList()` を使用します。

次の表に、オプションを示します。

構文	説明
public int updateXML(org.w3c.dom.Document doc);	DOM ツリー形式の XML 文書を指定して、表を更新します。
public int updateXML(java.io.InputStream xmlStream);	ストリーム形式の XML 文書を指定して、表を更新します。
public int updateXML(java.io.Reader xmlStream);	リーダー形式の XML 文書を指定して、表を更新します。
public int updateXML(String xmlDoc);	文字列形式の XML 文書を指定して、表を更新します。
public int updateXML(java.net.URL url);	提供された XML 文書内の要素の値に基づいて、データベース内の列を更新します。

パラメータ	説明
doc	DOM ツリー形式の XML 文書。
xmlStream	ストリーム形式の XML 文書。
xmlDoc	文字列形式の XML 文書。
url	表の更新に使用するドキュメントの URL。

OracleXMLSQLException クラス

OracleXMLSQLException の説明

このクラスは、XSU によって発生したすべての例外を管理します。

OracleXMLSQLException の構文

```
public class OracleXMLSQLException extends java.lang.RuntimeException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.lang.RuntimeException
                    |
                    +--oracle.xml.sql.OracleXMLSQLException
```

OracleXMLSQLException のダイレクト・サブクラス

- OracleXMLSQLNoRowsException クラス

OracleXMLSQLException の実装済インタフェース

- `java.io.Serializable`

OracleXMLSQLException のメソッド

表 8-5 OracleXMLSQLException のメソッドの概要

メソッド	説明
OracleXMLSQLException()（8-32 ページ）	新しい OracleXMLSQLException を作成します。
getErrorCode()（8-33 ページ）	発生した SQL エラー・コードを戻します。
getParentException()（8-33 ページ）	元の例外がある場合、それを戻します。ない場合は、`null` を戻します。
getXMLErrorString()（8-34 ページ）	XML エラー文字列を、指定されたエラー・タグ名で出力します。

表 8-5 OracleXMLSQLException のメソッドの概要（続き）

メソッド	説明
getXMLSQLErrorString()（8-34 ページ）	エラー・メッセージに SQL パラメータを出力します。
setErrorTag()（8-34 ページ）	XML エラー・レポートの生成に使用するエラー・タグを設定します。

OracleXMLSQLException()

説明

新しい OracleXMLSQLException を作成します。次の表に、オプションを示します。

構文	説明
public OracleXMLSQLException(Exception e);	引数として渡された親の例外を設定します。
public OracleXMLSQLException(Exception e, String errorTagName);	引数として渡されたエラー・タグ名を設定します。
public OracleXMLSQLException(String message);	戻されるエラー・メッセージを設定します。
public OracleXMLSQLException(String message, Exception e);	戻される親の例外およびエラー・メッセージを設定します。
public OracleXMLSQLException(String message, Exception e, String errorTagName);	使用されるエラー・メッセージ、親の例外およびエラー・タグを設定します。
public OracleXMLSQLException(String message, int errorCode);	エラー・メッセージおよび SQL エラー・コードを設定します。

構文	説明
public OracleXMLSQLException(String message, int errorCode, String errorTagName);	使用されるエラー・メッセージ、SQL エラー・コードおよびエラー・タグを設定します。
public OracleXMLSQLException(String message, String errorTagName);	使用されるエラー・メッセージおよびエラー・タグを設定します。

パラメータ	説明
e	例外。
errorTagName	エラー・タグ名。
message	エラー・メッセージ。
errorCode	SQL エラー・コード。

getErrorCode()

説明

発生した SQL エラー・コードを戻します。

構文

```
public int getErrorCode();
```

getParentException()

説明

元の例外がある場合、それを戻します。ない場合は、null を戻します。

構文

```
public java.lang.Exception getParentException();
```

getXMLErrorString()

説明

XML エラー文字列を、指定されたエラー・タグ名で出力します。

構文

```
public String getXMLErrorString();
```

getXMLSQLErrorString()

説明

エラー・メッセージにも SQL パラメータを出力します。

構文

```
public String getXMLSQLErrorString();
```

setErrorTag()

説明

XML エラー・レポートを生成するために、`getXMLErrorString()` および `getXMLSQLErrorString()` で使用されるエラー・タグ名を設定します。

構文

```
public void setErrorTag( String tagName);
```

パラメータ	説明
tagName	エラー・タグ名。

OracleXMLSQLNoRowsException クラス

OracleXMLSQLNoRowsException の説明

このクラスは、行が存在しない場合に発生する例外です。

OracleXMLSQLNoRowsException の構文

```
public class OracleXMLSQLNoRowsException extends OracleXMLSQLException

java.lang.Object
 |
 +--java.lang.Throwable
     |
     +--java.lang.Exception
          |
          +--java.lang.RuntimeException
               |
               +--OracleXMLSQLException
                    |
                    +--oracle.xml.sql.OracleXMLSQLNoRowsException
```

OracleXMLSQLNoRowsException の実装済インタフェース

```
java.io.Serializable
```

OracleXMLSQLNoRowsException のメソッド

OracleXMLSQLNoRowsException()

新しい OracleXMLSQLNoRowsException を作成します。次の表に、オプションを示します。

構文	説明
public OracleXMLSQLNoRowsException();	デフォルトのクラス・コンストラクタです。
public OracleXMLSQLNoRowsException(String errorTag);	引数として渡されたエラー・タグを設定します。

パラメータ	説明
errorTag	エラー・タグ。

9

XSQL Pages パブリッシング・フレームワーク for Java

XSQL Pages パブリッシング・フレームワークは、oracle.xml.xsql パッケージに含まれています。これは、Oracle XSQL Servlet ともいいます。

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

oracle.xml.xsql パッケージ

oracle.xml.xsql の説明

このパッケージは、Oracle XSQL Pages パブリッシング・フレームワーク・エンジンです。表 9-1 に、このパッケージのクラスおよびインタフェースの概要を示します。

表 9-1 oracle.xml.xsql のクラスおよびインタフェースの概要

クラスおよびインタフェース	説明
XSQLActionHandler インタフェース	すべての XSQL アクション・エレメント・ハンドラが実装する必要があるインタフェースです。
XSQLActionHandlerImpl クラス	ユーザー独自のカスタム・ハンドラを作成するために拡張可能な XSQLActionHandler の基本実装です。
XSQLPageRequest インタフェース	XSQL Pages に対するリクエストを表すインタフェースです。
XSQLParserHelper クラス	一般的な XML 解析ルーチンです。
XSQLRequest クラス	XSQL ページに対するリクエストのプログラム処理を行います。
XSQLRequestObjectListener インタフェース	このインタフェースを実装したオブジェクトは、現行のページ・リクエスト処理の完了の通知を受け取ることができます。
XSQLServletPageRequest クラス	サーブレット・ベースの XSQL ページ・リクエストのための XSQLPageRequest の実装です。
XSQLStylesheetProcessor クラス	XSLT スタイルシート処理エンジンです。
XSQLConnectionManager インタフェース	組込み Connection Manager の実装をオーバーライドするために実装される必要があります。
XSQLConnectionManagerFactory インタフェース	組込み Connection Manager の実装をオーバーライドするために実装される必要があります。
XSQLDocumentSerializer インタフェース	すべての XSQL シリアライザによって実装される必要があります。XSQL シリアライザは、XSQL データのページを XML 文書としてシリアル化してプリント・ライターに送信します。

XSQLActionHandler インタフェース

XSQLActionHandler の説明

このインタフェースは、すべての XSQL アクション・エレメント・ハンドラによって実装の必要があるインタフェースです。

XSQL ページで <xsql:xxxx> という形式の XSQL アクション要素が検出された場合、XSQL ページ・プロセッサは、次の方法によって、対応する XSQL アクション・ハンドラを起動します。

- 引数なしのコンストラクタを使用して、ハンドラのインスタンスを作成します。
- XSQL アクション・ハンドラの handleAction() メソッドをコールします。

注意 : 処理中の XSQL ページに対して接続が指定されていない場合は、conn パラメータが `null` になる可能性があります。

XSQLActionHandler の構文

```
public interface XSQLActionHandler
```

XSQLActionHandler のクラスの実装

`XSQLActionHandlerImpl` クラス

XSQLActionHandler のメソッド

表 9-2 XSQLActionHandler のメソッドの概要

メソッド	説明
handleAction()（9-4 ページ）	通常、コードを実行し、新しい子である DOM ノードを rootNode に追加することによって、アクションを処理します。
init()（9-4 ページ）	アクション・ハンドラを初期化します。

handleAction()

説明

通常、コードを実行し、新しい子である DOM ノードを rootNode に追加することによって、アクションを処理します。

XSQL ページ・プロセッサは、処理中の XSQL ページのアクション要素を、handleAction メソッドによって rootNode に追加されたノードのドキュメント・フラグメントに置換します。

構文

```
public void handleAction( oracle.xml.xsql.Node rootNode);
```

パラメータ	説明
rootNode	生成されたドキュメント・フラグメントのルート・ノード。

init()

説明

アクション・ハンドラを初期化します。

構文

```
public void init( XSQLPageRequest env,
                  oracle.xml.xsql.Element e);
```

パラメータ	説明
env	XSQLPageRequest オブジェクト。
e	処理中の Action Element を表す DOM 要素。

XSQLActionHandlerImpl クラス

XSQLActionHandlerImpl の説明

このクラスは、カスタム・ハンドラを作成するために拡張可能な XSQLActionHandler の基本となる実装です。一連の有効な補助メソッドが含まれます。このクラスを拡張し、`init()` メソッドをオーバーライドするには、`super.init(env,e)` をコールする必要があります。

XSQLActionHandlerImpl の構文

```
public abstract class XSQLActionHandlerImpl extends java.lang.Object implements
XSQLActionHandler Interface

java.lang.Object
  |
  +--oracle.xml.xsql.XSQLActionHandlerImpl
```

XSQLActionHandlerImpl の実装済インタフェース

```
XSQLActionHandler Interface
```

XSQLActionHandlerImpl のメソッド

表 9-3 XSQLActionHandlerImp のメソッドの概要

メソッド	説明
XSQLActionHandlerImpl()（9-5 ページ）	クラス・コンストラクタです。
init()（9-6 ページ）	アクション・ハンドラを初期化します。

XSQLActionHandlerImpl()

説明

クラス・コンストラクタです。

構文

```
public  XSQLActionHandlerImpl();
```

init()

説明

アクション・ハンドラを初期化します。

構文

```
public void init( XSQLPageRequest env,
                  oracle.xml.xsql.Element e);
```

パラメータ	説明
env	XSQLPageRequest コンテキスト。
e	アクション要素。

XSQLPageRequest インタフェース

XSQLPageRequest の説明

このインタフェースは、XSQL Pages に対するリクエストを表すインタフェースです。

XSQLPageRequest の構文

```
public interface XSQLPageRequest
```

XSQLPageRequest のクラスの実装

```
XSQLPageRequestImpl Class
```

XSQLPageRequest のメソッド

表 9-4 XSQLPageRequest() のメソッドの概要

メソッド	説明
createNestedRequest()（9-9 ページ）	ネストしたリクエストのインスタンスを戻します。
getConnectionName()（9-10 ページ）	リクエストに使用されている接続の名前を戻します。接続が設定または使用されてない場合は、null になる場合があります。
getErrorWriter()（9-10 ページ）	リクエストを処理するエラーを出力するためのプリント・ライターを戻します。
getJDBCConnection()（9-10 ページ）	リクエストに対して JDBC 接続を使用中にします。（null になる可能性があります。）
getPageEncoding()（9-10 ページ）	リクエストに対応するソース XSQL ページのエンコーディングを戻します。
getParameter()（9-11 ページ）	リクエストされたパラメータ値を戻します。
getPostedDocument()（9-11 ページ）	リクエストに対してポストされた XML の内容を、XML 文書として戻します。
getRequestParamsAsXMLDocument()（9-11 ページ）	リクエスト・パラメータの内容を、XML 文書として戻します。
getRequestType()（9-11 ページ）	作成しているページ・リクエストのタイプを識別する文字列を戻します。
getSourceDocumentURI()（9-12 ページ）	リクエストされたドキュメントの URI の文字列表現を戻します。

表 9-4 XSQLPageRequest() のメソッドの概要（続き）

メソッド	説明
getStylesheetParameter() (9-12 ページ)	スタイルシート・パラメータを名前で取得します。
getStylesheetParameter() (9-12 ページ)	スタイルシート・パラメータの名前の列挙を取得します。
getStylesheetURI() (9-12 ページ)	結果処理に使用されるスタイルシートの URI を戻します。
getUserAgent() (9-13 ページ)	リクエストしているプログラムの文字列識別子を戻します。
getWriter() (9-13 ページ)	ページ・リクエストの結果の出力に使用されるプリント・ライターを戻します。
getXSQLConnection() (9-13 ページ)	リクエストに使用されている XSQLConnection オブジェクトを取得します。null になる場合があります。
isIncludedRequest() (9-13 ページ)	リクエストが別のリクエストに含まれている場合、true を戻します。
isOracleDriver() (9-14 ページ)	現行の接続が Oracle JDBC ドライバを使用している場合、true を戻します。
printedErrorHeader() (9-14 ページ)	Error Header が出力されたかどうかを戻します。
requestProcessed() (9-14 ページ)	ページ・リクエストでリクエスト末尾処理を実行できるようにします。
setConnectionName() (9-14 ページ)	リクエストに使用する接続名を設定します。
setContentType() (9-15 ページ)	結果ページの Content Type を設定します。
setIncludingRequest() (9-15 ページ)	リクエストに対して Including Page Request オブジェクトを設定します。
setPageEncoding() (9-15 ページ)	リクエストに対応するソース XSQL ページのエンコーディングを設定します。
setPageParam() (9-16 ページ)	動的ページ・パラメータ値を設定します。
setPostedDocument() (9-16 ページ)	ポストされるドキュメントのプログラムを設定できるようにします。
setPrintedErrorHeader() (9-16 ページ)	Error Header が出力されるかどうかを設定します。
setStylesheetParameter() (9-17 ページ)	対応するスタイルシートへ渡されるパラメータ値を設定します。

表 9-4 XSQLPageRequest() のメソッドの概要（続き）

メソッド	説明
setStylesheetURI()（9-17 ページ）	結果処理に使用されるスタイルシートの URI を設定します。
translateURL()（9-18 ページ）	リクエストのベース URI に対して相対 URI に変換された絶対 URL を表す文字列を戻します。
useConnectionPooling()（9-18 ページ）	リクエストに対して接続プーリングが必要とされる場合、`true` を戻します。
useHTMLErrors()（9-18 ページ）	リクエストに対して HTML 形式のエラー・メッセージが必要とされる場合、`true` を戻します。
setRequestObject()（9-19 ページ）	リクエスト・スコープ・オブジェクトを設定します。
getRequestObject()（9-19 ページ）	リクエスト・スコープ・オブジェクトを取得します。

createNestedRequest()

説明

ネストしたリクエストのインスタンスを戻します。

構文

```
public XSQLPageRequest createNestedRequest(
                  java.net.URL pageurl,
                  java.util.Dictionary params);
```

パラメータ	説明
pageurl	ネストしたリクエストの URL。
params	追加パラメータのオプションのディクショナリ。

getConnectionName()

説明

リクエストに使用されている接続の名前を戻します。接続が設定または使用されてない場合、null になる場合があります。

構文

```
public java.lang.String getConnectionName();
```

getErrorWriter()

説明

リクエストを処理するエラーを出力するためのプリント・ライターを戻します。

構文

```
public java.io.PrintWriter getErrorWriter();
```

getJDBCConnection()

説明

リクエストに対して JDBC 接続を使用中にします。（null になる可能性があります。）

構文

```
public java.sql.Connection getJDBCConnection();
```

getPageEncoding()

説明

リクエストに対応するソース XSQL ページのエンコーディングを戻します。

構文

```
public java.lang.String getPageEncoding();
```

getParameter()

説明
リクエストされたパラメータ値を戻します。

構文
```
public String getParameter( String name);
```

パラメータ	説明
name	パラメータ名。

getPostedDocument()

説明
リクエストに対してポストされた XML の内容を、XML 文書として戻します。

構文
```
public oracle.xml.xsql.Document getPostedDocument();
```

getRequestParamsAsXMLDocument()

説明
リクエスト・パラメータの内容を、XML 文書として戻します。

構文
```
public oracle.xml.xsql.Document getRequestParamsAsXMLDocument();
```

getRequestType()

説明
作成しているページ・リクエストのタイプを識別する文字列を戻します。

構文
```
public String getRequestType();
```

getSourceDocumentURI()

説明
リクエストされたドキュメントの URI の文字列表現を戻します。

構文
```
public String getSourceDocumentURI();
```

getStylesheetParameter()

説明
スタイルシート・パラメータを名前で取得します。

構文
```
public String getStylesheetParameter( String name);
```

パラメータ	説明
name	スタイルシートのパラメータ名。

getStylesheetParameter()

説明
スタイルシート・パラメータの名前の列挙を取得します。

構文
```
public java.util.Enumeration getStylesheetParameters();
```

getStylesheetURI()

説明
結果処理に使用されるスタイルシートの URI を戻します。

構文
```
public java.lang.String getStylesheetURI();
```

getUserAgent()

説明

リクエストしているプログラムの文字列識別子を戻します。

構文

```
public java.lang.String getUserAgent();
```

getWriter()

説明

ページ・リクエストの結果の出力に使用されるプリント・ライターを戻します。

構文

```
public java.io.PrintWriter getWriter();
```

getXSQLConnection()

説明

リクエストに使用されている XSQLConnection オブジェクトを取得します。null になる場合があります。

構文

```
public oracle.xml.xsql.XSQLConnection getXSQLConnection();
```

isIncludedRequest()

説明

リクエストが別のリクエストに含まれている場合、true を戻します。

構文

```
public boolean isIncludedRequest();
```

isOracleDriver()

説明
現行の接続が Oracle JDBC ドライバを使用している場合、true を戻します。

構文
```
public boolean isOracleDriver();
```

printedErrorHeader()

説明
Error Header が出力されたかどうかを戻します。

構文
```
public boolean printedErrorHeader();
```

requestProcessed()

説明
ページ・リクエストでリクエスト末尾処理を実行できるようにします。

構文
```
public void requestProcessed();
```

setConnectionName()

説明
リクエストに使用する接続名を設定します。

構文
```
public void setConnectionName( String connName);
```

パラメータ	説明
connName	接続名。

setContentType()

説明
結果ページの Content Type を設定します。

構文
```
public void setContentType( String mimetype);
```

パラメータ	説明
mimetype	結果ページのコンテンツを示す MIME タイプ。

setIncludingRequest()

説明
リクエストに対して Including Page Request オブジェクトを設定します。

構文
```
public void setIncludingRequest( XMLPageRequest includingEnv);
```

パラメータ	説明
includingEnv	挿入側のページの XSQLPageRequest コンテキスト。

setPageEncoding()

説明
リクエストに対応するソース XSQL ページのエンコーディングを設定します。

構文
```
public void setPageEncoding( String enc);
```

パラメータ	説明
enc	現行のページのエンコーディング。

setPageParam()

説明
動的ページ・パラメータ値を設定します。

構文
```
public void setPageParam( String name,
                          String value);
```

パラメータ	説明
name	ページのプライベート・パラメータの名前。
value	ページのプライベート・パラメータの値。

setPostedDocument()

説明
ポストされるドキュメントのプログラムを設定できるようにします。

構文
```
public void setPostedDocument( org.w3c.dom.Document doc);
```

パラメータ	説明
doc	リクエストの一部としてポストされる XML 文書。

setPrintedErrorHeader()

説明
Error Header が出力されるかどうかを設定します。

構文
```
public void setPrintedErrorHeader( boolean yes);
```

パラメータ	説明
yes	Error Header が出力されるかどうかを設定します。

setStylesheetParameter()

説明

対応するスタイルシートへ渡されるパラメータ値を設定します。

構文

```
public void setStylesheetParameter( java.lang.String name,
                                    java.lang.String value);
```

パラメータ	説明
name	スタイルシート・パラメータの名前。
value	スタイルシート・パラメータの値。

setStylesheetURI()

説明

結果処理に使用されるスタイルシートの URI を設定します。

構文

```
public void setStylesheetURI( String uri);
```

パラメータ	説明
uri	リクエストの変換に使用されるスタイルシートの URI。

translateURL()

説明

リクエストのベース URI に対して相対 URI に変換された絶対 URL を表す文字列を戻します。

構文

```
public String translateURL( String url);
```

パラメータ	説明
url	リクエストの変換に使用されるスタイルシートの URL。

useConnectionPooling()

説明

リクエストに対して接続プーリングが必要とされる場合、`true` を戻します。

構文

```
public boolean useConnectionPooling();
```

useHTMLErrors()

説明

リクエストに対して HTML 形式のエラー・メッセージが必要とされる場合、`true` を戻します。

構文

```
public boolean useHTMLErrors();
```

setRequestObject()

説明

リクエスト・スコープ・オブジェクトを設定します。

構文

```
void setRequestObject( String name,
                       Object obj);
```

パラメータ	説明
name	設定するリクエスト・スコープ・オブジェクトの名前。
obj	リクエスト・スコープ・オブジェクト。

getRequestObject()

説明

リクエスト・スコープ・オブジェクトを取得します。

構文

```
Object getRequestObject( String name);
```

パラメータ	説明
name	取得するリクエスト・スコープ・オブジェクトの名前。

XSQLParserHelper クラス

XSQLParserHelper の説明

このクラスは、一般的な XML 解析ルーチンです。

XSQLParserHelper の構文

```
public final class XSQLParserHelper extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.xsql.XSQLParserHelper
```

XSQLParserHelper のメソッド

表 9-5 XSQLParserHelper のメソッドの概要

メソッド	説明
XSQLParserHelper() (9-20 ページ)	クラス・コンストラクタです。
newDocument() (9-21 ページ)	新しい空の XML 文書を戻します。
parse() (9-21 ページ)	様々なソースから XML 文書を解析します。
parseFromString() (9-22 ページ)	文字列から XML 文書を解析します。
print() (9-22 ページ)	XML 文書を出力します。

XSQLParserHelper()

説明

クラス・コンストラクタです。

構文

```
public  XSQLParserHelper();
```

newDocument()

説明

新しい空の XML 文書を戻します。

構文

```
public static oracle.xml.xsql.Document newDocument();
```

parse()

説明

様々なソースから XML 文書を解析します。次の表に、オプションを示します。

構文	説明
public static oracle.xml.xsql.Document parse(InputStream is, URL baseUrl, PrintWriter errorWriter);	InputStream オブジェクトから XML 文書を解析します。
public static oracle.xml.xsql.Document parse(Reader r, PrintWriter errorWriter);	Reader オブジェクトから XML 文書を解析します。
public static oracle.xml.xsql.Document parse(URL url, PrintWriter errorWriter);	URL から XML 文書を解析します。

パラメータ	説明
is	解析する XML を含む入力ストリーム。
baseUrl	XML 文書のベース URL。
errorWriter	エラー・メッセージを受信するプリント・ライター。
r	解析する XML を含むリーダー。
url	解析する XML 文書の URL。

parseFromString()

説明

文字列から XML 文書を解析します。次の表に、オプションを示します。

構文	説明
public static oracle.xml.xsql.Document parseFromString(StringBuffer xmlString, PrintWriter errorWriter);	文字列バッファから XML 文書の解析を行います。
public static oracle.xml.xsql.Document parseFromString(String xmlString, PrintWriter errorWriter);	文字列から XML 文書の解析を行います。

パラメータ	説明
xmlString	解析する XML を含む文字列。
errorWriter	エラー・メッセージを受信するプリント・ライター。

print()

説明

XML 文書を出力します。

構文

```
public static void print(  org.w3c.docm.Document doc,
                           java.io.PrintWriter out);
```

パラメータ	説明
doc	出力する XML 文書。
out	ドキュメントのシリアル化に使用するプリント・ライター。

XSQLRequest クラス

XSQLRequest の説明

このクラスは、XSQL ページに対するリクエストをプログラム処理します。

XSQLRequest の構文

```
public class XSQLRequest extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.xsql.XSQLRequest
```

XSQLRequest のメソッド

表 9-6 XSQLRequest のメソッドの概要

メソッド	説明
XSQLRequest()（9-23 ページ）	クラス・コンストラクタです。XSQL ページに対するリクエストを作成します。
process()（9-24 ページ）	出力またはエラーを書き込んでリクエストを処理します。
processToXML()（9-25 ページ）	出力またはエラーを書き込んでリクエストを処理します。
setPostedDocument()（9-25 ページ）	リクエストの一部としてポストされる場合と同じ方法で処理されるように、XML 文書のプログラム設定を行います。

XSQLRequest()

説明

様々なソースから XSQLRequest のインスタンスを作成します。

構文

```
public  XSQLRequest( java.lang.String url);
```

パラメータ	説明
url	処理するソース XSQL ページの URL。

process()

説明

出力またはエラーを書き込んでリクエストを処理します。次の表に、オプションを示します。

構文	説明
public void process();	System.out または System.err に出力またはエラーを書き込んでリクエストを処理します。
public void process(Dictionary params);	System.out または System.err に出力またはエラーを書き込んでリクエストを処理します。
public void process(Dictionary params, PrintWriter out, PrintWriter err);	対応するプリント・ライターに出力またはエラーを書き込んでリクエストを処理します。
public void process(PrintWriter out, PrintWriter err);	対応するプリント・ライターに出力またはエラーを書き込んでリクエストを処理します。

パラメータ	説明
params	XSQL ページ・パラメータを持つディクショナリ。
out	結果ページの結果の書込みに使用するプリント・ライター。
err	結果ページのエラーの書込みに使用するプリント・ライター。

processToXML()

説明

出力またはエラーを書き込んでリクエストを処理します。次の表に、オプションを示します。

構文	説明
public org.w3c.dom.Document processToXML()	System.out または System.err に出力またはエラーを書き込んでリクエストを処理します。
public org.w3c.dom.Document processToXML(Dictionary params)	System.out または System.err に出力またはエラーを書き込んでリクエストを処理します。
public org.w3c.dom.Document processToXML(Dictionary params, PrintWriter err)	対応するプリント・ライターに出力またはエラーを書き込んでリクエストを処理します。
public org.w3c.dom.Document processToXML(PrintWriter err);	対応するプリント・ライターにエラーを書き込んでリクエストを処理します。

パラメータ	説明
params	XSQL ページ・パラメータを持つディクショナリ。
err	結果ページのエラーの書込みに使用するプリント・ライター。

setPostedDocument()

説明

リクエストの一部としてポストされる場合と同じ方法で処理されるように、XML 文書のプログラム設定を行います。

構文

```
public void setPostedDocument( org.w3c.dom.Document postedDoc)
```

パラメータ	説明
postedDoc	DOM 文書。

XSQLRequestObjectListener インタフェース

XSQLRequestObjectListener の説明

このインタフェースを実装したオブジェクトは、現行のページ・リクエスト処理の完了の通知を受け取ることができます。このインタフェースを実装し、`XSQLPageRequest::setRequestObject()` を使用して現行のリクエスト・コンテキストに追加されるオブジェクトは、最も外側のページの処理完了時に通知を受け取ります。

XSQLRequestObjectListener のメソッド

pageProcessingCompleted()

説明

ページ処理の完了時にリクエスト・スコープ・オブジェクトに通知し、割り当てられたすべてのリソースのクリーンアップを可能にするコールバック・メソッドです。この方法によって、`XSQLRequestObjectListener` インタフェースを実装するすべてのリクエスト・スコープ・オブジェクトの通知を受け取ることができます。

構文

```
void pageProcessingCompleted();
```

XSQLServletPageRequest クラス

XSQLServletPageRequest の構文

```
public final class XSQLServletPageRequest extends XSQLPageRequestImpl

java.lang.Object
  |
  +--XSQLPageRequestImpl
        |
        +--oracle.xml.xsql.XSQLServletPageRequest
```

XSQLServletPageRequest の実装済インタフェース

XSQLPageRequest インタフェース

XSQLServletPageRequest のメソッド

表 9-7 XSQLServletPageRequest のメソッドの概要

メソッド	説明
XSQLServletPageRequest() (9-28 ページ)	サーブレット固有の XSQL ページ・リクエストを作成します。
createNestedRequest() (9-28 ページ)	ネストしたリクエストのインスタンスを戻します。
getCookie() (9-29 ページ)	HTTP リクエスト Cookie の値を取得します。
getHttpServletRequest() (9-29 ページ)	XSQL ページ・リクエストを開始した HttpServletRequest を取得します。
getHttpServletResponse() (9-29 ページ)	XSQL ページ・リクエストに対応する HttpServletResponse を取得します。
getParameter() (9-30 ページ)	HTTP パラメータをパラメータのソースとして使用します。
getPostedDocument() (9-30 ページ)	リクエストに対してポストされた XML 文書を取得します。
getRequestParamsAsXMLDocument() (9-30 ページ)	リクエスト・パラメータを XML 文書として取得します。
getRequestType() (9-30 ページ)	リクエストのタイプを戻します。
getUserAgent() (9-31 ページ)	ブラウザから設定されたユーザー・エージェント文字列を戻します。

表 9-7 XSQLServletPageRequest のメソッドの概要（続き）

メソッド	説明
setContentType()（9-31 ページ）	結果ページの Content Type を設定します。
setPageEncoding()（9-31 ページ）	ページのエンコーディングを設定します。
translateURL()（9-32 ページ）	リクエストのベース URI に対して相対 URI に変換された絶対 URL を表す文字列を戻します。
useHTMLErrors()（9-32 ページ）	HTML 形式のエラーを使用する必要があるかどうかを設定します。

XSQLServletPageRequest()

説明

サーブレット固有の XSQL ページ・リクエストを作成します。

構文

```
public XSQLServletPageRequest(
                    oracle.xml.xsql.HttpServletRequest req,
                    oracle.xml.xsql.HttpServletResponse resp);
```

パラメータ	説明
req	リクエスト。
resp	レスポンス。

createNestedRequest()

説明

ネストしたリクエストのインスタンスを戻します。

構文

```
public XSQLPageRequest createNestedRequest(
                    java.net.URL pageurl,
                    java.util.Dictionary params)
```

パラメータ	説明
pageurl	ページの URL。
params	リクエストのパラメータ。

getCookie()

説明

HTTP リクエスト Cookie の値を取得します。

構文

```
public java.lang.String getCookie(  String name);
```

パラメータ	説明
name	取得する Cookie の名前。

getHttpServletRequest()

説明

XSQL ページ・リクエストを開始した HttpServletRequest を取得します。

構文

```
public oracle.xml.xsql.HttpServletRequest getHttpServletRequest();
```

getHttpServletResponse()

説明

XSQL ページ・リクエストに対応する HttpServletResponse を取得します。

構文

```
public oracle.xml.xsql.HttpServletResponse getHttpServletResponse();
```

getParameter()

説明

HTTP パラメータをパラメータのソースとして使用します。

構文

```
public java.lang.String getParameter( String name);
```

パラメータ	説明
name	値を取得するパラメータの名前。

getPostedDocument()

説明

リクエストに対してポストされた XML 文書を取得します。存在しない場合は、null を戻します。

構文

```
public oracle.xml.xsql.Document getPostedDocument();
```

getRequestParamsAsXMLDocument()

説明

リクエスト・パラメータを XML 文書として取得します。

構文

```
public oracle.xml.xsql.Document getRequestParamsAsXMLDocument();
```

getRequestType()

説明

リクエストのタイプ（「Servlet」、「Programmatic」、「Command Line」または「Custorm」）を戻します。

構文

```
public java.lang.String getRequestType();
```

getUserAgent()

説明
ブラウザから設定されたユーザー・エージェント文字列を戻します。

構文
```
public java.lang.String getUserAgent();
```

setContentType()

説明
結果ページの Content Type を設定します。

構文
```
public void setContentType( String mimetype);
```

パラメータ	説明
mimetype	結果ページのコンテンツを示す MIME タイプ。

setPageEncoding()

説明
ページのエンコーディングを設定します。

構文
```
public void setPageEncoding( String enc);
```

パラメータ	説明
enc	ページのエンコーディング。

translateURL()

説明

リクエストのベース URI に対して相対 URI に変換された絶対 URL を表す文字列を戻します。

構文

```
public java.lang.String translateURL( String path);
```

パラメータ	説明
path	絶対 URL に変換される相対 URL。

useHTMLErrors()

説明

HTML 形式のエラーを使用する必要があるかどうかを設定します。

構文

```
public boolean useHTMLErrors();
```

XSQLStylesheetProcessor クラス

XSQLStylesheetProcessor の説明

このクラスは、XSLT スタイルシート処理エンジンです。

XSQLStylesheetProcessor の構文

```
public final class XSQLStylesheetProcessor extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.xsql.XSQLStylesheetProcessor
```

XSQLStylesheetProcessor のメソッド

表 9-8 XSQLStylesheetProcessor のメソッドの概要

メソッド	説明
XSQLStylesheetProcessor() (9-33 ページ)	XSLT スタイルシート変換を処理します。
processToDocument() (9-34 ページ)	XSLT スタイルシートによって XML 文書を変換し、結果を XML 文書として戻します。
processToWriter() (9-34 ページ)	XSLT スタイルシートによって XML 文書を変換し、結果をプリント・ライターに書き込みます。
getServletContext() (9-35 ページ)	HTTP Servlet のコンテキストを取得します。

XSQLStylesheetProcessor()

説明

XSLT スタイルシート変換を処理します。

構文

```
public  XSQLStylesheetProcessor();
```

processToDocument()

説明

XSLT スタイルシートによって XML 文書を変換し、結果を XML 文書として戻します。

構文

```
public static oracle.xml.xsql.Document processToDocument(
                    org.w3c.dom.Document xml,
                    String xslURI,
                    XSQLPageResuest env);
```

パラメータ	説明
xml	XML 文書。
xslURI	XSL スタイルシートの URI。
env	XSQLPageRequest コンテキスト。

processToWriter()

説明

XSLT スタイルシートによって CML 文書を変換し、結果をプリント・ライターに書き込みます。

構文

```
public static void processToWriter(
                    oracle.xml.xsql.Document xml,
                    String xslURI,
                    XSQLPageResuest env);
```

パラメータ	説明
xml	XML 文書。
xslURI	XSL スタイルシートの URI。
env	XSQLPageRequest コンテキスト。

getServletContext()

説明

HTTP Servlet のコンテキストを取得します。

構文

```
javax.servlet.ServletContext getServletContext();
```

XSQLConnectionManager インタフェース

XSQLConnectionManager の説明

組込み Connection Manager の実装をオーバーライドするために実装する必要がある 2 つのインタフェースのうちの 1 つです。XSQL ページ・プロセッサは、各リクエストに対応する `XSQLConnectionManagerFactory` に対し、XSQLConnectionManager のインスタンスを作成して現行のリクエストを処理するようにリクエストします。

マルチスレッド環境では、XSQLConnectionManager を実装することによって、`getConnection()` によって戻された XSQLConnection インスタンスが、`releaseConnection()` のコール後に XSQL ページ・プロセッサによって解放されるまで、他のスレッドによって使用されないようにする必要があります。

XSQLConnectionManager の構文

```
public interface XSQLConnectionManager;
```

XSQLConnectionManager のメソッド

表 9-9 XSQLConnectionManager のメソッドの概要

メソッド	説明
getConnection()（9-36 ページ）	Connection Manager から接続を取得します。
releaseConnection()（9-37 ページ）	接続を解放します。

getConnection()

説明

Connection Manager から接続を取得します。

構文

```
XSQLConnection getConnection( String connName,
                              XSQLPageRequest env);
```

パラメータ	説明
connName	接続名。
env	XSQLPageRequest コンテキスト。

releaseConnection()

説明

接続をリリースします。

構文

```
void releaseConnection( XSQLConnection theConn,
                        XSQLPageRequest env);
```

パラメータ	説明
theConn	リリースされる XSQLConnection オブジェクト。
env	XSQLPageRequest コンテキスト。

XSQLConnectionManagerFactory インタフェース

XSQLConnectionManagerFactory の説明

組込み Connection Manager の実装をオーバーライドするために実装する必要がある 2 つのインタフェースのうちの 1 つです。XSQL ページ・プロセッサは、各リクエストに対応する XSQLConnectionManagerFactory に対し、XSQLConnectionManager のインスタンスを作成して現行のリクエストを処理するようにリクエストします。

XSQLConnectionManagerFactory の構文

```
public interface XSQLConnectionManagerFactory
```

XSQLConnectionManagerFactory のメソッド

create()

説明

XSQLConnectionManager のインスタンスを戻します。実装することによって、新しい XSQLConnectionManager であるか、または共有される XSQLConnectionManager であるかを判別できます。

構文

```
XSQLConnectionManager create();
```

XSQLDocumentSerializer インタフェース

XSQLDocumentSerializer の説明

このインタフェースは、すべての XSQL シリアライザによって実装される必要があるインタフェースです。XSQL シリアライザは、XSQL データのページを XML 文書としてシリアル化してプリント・ライターに送信します。

処理命令 `<?xml-stylesheet?>` で serializer="XXX" 擬似属性が検出された場合、XSQL ページ・プロセッサは、次の手順で、対応するシリアライザを起動します。

- 引数なしのコンストラクタを使用して、シリアライザのインスタンスを作成します。
- XSQL ドキュメント・シリアライザの `serialize()` メソッドをコールします。

注意 : XSQLDocumentSerializer を実装する場合は、次の操作を実行する必要があります。

- まず、`env.setContentType()` をコールして内容の型を設定します。
- 次に、`env.getWriter()` をコールして書込み先のライターを取得します。

シリアライザで処理不可能な例外が発生した場合は、XSQL ページ・プロセッサがスタックトレースをフォーマットします。

oracle.xml.xsql.src.serializers.XSQLSampleSerializer の例を参照してください。

XSQLDocumentSerializer の構文

```
public interface XSQLDocumentSerializer
```

XSQLDocumentSerialize のメソッド

serialize()

説明

結果の XML 文書をシリアル化して出力されるためのライターに送信します。

構文

```
void serialize( org.w3c.dom.Document doc,
                XSQLPageRequest env);
```

パラメータ	説明
doc	シリアル化する XML 文書。
env	XSQLPageRequest コンテキスト。

10

Oracle XML JavaBeans

Oracle XML JavaBeans は、次のパッケージと同義です。

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

oracle.xml.async パッケージ

oracle.xml.async の説明

このパッケージは不可視 Bean です。これによって、個別のスレッドで非同期の DOM 解析がバックグラウンドで実行可能になります。このパッケージは、EventHandler インタフェースを使用して、ジョブの完了をコール側クラスに通知します。表 10-1 に、oracle.xml.async のクラスの概要を示します。

表 10-1 oracle.xml.async のクラスの概要

クラス	説明
DOMBuilder クラス（10-3 ページ）	eXtensible Markup Language（XML）1.0 のパーサーをカプセル化して、XML 文書を解析し、DOM ツリーを構築します。
DOMBuilderBeanInfo クラス（10-15 ページ）	DOMBuilder Bean に関する情報を提供します。
DOMBuilderErrorEvent クラス（10-17 ページ）	解析例外の発生時に送信されるエラー・イベントを定義します。
DOMBuilderErrorListener インタフェース（10-19 ページ）	解析中にエラーが検出された場合に通知を受け取るためには、このインタフェースを実装する必要があります。
DOMBuilderEvent クラス（10-20 ページ）	DOMBuilder が、すべての登録リスナーに解析イベントについて通知するために使用するイベント・オブジェクトです。
DOMBuilderListener インタフェース（10-22 ページ）	非同期解析中のイベントに関する通知を受け取るためには、このインタフェースを実装する必要があります。
ResourceManager クラス（10-24 ページ）	固定数の論理リソースへのアクセスを維持する単純セマフォを実装します。
XSLTransformer クラス（10-27 ページ）	バックグラウンド・スレッドで XSL 変換を適用します。
XSLTransformerBeanInfo クラス（10-34 ページ）	XSLTransformer Bean に関する情報を提供します。
XSLTransformerErrorEvent クラス（10-36 ページ）	XSLTransformer が、すべての登録リスナーに変換エラー・イベントについて通知するために使用するエラー・イベント・オブジェクトです。

DOMBuilder クラス

DOMBuilder の説明

このクラスは、eXtensible Markup Language（XML）1.0 のパーサーをカプセル化して、XML 文書を解析し、DOM ツリーを構築します。解析は別々のスレッドで行われ、ツリーが構築されたときの通知には DOMBuilderListener インタフェースが使用される必要があります。

DOMBuilder の構文

```
public class DOMBuilder extends java.lang.Object implements java.io.Serializable,
oracle.xml.async.DOMBuilderConstants, java.lang.Runnable

java.lang.Object
  |
  +--oracle.xml.async.DOMBuilder
```

DOMBuilder の実装済インタフェース

- `oracle.xml.async.DOMBuilderConstants`
- `java.lang.Runnable`
- `java.io.Serializable`

DOMBuilder のフィールド

表 10-2 DOMBuilder のフィールド

フィールド	構文	説明
inSource	protected org.xml.sax.InputSource inSource	解析する XML データを含む入力ソース。
url	protected java.net.URL url	XML データの解析元の URL。
inStream	protected java.io.InputStream inStream	解析する XML データを含む入力ストリーム。
inString	protected java.lang.String inString	XML データの解析元の URL を含む文字列。
methodToCall	protected int methodToCall	入力タイプに基づいてコールする XML パーサー・メソッド。

表 10-2 DOMBuilder のフィールド（続き）

フィールド	構文	説明
reader	protected java.io.Reader reader	解析する XML データを含む java.io.Reader。
result	protected oracle.xml.async.XMLDocument result	解析中の XML 文書。
rootName	protected java.lang.String rootName	ルートとして処理する XML 要素の名前。

DOMBuilder のメソッド

表 10-3 DOMBuilder のメソッドの概要

メソッド	説明
DOMBuilder()（10-5 ページ）	新しいパーサー・オブジェクトを作成します。
addDOMBuilderErrorListener()（10-6 ページ）	DOMBuilderErrorListener を追加します。
addDOMBuilderListener()（10-6 ページ）	DOMBuilderListener を追加します。
getDoctype()（10-6 ページ）	DTD を取得します。
getDocument()（10-7 ページ）	解析用の文書を取得します。
getId()（10-7 ページ）	パーサーのオブジェクト ID を戻します。
getReleaseVersion()（10-7 ページ）	Oracle XML Parser のバージョン番号を戻します。
getResult()（10-7 ページ）	解析中の文書を取得します。
getValidationMode()（10-8 ページ）	検証モードを戻します。
parse()（10-8 ページ）	指定された入力から XML を解析します。
parseDTD()（10-9 ページ）	XML 外部 DTD を解析します。
removeDOMBuilderErrorListener()（10-10 ページ）	DOMBuilderErrorListener を削除します。
removeDOMBuilderListener()（10-10 ページ）	DOMBuilderListener を削除します。
run()（10-11 ページ）	スレッドで実行します。

表 10-3 DOMBuilder のメソッドの概要 （続き）

メソッド	説明
setBaseURL()（10-11 ページ）	外部エンティティおよび DTD をロードするためのベース URL を設定します。
setDebugMode()（10-11 ページ）	ドキュメントのデバッグ情報に対して実行するフラグを設定します。
setDoctype()（10-12 ページ）	DTD を設定します。
setErrorStream()（10-12 ページ）	エラーおよび警告用の出力ストリームを設定します。
setNodeFactory()（10-13 ページ）	ノード・ファクトリを設定します。
setPreserveWhitespace()（10-13 ページ）	空白保持モードを設定します。
setValidationMode()（10-14 ページ）	検証モードを設定します。
showWarnings()（10-14 ページ）	警告を出力するかどうかを判別します。

DOMBuilder()

説明

新しいパーサー・オブジェクトを作成します。次の表に、オプションを示します。

構文	説明
public DOMBuilder();	新しいパーサー・オブジェクトを作成します。
public DOMBuilder(int id);	指定された ID を持つ新しいパーサー・オブジェクトを作成します。

パラメータ	説明
id	DOMBuilder の ID。

addDOMBuilderErrorListener()

説明

DOMBuilderErrorListener を追加します。

構文

```
public void addDOMBuilderErrorListener( DOMBuilderErrorListener p0);
```

パラメータ	説明
p0	追加する DOMBuilderErrorListener。

addDOMBuilderListener()

説明

DOMBuilderListener を追加します。

構文

```
public void addDOMBuilderListener(DOMBuilderListener p0);
```

パラメータ	説明
p0	追加する DOMBuilderListener。

getDoctype()

説明

DTD を取得します。

構文

```
public synchronized oracle.xml.async.DTD getDoctype();
```

getDocument()

説明
解析用の文書を取得します。

構文
```
public synchronized oracle.xml.async.XMLDocument getDocument();
```

getId()

説明
パーサーのオブジェクト ID（DOMBuilder）を戻します。

構文
```
public int getId();
```

getReleaseVersion()

説明
Oracle XML Parser のバージョン番号を文字列として戻します。

構文
```
public synchronized java.lang.String getReleaseVersion();
```

getResult()

説明
解析中の文書を取得します。

構文
```
public synchronized org.w3c.dom.Document getResult();
```

getValidationMode()

説明

検証モードを戻します。XML Parser が検証を実行する場合は `true`、実行しない場合は `false` を戻します。

構文

```
public synchronized boolean getValidationMode()
```

parse()

説明

指定された入力から XML を解析します。次の例外が発生します。

- `XMLParseException`（構文エラーまたはその他のエラー）
- `SAXException`（すべての SAX の例外。別の例外が隠されている可能性があります。）
- `IOExceptiopn`（I/O エラー）

次の表に、オプションを示します。

構文	説明
public final synchronized void parse (InputSource in);	入力ソースから XML を解析します。
public final synchronized void parse (InputStream in);	入力ソースから XML を解析します。外部エンティティおよび DTD を解決するためのベース URL を設定する必要があります。
public final synchronized void parse (Reader r);	リーダーから XML を解析します。外部エンティティおよび DTD を解決するためのベース URL を設定する必要があります。
public final synchronized void parse (String urlName);	引数で指定された URL から XML を解析します。
public final synchronized void parse (URL url);	指定された URL が指す XML 文書を解析し、それに対応する XML 文書の階層を作成します。

パラメータ	説明
in	解析する入力。
r	解析する XML データを含むリーダー。
urlName	解析元の URL を含む文字列。
url	解析する XML 文書を指す URL。

parseDTD()

説明

XML 外部 DTD を解析します。次の例外が発生します。

- `XMLParseException`（構文エラーまたはその他のエラー）
- `SAXException`（すべての SAX の例外。別の例外が隠されている可能性があります。）
- `IOExceptiopn`（I/O エラー）

次の表に、オプションを示します。

構文	説明
public final synchronized void parseDTD (InputSource in, String rootName);	指定された入力ソースから解析を実行します。
public final synchronized void parseDTD (InputStream in, String rootName);	指定された入力ストリームから解析を実行します。外部エンティティおよび DTD を解決するためのベース URL を設定する必要があります。
public final synchronized void parseDTD (Reader r, String rootName);	指定されたリーダーから解析を実行します。外部エンティティおよび DTD を解決するためのベース URL を設定する必要があります。
public final synchronized void parseDTD (String urlName, String rootName);	指定された URL から解析を実行します。
public final synchronized void parseDTD (URL url, String rootName);	指定された URL が指す XML 外部 DTD を解析し、対応する XML 文書の階層を作成します。

パラメータ	説明
in	解析する入力。
rootName	ルート要素として使用する要素。
r	解析する XML データを含むリーダー。
urlName	解析元の URL を含む文字列。
utl	解析する XML 文書を指す URL。

removeDOMBuilderErrorListener()

説明

DOMBuilderErrorListener を削除します。

構文

```
public synchronized void removeDOMBuilderErrorListener(
                  DOMBuilderErrorListener p0);
```

パラメータ	説明
p0	削除する DOMBuilderErrorListener。

removeDOMBuilderListener()

説明

DOMBuilderListener を削除します。

構文

```
public synchronized void removeDOMBuilderListener(
                  DOMBuilderListener p0);
```

パラメータ	説明
p1	削除する DOMBuilderListener。

run()

説明

スレッドで実行します。java.lang.Runnable インタフェースの java.lang.Runnable.run() で指定されます。

構文

```
public void run();
```

setBaseURL()

説明

外部エンティティおよび DTD をロードするためのベース URL を設定します。このメソッドは、XML 文書を解析する際に、parse(InputStream) オプションを使用する場合にコールする必要があります。

構文

```
public synchronized void setBaseURL( java.net.URL url);
```

パラメータ	説明
url	ベース URL。

setDebugMode()

説明

ドキュメントのデバッグ情報に対して実行するフラグを設定します。

構文

```
public void setDebugMode(boolean flag);
```

パラメータ	説明
flag	デバッグ情報が格納されているかどうかを判別します。格納されている場合は `true`、格納されていない場合は `false` を指定します。

setDoctype()

説明

DTD を設定します。

構文

```
public synchronized void setDoctype(oracle.xml.async.DTD dtd)
```

パラメータ	説明
dtd	解析中に設定および使用する DTD。

setErrorStream()

説明

エラーおよび警告を出力するための出力ストリームを設定します。エラー用の出力ストリームが指定されない場合、パーサーは、標準エラー出力ストリーム System.err を使用して、エラーおよび警告を出力します。次の表に、オプションを示します。

構文	説明
public final synchronized void setErrorStream(OutputStream out);	出力ストリームを使用します。
public final synchronized void setErrorStream(OutputStream out, String enc);	出力ストリームを使用します。指定されたエンコーディングがサポートされない場合、`IOException` が発生します。
public final synchronized void setErrorStream(PrintWriter out);	プリント・ライターを使用します。

パラメータ	説明
out	エラーおよび警告の出力先。
enc	使用するエンコーディング。

setNodeFactory()

説明

ノード・ファクトリを設定します。アプリケーションは、このメソッドを介して NodeFactory を拡張して登録できます。パーサーは、ユーザーが提供する NodeFactory を使用して、DOM ツリーのノードを作成します。次の例外が発生します。

- `XMLParseException`（無効なファクトリが設定された場合）

構文

```
public synchronized void setNodeFactory(
                  oracle.xml.async.NodeFactory factory);
```

パラメータ	説明
factory	設定する NodeFactory。

setPreserveWhitespace()

説明

空白保持モードを設定します。

構文

```
public synchronized void setPreserveWhitespace( boolean flag);
```

パラメータ	説明
flag	保持モード。空白を保持する場合は `true`、保持しない場合は `false` を指定します。

setValidationMode()

説明

検証モードを設定します。

構文

```
public synchronized void setValidationMode(boolean yes);
```

パラメータ	説明
yes	XML Parser が検証を実行するかどうかを判別します。検証を実行する場合は `true`、実行しない場合は `false` を指定します。

showWarnings()

説明

警告を出力するかどうかを判別します。

構文

```
public synchronized void showWarnings(boolean yes);
```

パラメータ	説明
yes	警告を出力するかどうかを指定します。警告を出力する場合は `true`、出力しない場合は `false` を指定します。

DOMBuilderBeanInfo クラス

DOMBuilderBeanInfo の説明

このクラスは、DOMBuilder Bean に関する情報を提供します。

DOMBuilderBeanInfo の構文

```
public class DOMBuilderBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.async.DOMBuilderBeanInfo
```

DOMBuilderBeanInfo の実装済インタフェース

```
java.beans.BeanInfo
```

DOMBuilderBeanInfo のメソッド

表 10-4 DOMBuilder のメソッドの概要

メソッド	説明
DOMBuilderBeanInfo()（10-15 ページ）	デフォルトのコンストラクタです。
getIcon()（10-16 ページ）	ツールバーやツールボックスなどで、DOMBuilder Bean を表すために使用可能なイメージ・オブジェクトを取得します。
getPropertyDescriptors()（10-16 ページ）	DOMBuilder Bean の編集可能な PropertyDescriptiors の配列を取得します。

DOMBuilderBeanInfo()

説明

デフォルトのコンストラクタです。

構文

```
public  DOMBuilderBeanInfo();
```

getIcon()

説明

ツールバーやツールボックスなどで、DOMBuilder Bean を表すために使用可能なイメージ・オブジェクトを取得します。リクエストされたアイコンの種類を表すイメージ・オブジェクトを戻します。java.beans.SimpleBeanInfo クラスの getIcon() メソッドをオーバーライドします。

構文

```
public java.awt.Image getIcon( int iconKind);
```

パラメータ	説明
iconKind	リクエストされたアイコンの種類。

getPropertyDescriptors()

説明

DOMBuilder Bean の編集可能な PropertyDescriptiors の配列を取得します。java.beans.SimpleBeanInfo クラスの getPropertyDescriptors() をオーバーライドします。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

DOMBuilderErrorEvent クラス

DOMBuilderErrorEvent の説明

このクラスは、解析例外の発生時に送信されるエラー・イベントを定義します。

DOMBuilderErrorEvent の構文

```
public class DOMBuilderErrorEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--oracle.xml.async.DOMBuilderErrorEvent
```

DOMBuilderErrorEvent の実装済インタフェース

```
java.io.Serializable
```

DOMBuilderErrorEvent のフィールド

表 10-5 DOMBuilderErrorEvent のフィールド

フィールド	構文	説明
e	protected java.lang.Exception	発生中の例外。

DOMBuilderErrorEvent のメソッド

表 10-6 DOMBuilderErrorEvent のメソッドの概要

メソッド	説明
DOMBuilderErrorEvent()（10-18 ページ）	DOMBuilderErrorEvent のコンストラクタです。
getException()（10-18 ページ）	発生中の例外を取得します。
getMessage()（10-18 ページ）	パーサーによって生成されたエラー・メッセージを戻します。

DOMBuilderErrorEvent()

説明

DOMBuilderErrorEvent のコンストラクタです。

構文

```
public  DOMBuilderErrorEvent( Object p0,
                              Exception e);
```

パラメータ	説明
p0	エラー・イベントを作成したオブジェクト。
e	発生中の例外。

getException()

説明

発生中の例外を取得します。

構文

```
public java.lang.Exception getException();
```

getMessage()

説明

パーサーによって生成されたエラー・メッセージを文字列として戻します。

構文

```
public java.lang.String getMessage();
```

DOMBuilderErrorListener インタフェース

DOMBuilderErrorListener の説明

解析中にエラーが検出された場合に通知を受け取るためには、このインタフェースを実装する必要があります。このインタフェースを実装しているクラスは、addDOMBuilderErrorListener メソッドを使用して DOMBuilder に追加する必要があります。

DOMBuilderErrorListener の構文

```
public interface DOMBuilderErrorListener extends java.util.EventListener
```

DOMBuilderErrorListener のメソッド

domBuilderErrorCalled()

説明

このメソッドは、解析エラーが発生した場合にコールされます。

構文

```
public void domBuilderErrorCalled( DOMBuilderErrorEvent p0);
```

パラメータ	説明
p0	解析エラーの結果として DOMBuilder によって生成された DOMBuilderErrorEvent オブジェクト。

DOMBuilderEvent クラス

DOMBuilderEvent の説明

このクラスは、DOMBuilder が、すべての登録リスナーに解析イベントについて通知するために使用するイベント・オブジェクトです。

DOMBuilderEvent の構文

```
public class DOMBuilderEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--oracle.xml.async.DOMBuilderEvent
```

DOMBuilderEvent の実装済インタフェース

```
java.io.Serializable
```

DOMBuilderEvent のフィールド

表 10-7 DOMBuilderEvent のフィールド

フィールド	構文	説明
id	protected int id	ソース DOMBuilder オブジェクトの ID。

DOMBuilderEvent のメソッド

表 10-8 DOMBuilderEvent のメソッドの概要

メソッド	説明
DOMBuilderEvent()（10-21 ページ）	新しい DOMBuilderEvent を作成します。
getID()（10-21 ページ）	イベントに対するソース DOMBuilder の一意の ID を戻します。

DOMBuilderEvent()

説明

新しい DOMBuilderEvent を作成します。

構文

```
public  DOMBuilderEvent( Object p0,
                         int p1);
```

パラメータ	説明
p0	イベントを作成するオブジェクト。
p1	イベントを作成する DOMBuilder の ID。

getID()

説明

イベントに対するソース DOMBuilder の一意の ID を戻します。この ID を使用すると、DOMBuilder の複数のインスタンスがバックグラウンドで動作する場合に、イベントを生成した DOMBuilder のインスタンスを識別できます。

構文

```
public int getID();
```

DOMBuilderListener インタフェース

DOMBuilderListener の説明

非同期解析中のイベントに関する通知を受け取るためには、このインタフェースを実装する必要があります。このインタフェースを実装しているクラスは、addDOMBuilderListener メソッドを使用して DOMBuilder に追加する必要があります。

DOMBuilderListener の構文

```
public interface DOMBuilderListener extends java.util.EventListener
```

DOMBuilderListener のメソッド

表 10-9 DOMBuilderListener のメソッドの概要

メソッド	説明
domBuilderError()（10-19 ページ）	このメソッドは、解析エラーが発生した場合にコールされます。
domBuilderOver()（10-23 ページ）	このメソッドは、解析が完了したときにコールされます。
domBuilderStarted()（10-23 ページ）	このメソッドは、解析が開始したときにコールされます。

domBuilderError()

説明

このメソッドは、解析エラーが発生した場合にコールされます。

構文

```
public void domBuilderError( DOMBuilderEvent p0);
```

パラメータ	説明
p0	DOMBuilder によって生成された DOMBuilderEvent オブジェクト。

domBuilderOver()

説明

このメソッドは、解析が完了したときにコールされます。

構文

```
public void domBuilderOver( DOMBuilderEvent p0);
```

パラメータ	説明
p0	DOMBuilder によって生成された DOMBuilderEvent オブジェクト。

domBuilderStarted()

説明

このメソッドは、解析が開始したときにコールされます。

構文

```
public void domBuilderStarted( DOMBuilderEvent p0);
```

パラメータ	説明
p0	DOMBuilder によって生成された DOMBuilderEvent オブジェクト。

ResourceManager クラス

ResourceManager の説明

このクラスは、セマフォを実装し、固定数の論理リソースへのアクセスを維持します。

ResourceManager の構文

```
public class ResourceManager extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.async.ResourceManager
```

ResourceManager のメソッド

表 10-10 ResourceManager のメソッドの概要

メソッド	説明
ResourceManager()（10-24 ページ）	ResourceManager のコンストラクタです。
activeFound()（10-25 ページ）	管理中のいずれかの論理リソースがアクティブに使用されているかどうかを確認します。
getResource()（10-25 ページ）	使用可能なリソースの数が 0（ゼロ）以外の場合に、リソースの数を 1 つずつ減らします。使用可能なリソースの数が 0（ゼロ）の場合は、リソースが解放されて使用可能になるまで待機します。
releaseResource()（10-25 ページ）	単一のリソースを解放し、使用可能なリソースの数を増やします。
sleep()（10-26 ページ）	try/catch なしで Thread.sleep() を使用できるようにします。

ResourceManager()

説明

ResourceManager のコンストラクタです。

構文

```
public  ResourceManager(int i);
```

パラメータ	説明
i	管理するリソースの数。

activeFound()

説明

管理中のいずれかの論理リソースがアクティブに使用されているかどうかを確認します。1 つ以上のリソースが使用中である場合は `true`、リソースが使用されていない場合は `false` を戻します。

構文

```
public boolean activeFound();
```

getResource()

説明

使用可能なリソースの数が 0（ゼロ）以外の場合に、リソースの数を 1 つずつ減らします。使用可能なリソースの数が 0（ゼロ）の場合は、リソースが解放されて使用可能になるまで待機します。

構文

```
public synchronized void getResource();
```

releaseResource()

説明

リソースを解放します。このメソッドをコールすると、使用可能なリソースの数が 1 つ増えます。

構文

```
public void releaseResource();
```

sleep()

説明

try/catch なしで Thread.sleep() を使用できるようにします。

構文

```
public void sleep(int i);
```

パラメータ	説明
i	管理するリソースの数。

XSLTransformer クラス

XSLTransformer の説明

このクラスは、バックグラウンド・スレッドで XSL 変換を適用します。

XSLTransformer の構文

```
public class XSLTransformer extends java.lang.Object implements
     java.io.Serializable, oracle.xml.async.XSLTransformerConstants,
     java.lang.Runnable

java.lang.Object
  |
  +--oracle.xml.async.XSLTransformer
```

XSLTransformer のフィールド

表 10-11 XSLTransformer のフィールド

フィールド	構文	説明
methodToCall	protected int methodToCall	入力タイプに基づいてコールする XSL 変換メソッド。
result	protected oracle.xml.async.DocumentFragment result	変換結果の文書。

XSLTransformer のメソッド

表 10-12 XSLTransformer のメソッドの概要

メソッド	説明
XSLTransformer()（10-28 ページ）	XSLTransformer クラスのコンストラクタです。
addXSLTransformerErrorListener()（10-29 ページ）	XSLTransformerErrorListener を追加します。
addXSLTransformerListener()（10-29 ページ）	XSLTransformerListener を追加します。
getId()（10-29 ページ）	XSLTransformer の一意の ID を戻します。
getResult()（10-30 ページ）	結果の文書に対して XSL 変換のドキュメント・フラグメントを戻します。

表 10-12 XSLTransformer のメソッドの概要（続き）

メソッド	説明
processXSL()（10-30 ページ）	バックグラウンドで XSL 変換を開始します。制御はすぐに戻ります。
removeDOMTransformerErrorListener()（10-31 ページ）	XSLTransformerErrorListener を削除します。
removeXSLTransformerListener()（10-32 ページ）	XSLTransformerListener を削除します。
run()（10-32 ページ）	XSLTransformation を実行する個別のスレッドを開始します。
setErrorStream()（10-32 ページ）	XSLT プロセッサが使用するエラー・ストリームを設定します。
showWarnings()（10-33 ページ）	XSLT プロセッサが使用する showWarnings フラグを設定します。

XSLTransformer()

説明

XSLTransformer のコンストラクタです。次の表に、オプションを示します。

構文	説明
public XSLTransformer();	XSLTransformer のコンストラクタです。
public XSLTransformer(int id);	識別子を受け入れる XSLTransformer のコンストラクタです。

パラメータ	説明
id	イベントの処理中に XSLTransformer インスタンスの識別に使用可能な一意の整数。

addXSLTransformerErrorListener()

説明

XSLTransformerErrortListener を追加します。

構文

```
public void addXSLTransformerErrorListener(
                  XSLTransformerErrorListerner p0);
```

パラメータ	説明
p0	追加する XSLTransformerErrorListener。

addXSLTransformerListener()

説明

XSLTransformerListener を追加します。

構文

```
public void addXSLTransformerListener( XSLTransformerListerner p0);
```

パラメータ	説明
p0	追加する XSLTransformerListener。

getId()

説明

XSLTransformer の一意の ID を戻します。

構文

```
public int getId();
```

getResult()

説明

結果の文書に対して XSL 変換のドキュメント・フラグメントを戻します。変換完了の通知を受信した後にのみコールされます。変換はバックグラウンドで非同期に実行されるため、processXSL() の直後にこのメソッドをコールすると、結果を取得するまで制御が保留されます。

構文

```
public synchronized oracle.xml.async.DocumentFragment getResult();
```

processXSL()

説明

バックグラウンドで XSL 変換を開始します。制御はすぐに戻ります。XSL 変換中にエラーが発生した場合は、XSLException が発生します。次の表に、オプションを示します。

構文	説明
public void processXSL(oracle.xml.async.XSLStylesheet xsl, InputStream xml, URL ref)	ソース XML 文書は、入力ストリームとして提供されます。
public void processXSL(oracle.xml.async.XSLStylesheet xsl, Reader xml, URL ref);	ソース XML 文書は、リーダーとして提供されます。
public void processXSL(oracle.xml.async.XSLStylesheet xsl, URL xml, URL ref);	ソース XML 文書は、URL を介して提供されます。
public void processXSL(oracle.xml.async.XSLStylesheet xsl, oracle.xml.async.XMLDocument xml);	ソース XML 文書は、DOM ツリーとして提供されます。

構文	説明
public void processXSL(oracle.xml.async.XSLStylesheet xsl, oracle.xml.async.XMLDocument xml, OutputStream os);	ソース XML 文書は DOM ツリーとして提供され、出力は出力ストリームに書き込まれます。

パラメータ	説明
xsl	XSL 変換に使用するスタイルシート。
xml	使用する XML 文書。
ref	入力された XML の外部エンティティを解決するための参照 URL。
os	XSL 変換結果が書き込まれる出力。

removeDOMTransformerErrorListener()

説明

XSLTransformerErrorListener を削除します。

構文

```
public synchronized void removeDOMTransformerErrorListener(
                  XSLTransformerErrorListener p0);
```

パラメータ	説明
p0	削除する XSLTransformerErrorListener。

removeXSLTransformerListener()

説明

XSLTransformerListener を削除します。

構文

```
public synchronized void removeXSLTransformerListener(
                  XSLTransformerListener p0);
```

パラメータ	説明
p0	削除する XSLTransformerListener。

run()

説明

XSLTransformation を実行する個別のスレッドを開始します。java.lang.Runnable インタフェースの java.lang.Runnable.run() で指定されます。

構文

```
public void run();
```

setErrorStream()

説明

XSLT プロセッサが使用するエラー・ストリームを設定します。

構文

```
public final void setErrorStream( java.io.OutputStream out);
```

パラメータ	説明
out	XSLT プロセッサ用のエラー出力ストリーム。

showWarnings()

説明

XSLT プロセッサが使用する showWarnings フラグを設定します。

構文

```
public final void showWarnings( boolean yes);
```

パラメータ	説明
yes	XSLT プロセッサの警告を表示するかどうかを指定します。警告を表示する場合は `true`、表示しない場合は `false` を指定します。

XSLTransformerBeanInfo クラス

XSLTransformerBeanInfo の説明

このクラスは、XSLTransformer Bean に関する情報を提供します。

XSLTransformerBeanInfo の構文

```
public class XSLTransformerBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.async.XSLTransformerBeanInfo
```

XSLTransformerBeanInfo の実装済インタフェース

```
java.beans.BeanInfo
```

XSLTransformerBeanInfo のメソッド

表 10-13 XSLTransformerBeanInfo のメソッドの概要

メソッド	説明
XSLTransformerBeanInfo() (10-34 ページ)	デフォルトのコンストラクタです。
getIcon() (10-35 ページ)	ツールバーやツールボックスなどで、XSLTransformer Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。
getPropertyDescriptors() (10-35 ページ)	XSLTransformer Bean の編集可能な PropertyDescriptior の配列を取得します。

XSLTransformerBeanInfo()

説明

デフォルトのコンストラクタです。

構文

```
public  XSLTransformerBeanInfo();
```

getIcon()

説明

ツールバーやツールボックスなどで、XSLTransformer Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。java.beans.SimpleBeanInfo クラスの getIcon() をオーバーライドします。

構文

```
public java.awt.Image getIcon( int iconKind);
```

パラメータ	説明
iconKind	リクエストされたアイコンの種類。

getPropertyDescriptors()

説明

XSLTransformer Bean の編集可能な PropertyDescriptior の配列を取得します。java.beans.SimpleBeanInfo クラスの getPropertyDescriptors() をオーバーライドします。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

XSLTransformerErrorEvent クラス

XSLTransformerErrorEvent の説明

このクラスは、XSLTransformer が、すべての登録リスナーに変換エラー・イベントについて通知するために使用するエラー・イベント・オブジェクトです。

XSLTransformerErrorEvent の構文

```
public class XSLTransformerErrorEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--oracle.xml.async.XSLTransformerErrorEvent
```

XSLTransformerErrorEvent の実装済インタフェース

```
java.io.Serializable
```

XSLTransformerErrorEvent のフィールド

表 10-14 XSLTransformerErrorEvent のフィールド

フィールド	構文	説明
e	protected java.lang.Exception e	発生中の例外。

XSLTransformerErrorEvent のメソッド

表 10-15 XSLTransformerErrorEvent のメソッドの概要

メソッド	説明
XSLTransformerErrorEvent()（10-37 ページ）	XSLTransformerErrorEvent のコンストラクタです。
getException()（10-37 ページ）	XSLTransformer が一意のオブジェクト ID を検出したという変換例外を戻します。
getMessage()（10-37 ページ）	XSLTransformer が検出したエラーを説明するエラー・メッセージを戻します。

XSLTransformerErrorEvent()

説明

XSLTransformerErrorEvent のコンストラクタです。

構文

```
public  XSLTransformerErrorEvent( Object p0,
                                  Exception e);
```

パラメータ	説明
p0	イベントを作成したオブジェクト。
e	発生した例外。

getException()

説明

XSLTransformer が一意のオブジェクト ID を検出したという変換例外を戻します。

構文

```
public Exception getException();
```

getMessage()

説明

XSLTransformer が検出したエラーを説明するエラー・メッセージを戻します。

構文

```
public String getMessage();
```

XSLTransformerErrorListener インタフェース

XSLTransformerErrorListener インタフェースの説明

非同期変換中のエラー・イベントに関する通知を受け取るためには、このインタフェースを実装する必要があります。このインタフェースを実装しているクラスは、addXSLTransformerListener メソッドを使用して XSLTransformer に追加する必要があります。

XSLTransformerErrorListener インタフェースの構文

```
public interface XSLTransformerErrorListener extends
     java.util.EventListener
```

XSLTransformerErrorListener インタフェースのメソッド

xslTransformerErrorCalled()

説明

このメソッドは、解析または変換エラーが発生した場合にコールされます。

構文

```
public void xslTransformerErrorCalled( XSLTransformerErrorEvent p0);
```

パラメータ	説明
p0	XSLTransformer によって生成された XSLTransformerErrorEvent オブジェクト。

XSLTransformerEvent クラス

XSLTransformerEvent の説明

このクラスは、XSLTransformer が、すべての登録リスナーに XSL 変換イベントを通知するために使用するイベント・オブジェクトです。

XSLTransformerEvent の構文

```
public class XSLTransformerEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--oracle.xml.async.XSLTransformerEvent
```

XSLTransformerEvent の実装済インタフェース

```
java.io.Serializable
```

XSLTransformerEvent のフィールド

表 10-16 XSLTransformerEvent のフィールド

フィールド	構文	説明
id	protected int id	ソース XSLTransformer オブジェクトの ID。

XSLTransformerEvent のメソッド

表 10-17 XSLTransformerEvent のメソッドの概要

メソッド	説明
XSLTransformerEvent()（10-40 ページ）	XSLTransformer ソース・オブジェクトおよびその一意の ID を使用して、XSLTransformerEvent オブジェクトを作成します。
getID()（10-40 ページ）	XSLTransformer オブジェクトの一意の ID を戻します。

XSLTransformerEvent()

構文

XSLTransformer ソース・オブジェクトおよびその一意の ID を使用して、XSLTransformerEvent オブジェクトを作成します。

説明

```
public  XSLTransformerEvent(java.lang.Object p0,
                            int p1);
```

パラメータ	説明
p0	イベントを発生させるソース XSLTransformer オブジェクト。
p1	ソース・オブジェクトを識別する一意の ID。

getID()

構文

XSLTransformer オブジェクトの一意の ID を戻します。この ID を使用すると、XSLTransformer の複数のインスタンスがバックグラウンドで動作する場合に、イベントを生成した XSLTransformer のインスタンスを識別できます。

説明

```
public int getID();
```

XSLTransformerListener インタフェース

XSLTransformerListener の構文

```
public interface XSLTransformerListener extends java.util.EventListener
```

XSLTransformerListener の説明

非同期変換中のイベントの通知を受信するには、このインタフェースが実装されている必要があります。このインタフェースを実装しているクラスは、addXSLTransformerListener メソッドを使用して XSLTransformer に追加する必要があります。

XSLTransformerListener のメソッド

表 10-18 XSLTransformerListener のメソッドの概要

メソッド	説明
xslTransformerError()（10-41 ページ）	解析または変換エラーが発生した場合にコールされます。
xslTransformerOver()（10-42 ページ）	変換が完了したときにコールされます。
xslTransformerStarted()（10-42 ページ）	変換が開始したときにコールされます。

xslTransformerError()

説明

解析または変換エラーが発生した場合にコールされます。

構文

```
public void xslTransformerError( XSLTransformerEvent p0);
```

パラメータ	説明
p0	XSLTransformer によって生成された XSLTransformerEvent オブジェクト。

xslTransformerOver()

説明

変換が完了したときにコールされます。

構文

```
public void xslTransformerOver( XSLTransformerEvent p0);
```

パラメータ	説明
p0	XSLTransformer によって生成された XSLTransformerEvent オブジェクト。

xslTransformerStarted()

説明

変換が開始したときにコールされます。

構文

```
public void xslTransformerStarted( XSLTransformerEvent p0);
```

パラメータ	説明
p0	XSLTransformer によって生成された XSLTransformerEvent オブジェクト。

oracle.xml.dbviewer パッケージ

oracle.xml.dbviewer の説明

oracle.xml.dbviewer パッケージは、HTML、XML または XSL ファイルの表示および編集に使用される可視 Bean です。XML ファイルは、ファイル・システム、データベースおよび Bean バッファの間で転送できます。これを使用して、データベース問合せを XML に変換することもできます。XML ファイルは、XSL を使用して解析および変換できます。表 10-19 に、`oracle.xml.dbviewer` のクラスの概要を示します。

表 10-19 Soracle.xml.dbviewer のクラスの概要

クラス	説明
DBViewer クラス（10-44 ページ）	XSL スタイルシートを適用し、結果の HTML を可視化して、スクロール可能な Swing パネル内にデータベース問合せまたは XML を表示することができる JavaBean です。
DBViewerBeanInfo クラス（10-73 ページ）	DBViewer Bean に関する情報を提供します。

DBViewer クラス

DBViewer の説明

このクラスは、XSL スタイルシートを適用し、結果の HTML を可視化して、スクロール可能な Swing パネル内にデータベース問合せまたは XML を表示することができる JavaBean です。この Bean には、XML バッファ、XSL バッファおよび結果バッファの 3 つのバッファがあります。この Bean API を使用すると、プログラムをコールして、様々なソースからバッファをロードまたは保存したり、XSL バッファ内のスタイルシートを使用して、XML バッファにスタイルシート変換を適用することができます。結果は、結果バッファに格納できます。

XML バッファおよび XSL バッファの内容をソースまたはツリー構造として表示できます。結果バッファの内容は、HTML にレンダリングして、ソースまたはツリー構造で表示することもできます。XML バッファは、データベースの問合せからロードできます。

すべてのバッファには、Oracle データベース内の CLOB 表のファイルおよびファイル・システムのファイルをロードして保存できます。そのため、ファイル・システムとデータベース内のユーザー・スキーマ間のファイルの移動を制御できます。

DBViewer の構文

```
public class DBViewer extends javax.swing.JPanel implements java.io.Serializable

java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--oracle.xml.dbviewer.DBViewer
```

DBViewer の実装済インタフェース

- javax.accessibility.Accessible
- java.awt.image.ImageObserver
- java.awt.MenuContainer
- java.io.Serializable

DBViewer のメソッド

表 10-20 DBViewer のメソッドの概要

メソッド	説明
DBViewer()（10-48 ページ）	DBViewer の新しいインスタンスを作成します。
getHostname()（10-48 ページ）	データベースのホスト名を取得します。
getInstancename()（10-48 ページ）	データベースのインスタンス名を取得します。
getPassword()（10-49 ページ）	ユーザーのパスワードを取得します。
getPort()（10-49 ページ）	データベースのポート番号を文字列として取得します。
getResBuffer()（10-49 ページ）	結果バッファの内容を取得します。
getResCLOBFileName()（10-49 ページ）	結果の CLOB のファイル名を取得します。
getResCLOBTableName()（10-50 ページ）	結果の CLOB の表名を取得します。
getResFileName()（10-50 ページ）	結果のファイル名を取得します。
getUsername()（10-50 ページ）	ユーザー名を取得します。
getXmlBuffer()（10-50 ページ）	XML バッファの内容を取得します。
getXmlCLOBFileName()（10-51 ページ）	XML の CLOB のファイル名を取得します。
getXmlCLOBTableName()（10-51 ページ）	XML の CLOB の表名を取得します。
getXmlFileName()（10-51 ページ）	XML のファイル名を取得します。
getXMLStringFromSQL()（10-51 ページ）	SQL 問合せから結果セットの XML 表示を XML 文字列として取得します。
getXslBuffer()（10-52 ページ）	XSL バッファの内容を取得します。
getXslCLOBFileName()（10-52 ページ）	XSL の CLOB のファイル名を取得します。
getXslCLOBTableName()（10-52 ページ）	XSL の CLOB の表名を取得します。
getXslFileName()（10-52 ページ）	XSL のファイル名を取得します。
loadResBuffer()（10-53 ページ）	結果バッファをロードします。
loadResBufferFromClob()（10-53 ページ）	CLOB ファイルから結果バッファをロードします。
loadResBufferFromFile()（10-54 ページ）	ファイルから結果バッファをロードします。
loadXmlBuffer()（10-54 ページ）	XML バッファをロードします。

表 10-20 DBViewer のメソッドの概要（続き）

メソッド	説明
loadXmlBufferFromClob()（10-55 ページ）	CLOB ファイルから XML バッファをロードします。
loadXmlBufferFromFile()（10-55 ページ）	ファイルから XML バッファをロードします。
loadXMLBufferFromSQL()（10-55 ページ）	SQL 結果セットから XML バッファをロードします。
loadXslBuffer()（10-56 ページ）	ファイルから XSL バッファをロードします。
loadXslBufferFromClob()（10-56 ページ）	CLOB ファイルから XSL バッファをロードします。
loadXslBufferFromFile()（10-57 ページ）	ファイルから XSL バッファをロードします。
parseResBuffer()（10-57 ページ）	結果バッファを解析し、ツリー・ビューおよびソース・ビューをリフレッシュして、XML 文書を戻します。
parseXmlBuffer()（10-57 ページ）	XML バッファを解析し、ツリー・ビューおよびソース・ビューをリフレッシュして、XML 文書を戻します。
parseXslBuffer()（10-57 ページ）	XSL バッファを解析し、ツリー・ビューおよびソース・ビューをリフレッシュして、XML 文書を戻します。
saveResBuffer()（10-58 ページ）	結果バッファをファイルに保存します。
saveResBufferToClob()（10-58 ページ）	結果バッファを CLOB ファイルに保存します。
saveResBufferToFile()（10-58 ページ）	結果バッファをファイルに保存します。
saveXmlBuffer()（10-59 ページ）	XML バッファをファイルに保存します。
saveXmlBufferToClob()（10-59 ページ）	XML バッファを CLOB ファイルに保存します。
saveXmlBufferToFile()（10-59 ページ）	XML バッファをファイルに保存します。
saveXslBuffer()（10-60 ページ）	XSL バッファをファイルに保存します。
saveXslBufferToClob()（10-60 ページ）	XSL バッファを CLOB ファイルに保存します。
saveXslBufferToFile()（10-60 ページ）	XSL バッファをファイルに保存します。
setHostname()（10-61 ページ）	データベースのホスト名を設定します。
setInstancename()（10-61 ページ）	データベースのインスタンス名を設定します。
setPassword()（10-61 ページ）	ユーザーのパスワードを設定します。
setPort()（10-62 ページ）	データベースのポート番号を設定します。

表 10-20 DBViewer のメソッドの概要（続き）

メソッド	説明
setResBuffer()（10-62 ページ）	結果バッファの新しいテキストを設定します。
setResCLOBFileName()（10-62 ページ）	結果の CLOB のファイル名を設定します。
setResCLOBTableName()（10-63 ページ）	結果の CLOB の表名を設定します。
setResFileName()（10-63 ページ）	結果のファイル名を設定します。
setResHtmlView()（10-63 ページ）	結果バッファをレンダリング済 HTML として表示します。
setResSourceEditView()（10-64 ページ）	結果バッファを XML ソースとして表示し、編集モードに入ります。
setResSourceView()（10-64 ページ）	結果バッファを XML ソースとして表示します。
setResTreeView()（10-65 ページ）	結果バッファを XML ツリー・ビューとして表示します。
setUsername()（10-65 ページ）	ユーザー名を設定します。
setXmlBuffer()（10-66 ページ）	XML バッファの新しいテキストを設定します。
setXmlCLOBFileName()（10-66 ページ）	XML の CLOB のファイル名を設定します。
setXmlCLOBTableName()（10-66 ページ）	XML の CLOB の表名を設定します。
setXmlFileName()（10-67 ページ）	XML のファイル名を設定します。
setXmlSourceEditView()（10-67 ページ）	XML バッファを XML ソースとして表示し、編集モードに入ります。
setXmlSourceView()（10-68 ページ）	XML バッファを XML ソースとして表示します。
setXmlTreeView()（10-68 ページ）	XML バッファをツリーとして表示します。
setXslBuffer()（10-69 ページ）	XSL バッファに新しいテキストを設定します。
setXslCLOBFileName()（10-69 ページ）	XSL の CLOB のファイル名を設定します。
setXslCLOBTableName()（10-69 ページ）	XSL の CLOB の表名を設定します。
setXslFileName()（10-70 ページ）	XSL のファイル名を設定します。
setXslSourceEditView()（10-70 ページ）	XSL バッファを XML ソースとして表示し、編集モードに入ります。
setXslSourceView()（10-71 ページ）	XSL バッファを XML ソースとして表示します。
setXslTreeView()（10-71 ページ）	XSL バッファをツリーとして表示します。

表 10-20 DBViewer のメソッドの概要（続き）

メソッド	説明
transformToDoc()（10-71 ページ）	XSL バッファのスタイルシートを適用して、XML バッファの内容を変換します。
transformToRes()（10-72 ページ）	XML バッファ内の XML に XSL バッファのスタイルシート変換を適用し、その結果を結果バッファに格納します。
transformToString()（10-72 ページ）	XSL バッファのスタイルシートを適用して、XML バッファの内容を変換します。

DBViewer()

説明

DBViewer の新しいインスタンスを作成します。

構文

```
public  DBViewer();
```

getHostname()

説明

データベースのホスト名を取得します。

構文

```
public java.lang.String getHostname();
```

getInstancename()

説明

データベースのインスタンス名を取得します。

構文

```
public java.lang.String getInstancename();
```

getPassword()

説明
ユーザーのパスワードを取得します。

構文
```
public java.lang.String getPassword();
```

getPort()

説明
データベースのポート番号を文字列として取得します。

構文
```
public java.lang.String getPort();
```

getResBuffer()

説明
結果バッファの内容を取得します。

構文
```
public java.lang.String getResBuffer();
```

getResCLOBFileName()

説明
結果の CLOB のファイル名を取得します。

構文
```
public java.lang.String getResCLOBFileName();
```

getResCLOBTableName()

説明
結果の CLOB の表名を取得します。

構文
```
public java.lang.String getResCLOBTableName();
```

getResFileName()

説明
結果のファイル名を取得します。

構文
```
public java.lang.String getResFileName();
```

getUsername()

説明
ユーザー名を取得します。

構文
```
public java.lang.String getUsername();
```

getXmlBuffer()

説明
XML バッファの内容を取得します。

構文
```
public java.lang.String getXmlBuffer();
```

getXmlCLOBFileName()

説明
XML の CLOB のファイル名を取得します。

構文
```
public java.lang.String getXmlCLOBFileName();
```

getXmlCLOBTableName()

説明
XML の CLOB の表名を取得します。

構文
```
public java.lang.String getXmlCLOBTableName();
```

getXmlFileName()

説明
XML のファイル名を取得します。

構文
```
public java.lang.String getXmlFileName();
```

getXMLStringFromSQL()

説明
SQL 問合せから結果セットの XML 表示を XML 文字列として取得します。

構文
```
public java.lang.String getXMLStringFromSQL( java.lang.String sqlText);
```

パラメータ	説明
sqlText	SQL テキスト。

getXslBuffer()

説明
XSL バッファの内容を取得します。

構文
```
public java.lang.String getXslBuffer();
```

getXslCLOBFileName()

説明
XSL の CLOB のファイル名を取得します。

構文
```
public java.lang.String getXslCLOBFileName();
```

getXslCLOBTableName()

説明
XSL の CLOB の表名を取得します。

構文
```
public java.lang.String getXslCLOBTableName();
```

getXslFileName()

説明
XSL のファイル名を取得します。

構文
```
public java.lang.String getXslFileName();
```

loadResBuffer()

説明

結果バッファをロードします。次の表に、オプションを示します。

構文	説明
public void loadResBuffer(String filename);	ファイルから結果バッファをロードします。
public void loadResBuffer(String clobTablename, String clobFilename);	CLOB ファイルから結果バッファをロードします。
public void loadResBuffer(oracle.xml.parser.v2.XMLDocument resDoc);	XML 文書から結果バッファをロードします。

パラメータ	説明
filename	ファイル名。
clobTablename	CLOB の表名。
clobFilename	CLOB のファイル名。
resDoc	XML 文書。

loadResBufferFromClob()

説明

CLOB ファイルから結果バッファをロードします。

構文

```
public void loadResBufferFromClob();
```

loadResBufferFromFile()

説明

ファイルから結果バッファをロードします。

構文

```
public void loadResBufferFromFile();
```

loadXmlBuffer()

説明

XML バッファをロードします。次の表に、オプションを示します。

構文	説明
public void loadXmlBuffer(j String filename);	ファイルから XML バッファをロードします。
public void loadXmlBuffer(String clobTablename, String clobFilename);	CLOB ファイルから XML バッファをロードします。
public void loadXmlBuffer(oracle.xml.parser.v2.XMLDocument xmlDoc);	XML 文書から XML バッファをロードします。

パラメータ	説明
filename	ファイル名。
clobTablename	CLOB の表名。
clobFilename	CLOB のファイル名。
xmlDoc	XML 文書。

loadXmlBufferFromClob()

説明

CLOB ファイルから XML バッファをロードします。

構文

```
public void loadXmlBufferFromClob();
```

loadXmlBufferFromFile()

説明

ファイルから XML バッファをロードします。

構文

```
public void loadXmlBufferFromFile();
```

loadXMLBufferFromSQL()

説明

SQL 結果セットから XML バッファをロードします。

構文

```
public void loadXMLBufferFromSQL( String sqlText);
```

パラメータ	説明
sqlText	SQL テキスト。

loadXslBuffer()

説明

ファイルから XSL バッファをロードします。次の表に、オプションを示します。

構文	説明
public void loadXslBuffer(String filename);	ファイルから XSL バッファをロードします。
public void loadXslBuffer(String clobTablename, String clobFilename);	CLOB ファイルから XSL バッファをロードします。
public void loadXslBuffer(oracle.xml.parser.v2.XMLDocument xslDoc);	XML 文書から XSL バッファをロードします。

パラメータ	説明
filename	ファイル名。
clobTablename	CLOB の表名。
clobFilename	CLOB のファイル名。
xslDoc	XML 文書。

loadXslBufferFromClob()

説明

CLOB ファイルから XSL バッファをロードします。

構文

```
public void loadXslBufferFromClob();
```

loadXslBufferFromFile()

説明

ファイルから XSL バッファをロードします。

構文

```
public void loadXslBufferFromFile();
```

parseResBuffer()

説明

結果バッファを解析し、ツリー・ビューおよびソース・ビューをリフレッシュして、XML 文書を戻します。

構文

```
public oracle.xml.parser.v2.XMLDocument parseResBuffer();
```

parseXmlBuffer()

説明

XML バッファを解析し、ツリー・ビューおよびソース・ビューをリフレッシュして、XML 文書を戻します。

構文

```
public oracle.xml.parser.v2.XMLDocument parseXmlBuffer();
```

parseXslBuffer()

説明

XSL バッファを解析し、ツリー・ビューおよびソース・ビューをリフレッシュして、XML 文書を戻します。

構文

```
public oracle.xml.parser.v2.XMLDocument parseXslBuffer();
```

saveResBuffer()

説明

結果バッファをファイルに保存します。次の表に、オプションを示します。

構文	説明
public void saveResBuffer(String filename);	結果バッファをファイルに保存します。
public void saveResBuffer(String tablename, String filename);	結果バッファを CLOB ファイルに保存します。

パラメータ	説明
tablename	CLOB の表名。
filename	CLOB のファイル名。

saveResBufferToClob()

説明

結果バッファを CLOB ファイルに保存します。

構文

```
public void saveResBufferToClob();
```

saveResBufferToFile()

説明

結果バッファをファイルに保存します。

構文

```
public void saveResBufferToFile();
```

saveXmlBuffer()

XML バッファをファイルに保存します。次の表に、オプションを示します。

構文	説明
public void saveXmlBuffer(String filename);	XML バッファをファイルに保存します。
public void saveXmlBuffer(String tablename, String filename);	XML バッファを CLOB ファイルに保存します。

パラメータ	説明
tablename	CLOB の表名。
filename	CLOB のファイル名。

saveXmlBufferToClob()

説明

XML バッファを CLOB ファイルに保存します。

構文

```
public void saveXmlBufferToClob();
```

saveXmlBufferToFile()

説明

XML バッファをファイルに保存します。

構文

```
public void saveXmlBufferToFile();
```

saveXslBuffer()

説明

XSL バッファをファイルに保存します。次の表に、オプションを示します。

構文	説明
public void saveXslBuffer(String fileName);	XSL バッファをファイル・システムのファイルに保存します。
public void saveXslBuffer(String tableName, String fileName);	XSL バッファを CLOB ファイルに保存します。

パラメータ	説明
tableName	表名。
fileName	ファイル名。

saveXslBufferToClob()

説明

XSL バッファを CLOB ファイルに保存します。

構文

```
public void saveXslBufferToClob();
```

saveXslBufferToFile()

説明

XSL バッファをファイルに保存します。

構文

```
public void saveXslBufferToFile();
```

setHostname()

説明
データベースのホスト名を設定します。

構文
```
public void setHostname( java.lang.String hostname);
```

パラメータ	説明
hostname	ホスト名。

setInstancename()

説明
データベースのインスタンス名を設定します。

構文
```
public void setInstancename( String instancename);
```

パラメータ	説明
instancename	データベースのインスタンス名。

setPassword()

説明
ユーザーのパスワードを設定します。

構文
```
public void setPassword( String password);
```

パラメータ	説明
password	ユーザーのパスワード。

setPort()

説明
データベースのポート番号を設定します。

構文
```
public void setPort( String port);
```

パラメータ	説明
port	ポート番号を含む文字列。

setResBuffer()

説明
結果バッファの新しいテキストを設定します。

構文
```
public void setResBuffer( String text);
```

パラメータ	説明
text	新しいテキスト。

setResCLOBFileName()

説明
結果の CLOB のファイル名を設定します。

構文
```
public void setResCLOBFileName( String name);
```

パラメータ	説明
name	結果の CLOB のファイル名。

setResCLOBTableName()

説明
結果の CLOB の表名を設定します。

構文
```
public void setResCLOBTableName( String name);
```

パラメータ	説明
name	結果の CLOB の表名。

setResFileName()

説明
結果のファイル名を設定します。

構文
```
public void setResFileName( String name);
```

パラメータ	説明
name	結果のファイル名。

setResHtmlView()

説明
結果バッファをレンダリング済 HTML として表示します。

構文
```
public void setResHtmlView( boolean on);
```

パラメータ	説明
on	結果バッファを HTML として表示するかどうかを切り替えます。HTML として表示する場合は true、表示しない場合は false を指定します。

setResSourceEditView()

説明

結果バッファを XML ソースとして表示し、編集モードに入ります。

構文

```
public void setResSourceEditView( boolean on);
```

パラメータ	説明
on	結果バッファを編集モードで HTML として表示するかどうかを切り替えます。編集モードで HTML として表示する場合は true、表示しない場合は false を指定します。

setResSourceView()

説明

結果バッファを XML ソースとして表示します。

構文

```
public void setResSourceView( boolean on);
```

パラメータ	説明
on	結果バッファを XML ソースとして表示するかどうかを切り替えます。XML ソースとして表示する場合は true、表示しない場合は false を指定します。

setResTreeView()

説明

結果バッファを XML ツリー・ビューとして表示します。

構文

```
public void setResTreeView( boolean on);
```

パラメータ	説明
on	結果バッファを XML ツリー・ビューとして表示するかどうかを切り替えます。XML ツリー・ビューとして表示する場合は `true`、表示しない場合は `false` を指定します。

setUsername()

説明

ユーザー名を設定します。

構文

```
public void setUsername( String username);
```

パラメータ	説明
username	ユーザー名。

setXmlBuffer()

説明
XML バッファの新しいテキストを設定します。

構文
```
public void setXmlBuffer( String text)
```

パラメータ	説明
text	XML テキスト。

setXmlCLOBFileName()

説明
XML の CLOB のファイル名を設定します。

構文
```
public void setXmlCLOBFileName( String name);
```

パラメータ	説明
name	XML の CLOB のファイル名。

setXmlCLOBTableName()

説明
XML の CLOB の表名を設定します。

構文
```
public void setXmlCLOBTableName( String name);
```

パラメータ	説明
name	XML の CLOB の表名。

setXmlFileName()

説明

XML のファイル名を設定します。

構文

```
public void setXmlFileName( String name);
```

パラメータ	説明
name	XML のファイル名。

setXmlSourceEditView()

説明

XML バッファを XML ソースとして表示し、編集モードに入ります。

構文

```
public void setXmlSourceEditView( boolean on);
```

パラメータ	説明
on	XML バッファを編集モードで XML ソースとして表示するかどうかを切り替えます。編集モードで XML ソースとして表示する場合は `true`、表示しない場合は `false` を指定します。

setXmlSourceView()

説明

XML バッファを XML ソースとして表示します。

構文

```
public void setXmlSourceView( boolean on);
```

パラメータ	説明
on	XML バッファを XML ソースとして表示するかどうかを切り替えます。XML バッファとして表示する場合は `true`、表示しない場合は `false` を指定します。

setXmlTreeView()

説明

XML バッファをツリーとして表示します。

構文

```
public void setXmlTreeView( boolean on);
```

パラメータ	説明
on	XML バッファを XML ツリーとして表示するかどうかを切り替えます。XML ツリーとして表示する場合は `true`、表示しない場合は `false` を指定します。

setXslBuffer()

説明

XSL バッファに新しいテキストを設定します。

構文

```
public void setXslBuffer( String text);
```

パラメータ	説明
text	XSL テキスト。

setXslCLOBFileName()

説明

XSL の CLOB のファイル名を設定します。

構文

```
public void setXslCLOBFileName( String name);
```

パラメータ	説明
name	XSL の CLOB のファイル名。

setXslCLOBTableName()

説明

XSL の CLOB の表名を設定します。

構文

```
public void setXslCLOBTableName( String name);
```

パラメータ	説明
name	XSL の CLOB の表名。

setXslFileName()

説明

XSL のファイル名を設定します。

構文

```
public void setXslFileName( String name);
```

パラメータ	説明
name	XSL のファイル名。

setXslSourceEditView()

説明

XSL バッファを XML ソースとして表示し、編集モードに入ります。

構文

```
public void setXslSourceEditView( boolean on);
```

パラメータ	説明
on	XSL バッファを編集モードで XML ソースとして表示するかどうかを切り替えます。編集モードで XSL ソースとして表示する場合は `true`、表示しない場合は `false` を指定します。

setXslSourceView()

説明

XSL バッファを XML ソースとして表示します。

構文

```
public void setXslSourceView( boolean on);
```

パラメータ	説明
on	XSL バッファを XML ソースとして表示するかどうかを切り替えます。XML ソースとして表示する場合は `true`、表示しない場合は `false` を指定します。

setXslTreeView()

説明

XSL バッファをツリーとして表示します。

構文

```
public void setXslTreeView( boolean on);
```

パラメータ	説明
on	XSL バッファを XSL ツリーとして表示するかどうかを切り替えます。XSL ツリーとして表示する場合は `true`、表示しない場合は `false` を指定します。

transformToDoc()

説明

XSL バッファのスタイルシートを適用して、XML バッファの内容を変換します。

構文

```
public oracle.xml.parser.v2.XMLDocument transformToDoc();
```

transformToRes()

説明

XML バッファ内の XML に XSL バッファのスタイルシート変換を適用し、その結果を結果バッファに格納します。

構文

```
public void transformToRes();
```

transformToString()

説明

XSL バッファのスタイルシートを適用して、XML バッファの内容を変換します。

構文

```
public java.lang.String transformToString();
```

DBViewerBeanInfo クラス

DBViewerBeanInfo の説明

このクラスは、DBViewer Bean に関する情報を提供します。

DBViewerBeanInfo の構文

```
public class DBViewerBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.dbviewer.DBViewerBeanInfo
```

DBViewerBeanInfo の実装済インタフェース

```
java.beans.BeanInfo
```

DBViewerBeanInfo のメソッド

表 10-21 DBViewerBeanInfo のメソッドの概要

メソッド	説明
DBViewerBeanInfo()（10-73 ページ）	クラス・コンストラクタです。
getIcon()（10-74 ページ）	ツールバーやツールボックスなどで、DBViewer Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。
getPropertyDescriptors()（10-74 ページ）	DBViwer Bean の編集可能な PropertyDescriptiors の配列を取得します。

DBViewerBeanInfo()

説明

クラス・コンストラクタです。

構文

```
public  DBViewerBeanInfo();
```

getIcon()

説明

ツールバーやツールボックスなどで、DBViewer Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。java.beans.SimpleBeanInfo クラスの getIcon() をオーバーライドします。

構文

```
public java.awt.Image getIcon( int iconKind);
```

パラメータ	説明
iconKind	リクエストされたアイコンの種類。

getPropertyDescriptors()

説明

DBViwer Bean の編集可能な PropertyDescriptiors の配列を取得します。java.beans.SimpleBeanInfo クラスの getPropertyDescriptors() をオーバーライドします。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

oracle.xml.srcviewer パッケージ

oracle.xml.srcviewer の説明

このパッケージは、構文をハイライト表示した XML ソース文書を表示するために使用される可視 Bean です。異なる XML 言語要素のカラー、フォントおよびサイズをカスタマイズできます。

表 10-22 に、oracle.xml.srcviewer クラスの概要を示します。

表 10-22 oracle.xml.srcviewer のクラスの概要

クラス	説明
XMLSourceView クラス （10-76 ページ）	XML 文書を表示します。
XMLSourceViewBeanInfo クラス （10-95 ページ）	XMLSourceView Bean に関する情報を提供します。

XMLSourceView クラス

XMLSourceView の説明

このクラスは、XML 文書を表示します。タグ、属性名、属性値、コメント、CDATA、PCDATA、処理命令（PI）データ、処理命令（PI）名および表記法の XMLToken 型を認識します。各トークン型には、フォアグラウンド・カラーおよびフォアグラウンド・フォントがあります。デフォルトのカラーおよびフォントの設定は、ユーザーが変更できます。この Bean は、入力として org.w3c.dom.Document オブジェクトを取ります。

XMLSourceView の構文

```
public class XMLSourceView extends javax.swing.JPanel implements
java.io.Serializable

java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--oracle.xml.srcviewer.XMLSourceView
```

XMLSourceView の実装済インタフェース

- javax.accessibility.Accessible
- java.awt.image.ImageObserver
- java.awt.MenuContainer
- java.io.Serializable

XMLSourceView のフィールド

表 10-23 ElementDecl のフィールド

フィールド	構文	説明
inputDOMDocument	protected org.w3c.dom.Document inputDOMDocument	表示される XML 文書。
jScrollPane	protected javax.swing.JScrollPane jScrollPane	スクロール可能にするために XMLSourveView が使用する Java Swing コンポーネント。
jTextPane	protected javax.swing.JTextPane jTextPane	テキストを表示するために XMLSourveView が使用する Java Swing コンポーネント。
xmlStyledDocument	protected oracle.xml.srcviewer.XMLStyledDocument xmlStyledDocument	スタイル化された XML 文書。XMLToken を、フォントやカラーなどの属性に関連付けます。

XMLSourceView のメソッド

表 10-24 XMLSourceView のメソッドの概要

メソッド	説明
XMLSourceView()（10-80 ページ）	クラス・コンストラクタです。XMLSourceView 型のオブジェクトを作成します。
fontGet()（10-80 ページ）	指定された属性セットからフォントを取得して戻します。
fontSet()（10-80 ページ）	可変属性セットのフォントを設定します。
getAttributeNameFont()（10-81 ページ）	属性名に使用するフォントを戻します。
getAttributeNameForeground()（10-81 ページ）	属性名に使用するフォアグラウンド・カラーを戻します。
getAttributeValueFont()（10-81 ページ）	属性値に使用するフォントを戻します。
getAttributeValueForeground()（10-81 ページ）	属性値に使用するフォアグラウンド・カラーを戻します。

表 10-24 XMLSourceView のメソッドの概要（続き）

メソッド	説明
getBackground()（10-82 ページ）	バックグラウンド・カラーを戻します。
getCDATAFont()（10-82 ページ）	CDATA に使用するフォントを戻します。
getCDATAForeground()（10-82 ページ）	CDATA に使用するフォアグラウンド・カラーを戻します。
getCommentDataFont()（10-82 ページ）	コメントに使用するフォントを戻します。
getCommentDataForeground()（10-83 ページ）	コメントに使用するフォアグラウンド・カラーを戻します。
getEditedText()（10-83 ページ）	編集済テキストを戻します。
getJTextPane()（10-83 ページ）	XMLSourceViewer が使用する `JTextPane` ビューアのコンポーネントを戻します。
getMinimumSize()（10-83 ページ）	XMLSourceView の最小サイズを戻します。
getNodeAtOffset()（10-84 ページ）	指定されたオフセットにある XML ノードを戻します。
getPCDATAFont()（10-84 ページ）	PCDATA に使用するフォントを戻します。
getPCDATAForeground()（10-84 ページ）	PCDATA に使用するフォアグラウンド・カラーを戻します。
getPIDataFont()（10-84 ページ）	処理命令データに使用するフォントを戻します。
getPIDataForeground()（10-85 ページ）	処理命令データに使用するフォアグラウンド・カラーを戻します。
getPINameFont()（10-85 ページ）	処理命令名に使用するフォントを戻します。
getPINameForeground()（10-85 ページ）	処理命令名に使用するフォアグラウンド・カラーを戻します。
getSymbolFont()（10-85 ページ）	表記法に使用するフォントを戻します。
getSymbolForeground()（10-86 ページ）	表記法に使用するフォアグラウンド・カラーを戻します。
getTagFont()（10-86 ページ）	タグに使用するフォントを戻します。
getTagForeground()（10-86 ページ）	タグに使用するフォアグラウンド・カラーを戻します。
getText()（10-86 ページ）	XML 文書を文字列として戻します。
isEditable()（10-87 ページ）	オブジェクトが編集可能かどうかを示すブール値を戻します。

表 10-24 XMLSourceView のメソッドの概要（続き）

メソッド	説明
selectNodeAt()（10-87 ページ）	指定されたオフセットにある XML ノードにカーソルを移動します。
setAttributeNameFont()（10-87 ページ）	属性名に使用するフォントを設定します。
setAttributeNameForeground()（10-88 ページ）	属性名に使用するフォアグラウンド・カラーを設定します。
setAttributeValueFont()（10-88 ページ）	属性値に使用するフォントを設定します。
setAttributeValueForeground()（10-88 ページ）	属性値に使用するフォアグラウンド・カラーを設定します。
setBackground()（10-89 ページ）	バックグラウンド・カラーを設定します。
setCDATAFont()（10-89 ページ）	CDATA に使用するフォントを設定します。
setCDATAForeground()（10-89 ページ）	CDATA に使用するフォアグラウンド・カラーを設定します。
setCommentDataFont()（10-90 ページ）	コメントに使用するフォントを設定します。
setCommentDataForeground()（10-90 ページ）	コメントに使用するフォアグラウンド・カラーを設定します。
setEditable()（10-90 ページ）	オブジェクトを編集可能にするかどうかを示すために指定されたブール値を設定します。
setPCDATAFont()（10-91 ページ）	PCDATA に使用するフォントを設定します。
setPCDATAForeground()（10-91 ページ）	PCDATA に使用するフォアグラウンド・カラーを設定します。
setPIDataFont()（10-91 ページ）	処理命令データに使用するフォントを設定します。
setPIDataForeground()（10-92 ページ）	処理命令データに使用するフォアグラウンド・カラーを設定します。
setPINameFont()（10-92 ページ）	処理命令名に使用するフォントを設定します。
setPINameForeground()（10-92 ページ）	処理命令名に使用するフォアグラウンド・カラーを設定します。
setSelectedNode()（10-93 ページ）	選択された XML ノードにカーソル位置を設定します。
setSymbolFont()（10-93 ページ）	表記法に使用するフォントを設定します。
setSymbolForeground()（10-93 ページ）	表記法に使用するフォアグラウンド・カラーを設定します。

表 10-24 XMLSourceView のメソッドの概要（続き）

メソッド	説明
setTagFont()（10-94 ページ）	タグに使用するフォントを設定します。
setTagForeground()（10-94 ページ）	タグに使用するフォアグラウンド・カラーを設定します。
setXMLDocument()（10-94 ページ）	XMLviewer と XML 文書を対応付けます。

XMLSourceView()

説明

クラス・コンストラクタです。XMLSourceView 型のオブジェクトを作成します。

構文

```
public  XMLSourceView();
```

fontGet()

説明

指定された属性セットからフォントを取得して戻します。

構文

```
public static java.awt.Font fontGet(
                javax.swing.text.AttributeSet attributeSet);
```

パラメータ	説明
attributeSet	ソース AttributeSet。

fontSet()

説明

可変属性セットのフォントを設定します。

構文

```
public static void fontSet(
                javax.swing.text.MutableAttributeSet mutAttributeSet,
                java.awt.Font font);
```

パラメータ	説明
mutAttributeSet	更新する可変属性セット。
font	可変属性セットの新しいフォント。

getAttributeNameFont()

説明
属性名に使用するフォントを戻します。

構文
```
public java.awt.Font getAttributeNameFont();
```

getAttributeNameForeground()

説明
属性名に使用するフォアグラウンド・カラーを戻します。

構文
```
public java.awt.Color getAttributeNameForeground();
```

getAttributeValueFont()

説明
属性値に使用するフォントを戻します。

構文
```
public java.awt.Font getAttributeValueFont();
```

getAttributeValueForeground()

説明
属性値に使用するフォアグラウンド・カラーを戻します。

構文
```
public java.awt.Color getAttributeValueForeground();
```

getBackground()

説明

バックグラウンド・カラーを戻します。java.awt.Component クラスの getBackground() をオーバーライドします。

構文

```
public java.awt.Color getBackground();
```

getCDATAFont()

説明

CDATA に使用するフォントを戻します。

構文

```
public java.awt.Font getCDATAFont();
```

getCDATAForeground()

説明

CDATA に使用するフォアグラウンド・カラーを戻します。

構文

```
public java.awt.Color getCDATAForeground();
```

getCommentDataFont()

説明

コメントに使用するフォントを戻します。

構文

```
public java.awt.Font getCommentDataFont();
```

getCommentDataForeground()

説明

コメントに使用するフォアグラウンド・カラーを戻します。

構文

```
public java.awt.Color getCommentDataForeground();
```

getEditedText()

説明

編集済テキストを戻します。

構文

```
public java.lang.String getEditedText();
```

getJTextPane()

説明

XMLSourceViewer が使用する `JTextPane` ビューアのコンポーネントを戻します。

構文

```
public javax.swing.JTextPane getJTextPane();
```

getMinimumSize()

説明

XMLSourceView の最小サイズを戻します。javax.swing.JComponent クラスの getMinimumSize() をオーバーライドします。

構文

```
public java.awt.Dimension getMinimumSize();
```

getNodeAtOffset()

説明
指定されたオフセットにある XML ノードを戻します。

構文
```
public org.w3c.dom.Node getNodeAtOffset( int i);
```

パラメータ	説明
i	ノード・オフセット。

getPCDATAFont()

説明
PCDATA に使用するフォントを戻します。

構文
```
public java.awt.Font getPCDATAFont();
```

getPCDATAForeground()

説明
PCDATA に使用するフォアグラウンド・カラーを戻します。

構文
```
public java.awt.Color getPCDATAForeground();
```

getPIDataFont()

説明
処理命令データに使用するフォントを戻します。

構文
```
public java.awt.Font getPIDataFont();
```

getPIDataForeground()

説明
処理命令データに使用するフォアグラウンド・カラーを戻します。

構文
```
public java.awt.Color getPIDataForeground();
```

getPINameFont()

説明
処理命令名に使用するフォントを戻します。

構文
```
public java.awt.Font getPINameFont();
```

getPINameForeground()

説明
処理命令名に使用するフォアグラウンド・カラーを戻します。

構文
```
public java.awt.Color getPINameForeground();
```

getSymbolFont()

説明
表記法に使用するフォントを戻します。

構文
```
public java.awt.Font getSymbolFont();
```

getSymbolForeground()

説明
表記法に使用するフォアグラウンド・カラーを戻します。

構文
```
public java.awt.Color getSymbolForeground();
```

getTagFont()

説明
タグに使用するフォントを戻します。

構文
```
public java.awt.Font getTagFont();
```

getTagForeground()

説明
タグに使用するフォアグラウンド・カラーを戻します。

構文
```
public java.awt.Color getTagForeground();
```

getText()

説明
XML 文書を文字列として戻します。

構文
```
public java.lang.String getText();
```

isEditable()

説明
オブジェクトが編集可能かどうかを示すブール値を戻します。

構文
```
public boolean isEditable();
```

selectNodeAt()

説明
指定されたオフセットにある XML ノードにカーソルを移動します。

構文
```
public void selectNodeAt( int i);
```

パラメータ	説明
i	ノード・オフセット。

setAttributeNameFont()

説明
属性名に使用するフォントを設定します。

構文
```
public void setAttributeNameFont( java.awt.Font font);
```

パラメータ	説明
font	新しいフォント属性名。

setAttributeNameForeground()

説明
属性名に使用するフォアグラウンド・カラーを設定します。

構文
```
public void setAttributeNameForeground( java.awt.Color color);
```

パラメータ	説明
color	属性名用の新しいカラー。

setAttributeValueFont()

説明
属性値に使用するフォントを設定します。

構文
```
public void setAttributeValueFont( java.awt.Font font);
```

パラメータ	説明
font	新しいフォント属性値。

setAttributeValueForeground()

説明
属性値に使用するフォアグラウンド・カラーを設定します。

構文
```
public void setAttributeValueForeground( java.awt.Color color);
```

パラメータ	説明
color	属性値用の新しいカラー。

setBackground()

説明

バックグラウンド・カラーを設定します。javax.swing.JComponent クラスの setBackground() をオーバーライドします。

構文

```
public void setBackground( java.awt.Color color);
```

パラメータ	説明
font	新しいバックグラウンド・カラー。

setCDATAFont()

説明

CDATA に使用するフォントを設定します。

構文

```
public void setCDATAFont( java.awt.Font font);
```

パラメータ	説明
font	CDATA 用の新しいフォント。

setCDATAForeground()

説明

CDATA に使用するフォアグラウンド・カラーを設定します。

構文

```
public void setCDATAForeground( java.awt.Color color);
```

パラメータ	説明
color	CDATA 用の新しいカラー。

setCommentDataFont()

説明
コメントに使用するフォントを設定します。

構文
```
public void setCommentDataFont( java.awt.Font font);
```

パラメータ	説明
font	XML コメント用の新しいフォント。

setCommentDataForeground()

説明
コメントに使用するフォアグラウンド・カラーを設定します。

構文
```
public void setCommentDataForeground( java.awt.Color color);
```

パラメータ	説明
color	コメント用の新しいカラー。

setEditable()

説明
オブジェクトを編集可能にするかどうかを示すために指定されたブール値を設定します。

構文
```
public void setEditable( boolean edit);
```

パラメータ	説明
edit	オブジェクトを編集可能にするかどうかを示すフラグ。表示されるテキストを編集可能にする場合は `true`、編集可能にしない場合は `false` を指定します。

setPCDATAFont()

説明
PCDATA に使用するフォントを設定します。

構文
```
public void setPCDATAFont( java.awt.Font font);
```

パラメータ	説明
font	PCDATA 用の新しいフォント。

setPCDATAForeground()

説明
PCDATA に使用するフォアグラウンド・カラーを設定します。

構文
```
public void setPCDATAForeground( java.awt.Color color);
```

パラメータ	説明
color	PCDATA 用の新しいカラー。

setPIDataFont()

説明
処理命令データに使用するフォントを設定します。

構文
```
public void setPIDataFont( java.awt.Font font);
```

パラメータ	説明
font	処理命令データ用の新しいフォント。

setPIDataForeground()

説明
処理命令データに使用するフォアグラウンド・カラーを設定します。

構文
```
public void setPIDataForeground( java.awt.Color color);
```

パラメータ	説明
color	処理命令データ用の新しいカラー。

setPINameFont()

説明
処理命令名に使用するフォントを設定します。

構文
```
public void setPINameFont(java.awt.Font font);
```

パラメータ	説明
font	処理命令名用の新しいフォント。

setPINameForeground()

説明
処理命令名に使用するフォアグラウンド・カラーを設定します。

構文
```
public void setPINameForeground( java.awt.Color color);
```

パラメータ	説明
color	処理命令名用の新しいカラー。

setSelectedNode()

説明
選択された XML ノードにカーソル位置を設定します。

構文
```
public void setSelectedNode( org.w3c.dom.Node node);
```

パラメータ	説明
node	選択されたノード。

setSymbolFont()

説明
表記法に使用するフォントを設定します。

構文
```
public void setSymbolFont( java.awt.Font font);
```

パラメータ	説明
font	表記法用の新しいフォント。

setSymbolForeground()

説明
表記法に使用するフォアグラウンド・カラーを設定します。

構文
```
public void setSymbolForeground( java.awt.Color color);
```

パラメータ	説明
color	表記法用の新しいカラー。

setTagFont()

説明
タグに使用するフォントを設定します。

構文
```
public void setTagFont( java.awt.Font font);
```

パラメータ	説明
font	XML タグ用の新しいフォント。

setTagForeground()

説明
タグに使用するフォアグラウンド・カラーを設定します。

構文
```
public void setTagForeground( java.awt.Color color);
```

パラメータ	説明
color	XML タグ用の新しいカラー。

setXMLDocument()

説明
XMLviewer と XML 文書を対応付けます。

構文
```
public void setXMLDocument( org.w3c.dom.Document document);
```

パラメータ	説明
document	表示するドキュメント。

XMLSourceViewBeanInfo クラス

XMLSourceViewBeanInfo の説明

このクラスは、XMLSourceView Bean に関する情報を提供します。

XMLSourceViewBeanInfo の構文

```
public class XMLSourceViewBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.srcviewer.XMLSourceViewBeanInfo
```

XMLSourceViewBeanInfo の実装済インタフェース

```
java.beans.BeanInfo
```

XMLSourceViewBeanInfo のメソッド

表 10-25 XMLSourceViewBeanInfo のメソッドの概要

メソッド	説明
XMLSourceViewBeanInfo()（10-95 ページ）	クラス・コンストラクタです。
getIcon()（10-96 ページ）	ツールバーやツールボックスなどで、XMLSourceView Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。
getPropertyDescriptors()（10-96 ページ）	XMLSourveView Bean の編集可能な PropertyDescriptiors の配列を取得します。

XMLSourceViewBeanInfo()

説明

クラス・コンストラクタです。

構文

```
public  XMLSourceViewBeanInfo();
```

getIcon()

説明

ツールバーやツールボックスなどで、XMLSourceView Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。java.beans.SimpleBeanInfo クラスの getIcon() をオーバーライドします。

構文

```
public java.awt.Image getIcon( int iconKind);
```

パラメータ	説明
iconKind	リクエストされたアイコンの種類。

getPropertyDescriptors()

説明

XMLSourveView Bean の編集可能な PropertyDescriptiors の配列を取得します。java.beans.SimpleBeanInfo クラスの getPropertyDescriptors() をオーバーライドします。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

oracle.xml.transviewer パッケージ

oracle.xml.transviewer の説明

このパッケージは、可視 Bean です。このパッケージを使用すると、ユーザーは、ファイル・システムまたはデータベースの CLOB 表から XML バッファおよび XSL バッファをロードできます。XML バッファは、XSL バッファを使用して変換できます。XML バッファ、XSL バッファまたは HTML バッファは、ファイル・システムまたはデータベースに CLOB 表として保存できます。各 CLOB 表には、2 つの列が含まれ、1 つはファイル名を保持するためで文字列型、もう 1 つはファイル・データを保持するためで CLOB 型です。CLOB 表は、作成または削除できます。XML バッファおよび XSL バッファは編集および解析できます。

表 10-26 に、`oracle.xml.srcviewer` のクラスの概要を示します。

表 10-26 oracle.xml.transviewer のクラスの概要

クラス	説明
DBAccess クラス（10-98 ページ）	複数の XML 文書およびテキスト・ドキュメントを格納する CLOB 表を保持します。
DBAccessBeanInfo クラス（10-108 ページ）	DBAccess Bean に関する情報を提供します。
XMLTransformPanel クラス（10-110 ページ）	XSL 変換を XML 文書へ適用し、結果を可視化します。
XMLTransformPanelBeanInfo クラス（10-111 ページ）	XMLTransformPanel Bean に関する情報を提供します。
XMLTransViewer クラス（10-113 ページ）	XMLTransformPanel を使用する簡単なアプリケーションです。これはコマンドラインから使用でき、XML ファイルの編集および解析、XSL 変換の編集および適用、ファイル・システムまたはデータベース内の XML、XSL および結果ファイルの取得および保存を実行できます。

DBAccess クラス

DBAccess の説明

このクラスは、複数の XML 文書およびテキスト・ドキュメントを格納する CLOB 表を保持します。各表は、次の文を使用して作成されます。

```
CREATE TABLE tablename FILENAME CHAR( 16) UNIQUE, FILEDATA CLOB) LOB(FILEDATA) STORE AS (DISABLE STORAGE IN ROW).
```

- 各 XML（またはテキスト）文書が表に行として格納されます。FILENAME フィールドには、その行を検索、更新または削除を行うためにキーとして使用される一意の文字列があります。
- ドキュメント・テキストは FILEDATA フィールドに格納されます。これは CLOB オブジェクトです。
- これらの CLOB 表は Transviewer Bean が自動的に保持します。
- このクラスが保持する CLOB 表は、Transviewer Bean で使用できます。
- このクラスは、CLOB 表の作成および削除、CLOB 表の内容の表示、および CLOB 表にあるテキスト・ドキュメントの追加、置換、削除を行います。

DBAccess の構文

```
public class DBAccess extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.transviewer.DBAccess
```

DBAccess のメソッド

表 10-27 DBAcceess のメソッドの概要

メソッド	説明
DBAccess()（10-100 ページ）	クラス・コンストラクタです。
createBLOBTable()（10-100 ページ）	BLOB 表を作成します。正常に終了した場合は true を戻します。
createXMLTable()（10-100 ページ）	XML 表を作成します。正常に終了した場合は true を戻します。
deleteBLOBName()（10-101 ページ）	BLOB 表からバイナリ・ファイルを削除します。正常に終了した場合は true を戻します。

表 10-27 DBAcceess のメソッドの概要（続き）

メソッド	説明
deleteXMLName()（10-101 ページ）	XML 表からファイルを削除します。正常に終了した場合は true を戻します。
dropBLOBTable()（10-102 ページ）	BLOB 表を削除します。正常に終了した場合は true を戻します。
dropXMLTable()（10-102 ページ）	XML 表を削除します。正常に終了した場合は true を戻します。
getBLOBData()（10-103 ページ）	BLOB 表からバイナリ・ファイルをバイナリ配列として取得します。正常に終了した場合は true を戻します。
getNameSize()（10-103 ページ）	ファイル名が格納されているフィールドのサイズを戻します。
getXMLData()（10-103 ページ）	XML 表からテキスト・ファイルを文字列として取得します。
getXMLNames()（10-104 ページ）	XML 表のすべてのファイル名を文字列の配列として戻します。
getXMLTableNames()（10-104 ページ）	ユーザー指定の文字列で始まるすべての XML 表名の配列を取得します。
insertBLOBData()（10-105 ページ）	バイナリ・ファイルを BLOB 表の行として挿入します。正常に終了した場合は true を戻します。
insertXMLData()（10-105 ページ）	テキスト・ファイルを XML 表の行として挿入します。正常に終了した場合は true を戻します。
isXMLTable()（10-106 ページ）	表が XML 表であるかどうかを確認します。XML 表である場合は true を戻します。
replaceXMLData()（10-106 ページ）	テキスト・ファイルを XML 表の行と置換します。正常に終了した場合は true を戻します。
xmlTableExists()（10-107 ページ）	XML 表が存在するかどうかを確認します。存在する場合は true を戻します。

DBAccess()

説明
クラス・コンストラクタです。

構文
```
public  DBAccess();
```

createBLOBTable()

説明
BLOB 表を作成します。正常に終了した場合は true を戻します。

構文
```
public boolean createBLOBTable( Connection con,
                                String tableName);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。

createXMLTable()

説明
XML 表を作成します。正常に終了した場合は true を戻します。

構文
```
public boolean createXMLTable( Connection con,
                               String tableName);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。

deleteBLOBName()

説明

BLOB 表からバイナリ・ファイルを削除します。正常に終了した場合は true を戻します。

構文

```
public boolean deleteBLOBName( java.sql.Connection con,
                               String tableName,
                               String blobName);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。
blobName	ファイル名。

deleteXMLName()

説明

XML 表からファイルを削除します。正常に終了した場合は true を戻します。

構文

```
public boolean deleteXMLName( java.sql.Connection con,
                              String tableName,
                              String xmlName);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。
xmlName	ファイル名。

dropBLOBTable()

説明

BLOB 表を削除します。正常に終了した場合は true を戻します。

構文

```
public boolean dropBLOBTable( java.sql.Connection con,
                              String tableName);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。

dropXMLTable()

説明

XML 表を削除します。正常に終了した場合は true を戻します。

構文

```
public boolean dropXMLTable( java.sql.Connection con,
                  java.lang.String tableName);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。

getBLOBData()

説明

BLOB 表からバイナリ・ファイルをバイナリ配列として取得します。正常に終了した場合は true を戻します。

構文

```
public byte[] getBLOBData( java.sql.Connection con,
                           String tableName,
                           String xmlName);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。
xmlName	ファイル名。

getNameSize()

説明

ファイル名が格納されているフィールドのサイズを戻します。

構文

```
public int getNameSize();
```

getXMLData()

説明

XML 表からテキスト・ファイルを文字列として取得します。

構文

```
public java.lang.String getXMLData( java.sql.Connection con,
                                    String tableName,
                                    String xmlName);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。
xmlName	ファイル名。

getXMLNames()

説明

XML 表のすべてのファイル名を文字列の配列として戻します。

構文

```
public java.lang.String[] getXMLNames( java.sql.Connection con,
                                       String tableName);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。

getXMLTableNames()

説明

ユーザー指定の文字列で始まるすべての XML 表名の配列を取得します。

構文

```
public java.lang.String[] getXMLTableNames( java.sql.Connection con,
                                            String tablePrefix);
```

パラメータ	説明
con	Connection オブジェクト。
tablePrefix	取得された XML 表名の配列の先頭にある表の接頭辞文字列。

insertBLOBData()

説明

バイナリ・ファイルを BLOB 表の行として挿入します。正常に終了した場合は true を戻します。

構文

```
public boolean insertBLOBData( java.sql.Connection con,
                               String tableName,
                               String xmlName,
                               byte[] xmlData);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。
xmlName	ファイル名。
xmlData	ファイル・データを含むバイト配列。

insertXMLData()

説明

テキスト・ファイルを XML 表の行として挿入します。正常に終了した場合は true を戻します。

構文

```
public boolean insertXMLData( java.sql.Connection con,
                              String tableName,
                              String xmlName,
                              String xmlData);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。
xmlName	ファイル名。
xmlData	ファイル・データを含む文字列。

isXMLTable()

説明

表が XML 表であるかどうかを確認します。XML 表である場合は `true` を戻します。

構文

```
public boolean isXMLTable( java.sql.Connection con,
                           String tableName);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。

replaceXMLData()

説明

テキスト・ファイルを XML 表の行と置換します。正常に終了した場合は `true` を戻します。

構文

```
public boolean replaceXMLData( java.sql.Connection con,
                               String tableName,
                               String xmlName,
                               String xmlData);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。
xmlName	ファイル名。
xmlData	ファイル・データを含む文字列。

xmlTableExists()

説明

XML 表が存在するかどうかを確認します。存在する場合は `true` を戻します。

構文

```
public boolean xmlTableExists( java.sql.Connection con,
                               String tableName);
```

パラメータ	説明
con	Connection オブジェクト。
tableName	表名。

DBAccessBeanInfo クラス

DBAccessBeanInfo の説明

このクラスは、DBAccess Bean に関する情報を提供します。

DBAccessBeanInfo の構文

```
public class DBAccessBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.transviewer.DBAccessBeanInfo
```

DBAccessBeanInfo の実装済インタフェース

```
java.beans.BeanInfo
```

DBAccessBeanInfo のメソッド

表 10-28 DBAccessBeanInfo のメソッドの概要

メソッド	説明
DBAccessBeanInfo()（10-108 ページ）	クラス・コンストラクタです。
getIcon()（10-109 ページ）	ツールバーやツールボックスなどで、DBAcceess Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。
getPropertyDescriptors()（10-109 ページ）	DBAccess Bean の編集可能な PropertyDescriptiors の配列を取得します。

DBAccessBeanInfo()

説明

クラス・コンストラクタです。

構文

```
public  DBAccessBeanInfo();
```

getIcon()

説明

ツールバーやツールボックスなどで、DBAcceess Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。java.beans.SimpleBeanInfo クラスの getIcon() をオーバーライドします。

構文

```
public java.awt.Image getIcon(int iconKind);
```

パラメータ	説明
iconKind	リクエストされたアイコンの種類。

getPropertyDescriptors()

説明

DBAccess Bean の編集可能な PropertyDescriptiors の配列を取得します。java.beans.SimpleBeanInfo クラスの getPropertyDescriptors() をオーバーライドします。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

XMLTransformPanel クラス

XMLTransformPanel の説明

このクラスは可視 Bean です。XSL 変換を XML 文書へ適用し、結果を可視化します。これによって、入力した XML および XSL の文書やファイルが編集可能になります。

XMLTransformPanel の構文

```
public class XMLTransformPanel extends javax.swing.JPanel

java.lang.Object
  |
  +--java.awt.Component
     |
     +--java.awt.Container
        |
        +--javax.swing.JComponent
           |
           +--javax.swing.JPanel
              |
              +--oracle.xml.transviewer.XMLTransformPanel
```

XMLTransformPanel の実装済インタフェース

- javax.accessibility.Accessible
- java.awt.image.ImageObserver
- java.awt.MenuContainer
- java.io.Serializable

XMLTransformPanel のメソッド

XMLTransformPanel()

説明

クラス・コンストラクタです。XMLTransformPanel 型のオブジェクトを作成します。

構文

```
public  XMLTransformPanel();
```

XMLTransformPanelBeanInfo クラス

XMLTransformPanelBeanInfo の説明

このクラスは、XMLTransformPanel Bean に関する情報を提供します。

XMLTransformPanelBeanInfo の構文

```
public class XMLTransformPanelBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.transviewer.XMLTransformPanelBeanInfo
```

XMLTransformerBeanInfo の実装済インタフェース

```
java.beans.BeanInfo
```

XMLTransformPanelBeanInfo のメソッド

表 10-29 XMLTransformPanelBeanInfo のメソッドの概要

メソッド	説明
XMLTransformPanelBeanInfo()（10-111 ページ）	クラス・コンストラクタです。
getIcon()（10-112 ページ）	ツールバーやツールボックスなどで、XMLTransformPanel Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。
getPropertyDescriptors()（10-112 ページ）	XMLTransformPanel Bean の編集可能な PropertyDescriptiors の配列を取得します。

XMLTransformPanelBeanInfo()

説明

クラス・コンストラクタです。

構文

```
public  XMLTransformPanelBeanInfo();
```

getIcon()

説明

ツールバーやツールボックスなどで、XMLTransformPanel Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。java.beans.SimpleBeanInfo クラスの getIcon() をオーバーライドします。

構文

```
public java.awt.Image getIcon( int iconKind);
```

パラメータ	説明
iconKind	リクエストされたアイコンの種類。

getPropertyDescriptors()

説明

XMLTransformPanel Bean の編集可能な PropertyDescriptiors の配列を取得します。java.beans.SimpleBeanInfo クラスの getPropertyDescriptors() をオーバーライドします。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

XMLTransViewer クラス

XMLTransViewer の説明

このクラスは、XMLTransformPanel を使用する簡単なアプリケーションです。これはコマンドラインから使用でき、XML ファイルの編集および解析、XSL 変換の編集および適用、ファイル・システムまたはデータベース内の XML、XSL および結果ファイルの取得および保存を実行できます。

XMLTransViewer の構文

```
public class XMLTransViewer extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.transviewer.XMLTransViewer
```

XMLTransViewer のメソッド

表 10-30 XMLTransViewer のメソッドの概要

メソッド	説明
XMLTransViewer()（10-113 ページ）	クラス・コンストラクタです。
getReleaseVersion()（10-114 ページ）	Oracle XML Transviewer のバージョン番号を文字列として戻します。
main()（10-114 ページ）	XMLTransViewer の実行を開始します。

XMLTransViewer()

説明

クラス・コンストラクタです。

構文

```
public  XMLTransViewer();
```

getReleaseVersion()

説明

Oracle XML Transviewer のバージョン番号を文字列として戻します。

構文

```
public static java.lang.String getReleaseVersion();
```

main()

説明

XMLTransViewer の実行を開始します。

構文

```
public static void main( String[] args);
```

パラメータ	説明
args	XMLTransViewer クラスの実行のための引数。

oracle.xml.treeviewer パッケージ

oracle.xml.treeviewer の説明

このパッケージは、可視 Bean です。XML 文書をツリーとして表示し、XML 文書の DOM ツリー構造を示します。ユーザーは、ノードを縮小または拡張できます。

表 10-31 に、`oracle.xml.treeviewer` のクラスの概要を示します。

表 10-31 oracle.xml.treeviewer のクラスの概要

クラス	説明
XMLTreeView クラス（10-116 ページ）	XML 文書をツリーとして表示します。
XMLTreeViewBeanInfo クラス（10-120 ページ）	XMLTreeView Bean に関する情報を提供します。

XMLTreeView クラス

XMLTreeView の説明

このクラスは、XML 文書をツリーとして表示します。タグ、属性名、属性値、コメント、CDATA、PCDATA、処理命令（PI）データ、処理命令（PI）名および表記法の XML DOM ノードを認識します。この Bean は、入力として `org.w3c.dom.Document` オブジェクトを取ります。

XMLTreeView の構文

```
public class XMLTreeView extends javax.swing.JPanel

java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--oracle.xml.treeviewer.XMLTreeView
```

XMLTreeView の実装済インタフェース

- `javax.accessibility.Accessible`
- `java.awt.image.ImageObserver`
- `java.awt.MenuContainer`
- `java.io.Serializable`

XMLTreeView のフィールド

表 10-32 XMLTreeView のフィールド

フィールド	構文	説明
model	protected oracle.xml.treeviewer.XMLTreeModel model	DOM ノードからの情報を使用する JTree 用のデータ・モデル。
scrollPane	protected transient javax.swing.JScrollPane scrollPane	ツリーの表示に使用されるコンテナ。上下左右のスクロールをサポートします。
theTree	protected transient javax.swing.JTree theTree	DOM ツリーの様々なノードをレンダリングする Java コンポーネント。

XMLTreeView のメソッド

表 10-33 XMLTreeView のメソッドの概要

メソッド	説明
XMLTreeView()（10-117 ページ）	クラス・コンストラクタです。
getPreferredSize()（10-118 ページ）	XMLTreeView の推奨サイズを戻します。
getTree()（10-118 ページ）	可視ツリーを JTree として戻します。
getXMLTreeModel()（10-118 ページ）	JTree 用のデータ・モデルを XMLTreeModel として戻します。
setXMLDocument()（10-118 ページ）	XMLTreeViewer を XML 文書に対応付けます。
updateUI()（10-119 ページ）	XMLTreeView に UI 画面を更新 / 再描画させます。

XMLTreeView()

説明

クラス・コンストラクタです。`XMLTreeView` 型のオブジェクトを作成します。

構文

```
public  XMLTreeView();
```

getPreferredSize()

説明

XMLTreeView の推奨サイズを含む Dimension オブジェクトを戻します。
javax.swing.JComponent クラスの getPreferredSize() をオーバーライドします。

構文

```
public java.awt.Dimension getPreferredSize();
```

getTree()

説明

可視ツリーを JTree として戻します。

構文

```
protected javax.swing.JTree getTree();
```

getXMLTreeModel()

説明

JTree 用のデータ・モデルを XMLTreeModel として戻します。

構文

```
protected oracle.xml.treeviewer.XMLTreeModel getXMLTreeModel();
```

setXMLDocument()

説明

XML TreeViewer を XML 文書に対応付けます。

構文

```
public void setXMLDocument( org.w3c.dom.Document document);
```

パラメータ	説明
document	表示するドキュメント。

updateUI()

説明

XML TreeView に UI 画面を更新 / 再描画させます。

構文

```
public void updateUI();
```

XMLTreeViewBeanInfo クラス

XMLTreeViewBeanInfo の説明

このクラスは、XMLTreeView Bean に関する情報を提供します。

XMLTreeViewBeanInfo の構文

```
public class XMLTreeViewBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.treeviewer.XMLTreeViewBeanInfo
```

XMLTreeViewBeanInfo の実装済インタフェース

```
java.beans.BeanInfo
```

XMLTreeViewBeanInfo のメソッド

表 10-34 XMLTreeViewBeanInfo のメソッドの概要

メソッド	説明
XMLTreeViewBeanInfo() (10-120 ページ)	クラス・コンストラクタです。
getIcon() (10-121 ページ)	ツールバーやツールボックスなどで、XMLTreeView Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。
getPropertyDescriptors() (10-121 ページ)	XMLTreeView Bean の編集可能な PropertyDescriptiors の配列を取得します。

XMLTreeViewBeanInfo()

説明

クラス・コンストラクタです。

構文

```
public  XMLTreeViewBeanInfo();
```

getIcon()

説明

ツールバーやツールボックスなどで、XMLTreeView Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。

構文

```
public java.awt.Image getIcon( int iconKind);
```

パラメータ	説明
iconKind	リクエストされたアイコンの種類。

getPropertyDescriptors()

説明

XMLTreeView Bean の編集可能な PropertyDescriptiors の配列を取得します。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

oracle.xml.differ パッケージ

oracle.xml.differ の説明

このパッケージは、可視デモを使用した不可視 Bean です。この Bean によって、2 つの XML 文書を比較して、差異を XSLT コードとして生成できます。XSLT を最初のファイルに適用して、2 つ目のファイルに変換できます。可視デモを使用すると、ユーザーは 2 つの XML 文書の差異をグラフィカルに参照できます。

表 10-35 に、`oracle.xml.differ` のクラスの概要を示します。

表 10-35 oracle.xml.differ のクラスの概要

クラス	説明
XMLDiff クラス（10-123 ページ）	2 つの XML ファイルを比較するためのインタフェースを定義します。
XMLDiffBeanInfo クラス（10-134 ページ）	XMLDiff Bean に関する情報を提供します。

XMLDiff クラス

XMLDiff の説明

このクラスは、2 つの XML ファイルを比較するためのインタフェースを定義します。これによって、比較する 2 つの XML ファイルが等しいかどうかを確認できます。差異がある場合、その差異をグラフィカルに参照できるオブジェクトが提供されます。差異は、XSL として表すこともできます。対応する差異を示した XSL スタイルシートは、ファイルまたは XMLDocument オブジェクトとして生成できます。生成された XSL スタイルシートを使用して、最初の XML ファイルを 2 つ目のファイルに変換できます。

XMLDiff の構文

```
oracle.xml.differ

java.lang.Object
  |
  +--oracle.xml.differ.XMLDiff
```

XMLDiff の実装済インタフェース

- DOMBuilderListener インタフェース
- DOMBuilderErrorListener インタフェース
- java.io.Serializablen インタフェース

XMLDiff のメソッド

表 10-36 XMLDiff のメソッドの概要

メソッド	説明
XMLDiff()（10-125 ページ）	クラス・コンストラクタです。
setFiles()（10-125 ページ）	比較する必要がある XML ファイルを設定します。
setDocuments()（10-126 ページ）	比較する必要がある XML 文書を設定します。
setInput1()（10-126 ページ）	比較する必要がある最初の XML ファイルまたは XML 文書を設定します。
setInput2()（10-127 ページ）	比較する必要がある 2 つ目の XML ファイルまたは XML 文書を設定します。
getDocument1()（10-128 ページ）	ドキュメント・ルートを最初の XML ツリーの XMLDocument オブジェクトとして取得します。

表 10-36 XMLDiff のメソッドの概要（続き）

メソッド	説明
getDocument2()（10-128 ページ）	ドキュメント・ルートを 2 つ目の XML ツリーの XMLDocument オブジェクトとして取得します。
diff()（10-128 ページ）	2 つの XML ファイルまたは 2 つの XMLDocument オブジェクトの差異を検索します。ビジュアル・テキスト・パネルを JTextPane オブジェクトとして取得します。このオブジェクトは、最初の XML ファイルまたは XML 文書の差異をビジュアルに表示します。
getDiffPane1()（10-128 ページ）	ビジュアル・テキスト・パネルを JTextPane オブジェクトとして取得します。このオブジェクトは、最初の XML ファイルまたは XML 文書の差異をビジュアルに表示します。
getDiffPane2()（10-129 ページ）	ビジュアル・テキスト・パネルを JTextPane オブジェクトとして取得します。このオブジェクトは、2 つ目の XML ファイルまたは XML 文書の差異をビジュアルに表示します。
setIndentIncr()（10-129 ページ）	XSL 生成時のインデントを設定します。
setNewNodeIndentIncr()（10-129 ページ）	XSL 生成時のインデントを設定します。
generateXSLFile()（10-130 ページ）	最初に設定された 2 つの XML ファイルの差異を表す XSL ファイルをユーザー定義のファイル名で生成します。
generateXSLDOC()（10-130 ページ）	最初に設定された 2 つの XML 文書の差異を表す XSL スタイルシートを XML 文書として生成します。
equals()（10-131 ページ）	2 つのノードを比較します。このメソッドは、differ アルゴリズムによってコールされます。必要に応じて、このファンクションを上書きして、カスタマイズした比較を実行できます。
domBuilderErrorCalled()（10-131 ページ）	DOMBuilderErrorListener インタフェースを実装するメソッドで、解析中にエラーが発生したときに、DOM パーサーによってのみコールされます。
domBuilderError()（10-131 ページ）	DOMBuilderErrorListener インタフェースを実装するメソッドで、DOM パーサーによってのみコールされます。
domBuilderOver()（10-132 ページ）	DOMBuilderListener インタフェースを実装するメソッドで、解析の完了時に DOM パーサー・スレッドによってのみコールされます。
domBuilderStarted()（10-132 ページ）	DOMBuilderListener インタフェースを実装するメソッドで、解析の開始時に DOM パーサーによってのみコールされます。

表 10-36 XMLDiff のメソッドの概要（続き）

メソッド	説明
printDiffTree()（10-133 ページ）	差異を示すツリーを出力します。このツリーには、アルゴリズムによって差異と識別されたノード名および値が含まれます。これは、デバッグに便利です。
setNoMoves()（10-133 ページ）	diff アルゴリズムによって検出される差異が存在しないと想定します。このファンクションは、diff() ファンクションの前にコールする必要があります。これによって、パフォーマンスが向上します。

XMLDiff()

説明

クラス・コンストラクタです。

構文

```
public XMLDiff();
```

setFiles()

説明

比較する必要がある XML ファイルを設定します。両方のファイルが DOM ツリーに解析され、比較されます。この方法は、setInput1() および setInput2() をコールする方法より高速です。次の例外が発生します。

- `java.io.IOException`（I/O エラーが発生した場合）
- `XMLParseException`（XML 文書の解析時）
- `SAXException`（XML 文書の解析時）
- `java.lang.InterruptedException`（スリープ中のスレッドが中断された場合）

構文

```
public void setFiles( java.io.File file1,
                      java.io.File file2);
```

パラメータ	説明
file1	最初の XML ファイル。
file2	2 つ目の XML ファイル。

setDocuments()

説明

比較する必要がある XML 文書を設定します。

構文

```
public void setDocuments( XMLDocument doc1,
                          XMLDocument doc2);
```

パラメータ	説明
doc1	最初の XML 文書。
doc2	2 つ目の XML 文書。

setInput1()

説明

比較する必要がある最初の XML ファイルまたは XML 文書を設定します。入力ファイルが DOM ツリーに解析され、比較されます。次の例外が発生します。

- `java.io.IOException`（I/O エラーが発生した場合）
- `XMLParseException`（XML 文書の解析時）
- `SAXException`（XML 文書の解析時）
- `java.lang.InterruptedException`（スリープ中のスレッドが中断された場合）

次の表に、オプションを示します。

構文	説明
public void setInput1(File file1);	比較する必要がある最初の XML ファイルを設定します。

構文	説明
public void setInput1(XMLDocument doc1);	比較する必要がある最初の XML 文書を設定します。

パラメータ	説明
file1	最初の XML ファイル。
doc1	最初の XML 文書。

setInput2()

説明

比較する必要がある 2 つ目の XML ファイルまたは XML 文書を設定します。入力ファイルが DOM ツリーに解析され、比較されます。次の例外が発生します。

- `java.io.IOException`（I/O エラーが発生した場合）
- `XMLParseException`（XML 文書の解析時）
- `SAXException`（XML 文書の解析時）
- `java.lang.InterruptedException`（スリープ中のスレッドが中断された場合）

次の表に、オプションを示します。

構文	説明
public void setInput2(File file2);	比較する必要がある 2 つ目の XML ファイルを設定します。
public void setInput2(XMLDocument doc2);	比較する必要がある 2 つ目の XML 文書を設定します。

パラメータ	説明
file2	2 つ目の XML ファイル。
doc2	2 つ目の XML 文書。

getDocument1()

説明

ドキュメント・ルートを最初の XML ツリーの XMLDocument オブジェクトとして取得します。

構文

```
public XMLDocument getDocument1();
```

getDocument2()

説明

ドキュメント・ルートを 2 つ目の XML ツリーの XMLDocument オブジェクトとして取得します。

構文

```
public XMLDocument getDocument2();
```

diff()

説明

2 つの XML ファイルまたは 2 つの XMLDocument オブジェクトの差異を検索します。XML ファイルまたは XML 文書が同じ場合は `false`、異なる場合は `true` を戻します。XML ファイルが正常に解析されない場合、または XML 文書が設定されていない場合は、`java.lang.NullPointerException` が発生します。

構文

```
public boolean diff();
```

getDiffPane1()

説明

ビジュアル・テキスト・パネルを JTextPane オブジェクトとして取得します。このオブジェクトは、最初の XML ファイルまたは XML 文書の差異をビジュアルに表示します。

構文

```
public javax.swing.JTextPane getDiffPane1();
```

getDiffPane2()

説明

ビジュアル・テキスト・パネルを JTextPane オブジェクトとして取得します。このオブジェクトは、2 つ目の XML ファイルまたは XML 文書の差異をビジュアルに表示します。

構文

```
public javax.swing.JTextPane getDiffPane2();
```

setIndentIncr()

説明

XSL 生成時のインデントを設定します。このメソッドは、generateXSLFile() または generateXSLDoc() の前にコールする必要があります。インデントは、すべての属性にのみ適用されます。新しく挿入されたノードのインデントについては、setNewNodeIndentIncr() を参照してください。

構文

```
public void setIndentIncr( int spaces);
```

パラメータ	説明
spaces	属性用のインデントの増分。空白の数を指定します。

setNewNodeIndentIncr()

説明

XSL 生成時のインデントを設定します。このメソッドは、generateXSLFile() または generateXSLDoc() の前にコールする必要があります。インデントは、新しく挿入されたすべてのノードにのみ適用されます。属性のインデント・サポートについては、setIndentIncr() を参照してください。

構文

```
public void setNewNodeIndentIncr( int spaces);
```

パラメータ	説明
spaces	新しいノード用のインデントの増分。空白の数を指定します。

generateXSLFile()

説明

最初に設定された 2 つの XML ファイルの差異を表す XSL ファイルをユーザー定義のファイル名で生成します。入力ファイル名が null である場合、XMLDiff.xsl という名前のデフォルトの XSL ファイルが生成されます。生成された XSL スタイルシートを使用して、最初の XML ファイルを 2 つ目のファイルに変換できます。XML が同じ場合、生成された XSL が最初の XML ファイルを 2 つ目の XML ファイルに変換して、2 つのファイルを同じにします。XSL ファイルが正常に作成されない場合は、java.io.IOException が発生します。

構文

```
public void generateXSLFile( String fileName);
```

パラメータ	説明
fileName	出力 XSL のファイル名。

generateXSLDOC()

説明

最初に設定された 2 つの XML 文書の差異を表す XSL スタイルシートを XML 文書として生成します。入力ファイル名が null である場合、XMLDiff.xsl という名前のデフォルトの XSL ファイルが生成されます。生成された XSL スタイルシートを使用して、最初の XML ファイルを 2 つ目のファイルに変換できます。XML が同じ場合、生成された XSL が最初の XML ファイルを 2 つ目の XML ファイルに変換して、2 つのファイルを同じにします。次の例外が発生します。

- java.io.IOException（I/O エラーが発生した場合）
- java.io.FileNotFoundException（生成された XSL ファイルが検出されない場合）
- SAXException（XML 文書の解析時）
- java.lang.InterruptedException（スリープ中のスレッドが中断された場合）

構文

```
public void generateXSLDoc();
```

equals()

説明

2 つのノードを比較します。このメソッドは、differ アルゴリズムによってコールされます。必要に応じて、このファンクションを上書きして、カスタマイズした比較を実行できます。

構文

```
protected boolean equals( Node node1,
                          Node node2);
```

パラメータ	説明
node1	比較する最初のノード。
node2	比較する 2 つ目のノード。

domBuilderErrorCalled()

説明

DOMBuilderErrorListener インタフェースを実装するメソッドで、解析中にエラーが発生したときに、DOM パーサーによってのみコールされます。DOMBuilderErrorListener インタフェースの domBuilderErrorCalled によって指定されます。

構文

```
public void domBuilderErrorCalled( DOMBuilderErrorEvent p0);
```

パラメータ	説明
p0	パーサーによって発生するエラー・オブジェクト。

domBuilderError()

説明

DOMBuilderErrorListener インタフェースを実装するメソッドで、DOM パーサーによってのみコールされます。DOMBuilderListener インタフェースの domBuilderError によって指定されます。

構文

```
public void domBuilderError( DOMBuilderEvent p0);
```

パラメータ	説明
p0	domBuilderErrorCalled() によって処理されるパーサー・イベント・エラー。

domBuilderOver()

説明

DOMBuilderListener インタフェースを実装するメソッドで、解析の完了時に DOM パーサー・スレッドによってのみコールされます。DOMBuilderListener インタフェースの domBuilderOver によって指定されます。

構文

```
public void domBuilderOver ( DOMBuilderEvent p0);
```

パラメータ	説明
p0	パーサー・イベント。

domBuilderStarted()

説明

DOMBuilderListener インタフェースを実装するメソッドで、解析の開始時に DOM パーサーによってのみコールされます。DOMBuilderListener インタフェースの domBuilderStarted によって指定されます。

構文

```
public void domBuilderStarted( DOMBuilderEvent p0);
```

パラメータ	説明
p0	パーサー・イベント。

printDiffTree()

説明

diff ツリーを出力します。このツリーには、アルゴリズムによって差異と識別されたノード名および値が含まれます。これは、デバッグに便利です。XSL ファイルが正常に作成されない場合は、java.io.IOException が発生します。

構文

```
public void printDiffTree( int tree,
                           BufferedWriter out);
```

パラメータ	説明
tree	出力するツリー（1 または 2）。
out	出力された diff ツリーを含む BufferredWriter。

setNoMoves()

説明

diff アルゴリズムによって検出される差異が存在しないと想定します。このファンクションは、diff() ファンクションの前にコールする必要があります。これによって、パフォーマンスが向上します。

構文

```
public void setNoMoves();
```

XMLDiffBeanInfo クラス

XMLDiffBeanInfo の説明

このクラスは、XMLDiff Bean に関する情報を提供します。

XMLDiffBeanInfo の構文

```
public class XMLDiffBeanInfo extends java.beans.SimpleBeanInfo

XMLSjava.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.differ.XMLDiffBeanInfo
```

XMLDiffBeanInfo のメソッド

表 10-37 XMLDiffBeanInfo のメソッドの概要

メソッド	説明
XMLDiffBeanInfo()（10-134 ページ）	クラス・コンストラクタです。
getPropertyDescriptors()（10-135 ページ）	ツールバーやツールボックスなどで、XMLDiff Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。
getIcon()（10-135 ページ）	XMLDiff Bean の編集可能な PropertyDescriptiors の配列を取得します。

XMLDiffBeanInfo()

説明

クラス・コンストラクタです。

構文

```
public XMLDiffBeanInfo();
```

getPropertyDescriptors()

説明

XMLDiff Bean の編集可能な PropertyDescriptiors の配列を取得します。
java.beans.SimpleBeanInfo クラスの getPropertyDescriptors() をオーバーライドします。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

getIcon()

説明

ツールバーやツールボックスなどで、XMLDiff Bean 用にリクエストされたアイコンの種類を表すイメージ・オブジェクトを取得します。java.beans.SimpleBeanInfo クラスの getIcon() をオーバーライドします。

構文

```
public java.awt.Image getIcon( int iconKind);
```

パラメータ	説明
iconKind	リクエストされたアイコンの種類。

11

Simple Object Access Protocol（SOAP）for Java

Oracle SOAP は、Simple Object Access Protocol の実装です。Oracle SOAP は、Apache Software Foundation によって開発された SOAP オープン・ソース実装に基づいています。

SOAP は、インターネット経由でリクエストおよびレスポンスを送受信するための転送プロトコルで、XML および HTTP に基づいています。SOAP は、転送プロトコルおよびオペレーティング・システムに依存しません。SOAP は、すべてのアプリケーションに、標準の XML メッセージ・フォーマットを提供します。SOAP は、World Wide Web Consortium（W3C）の XML Schema 標準を使用します。

Oracle SOAP API は、次のパッケージおよびクラスに含まれています。

この章の内容は次のとおりです。

参照： 次の URL またはマニュアルを参照してください。

- `http://www.w3.org/TR/SOAP/`
- `http://xml.apache.org/soap/`
- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

oracle.soap.server パッケージ

oracle.soap.server の説明

このパッケージには、SOAP 管理クライアント用の API を実装するインタフェースとクラス、および Java クラス用のプロバイダ実装が含まれています。これらには、サービス・マネージャおよびプロバイダ・マネージャが含まれます。これらの管理クライアントは、新しいサービスおよび新しいプロバイダの動的なデプロイをサポートするサービスです。

表 11-1 に、XDK for Java で Oracle SOAP をサポートするインタフェースおよびクラスを示します。

表 11-1 oracle.soap.server のクラスおよびインタフェースの概要

クラス / インタフェース	説明
Handler インタフェース（11-3 ページ）	SOAP サーバーの交換可能なハンドラの作成のためのインタフェースを定義します。
Provider インタフェース（11-7 ページ）	サービス・プロバイダのタイプごとにサポートが必要な機能を定義します。
ProviderManager インタフェース（11-10 ページ）	プロバイダのデプロイ、プロバイダのアンデプロイおよびプロバイダのデプロイ情報へのアクセスを行うために SOAP エンジンが使用するプロバイダ・マネージャを定義します。
ServiceManager インタフェース（11-14 ページ）	サービスのデプロイ、サービスのアンデプロイおよびサービスのデプロイ情報へのアクセスを行うために SOAP エンジンが使用するサービス・マネージャを定義します。
ContainerContext クラス（11-18 ページ）	SOAP サーバーが実行されているコンテナのコンテキストを定義します。
Logger クラス（11-23 ページ）	ロガーの実装の際にサポートされる必要がある機能を定義します。ロガーは、エラー・メッセージおよび情報メッセージを永続的に記録するために使用されます。
ProviderDeploymentDescriptor クラス（11-29 ページ）	特定のプロバイダに関するデプロイ情報を定義します。
RequestContext クラス（11-34 ページ）	SOAP リクエストに対するすべてのコンテキストを定義します。これには、プロバイダに渡される情報、および戻す前にプロバイダで設定する必要がある情報が含まれます。
ServiceDeploymentDescriptor クラス（11-42 ページ）	プロバイダのタイプに関係なく、SOAP サービスのデプロイ情報を定義します。
UserContext クラス（11-58 ページ）	SOAP サービス・リクエストのユーザー・コンテキストを定義します。

Handler インタフェース

Handler の説明

SOAP サーバーの交換可能なハンドラの作成のためのインタフェースを定義します。このクラスは、ハンドラの起動時期に関するポリシーは示しません。

ハンドラの実装は、次の条件を満たす必要があります。

- 引数なしのコンストラクタを提供する。
- スレッド・セーフである。

Handler の構文

```
oracle.soap.server.Handler

public interface Handler
```

Handler のフィールド

表 11-2 Handler のフィールド

フィールド	構文	説明
REQUEST_TYPE	public static final int REQUEST_TYPE	ハンドラの起動は、リクエスト連鎖の一部です。
RESPONSE_TYPE	public static final int RESPONSE_TYPE	ハンドラの起動は、レスポンス連鎖の一部です。
ERROR_TYPE	public static final int ERROR_TYPE	ハンドラの起動は、エラー連鎖の一部です。

Handler のメソッド

表 11-3 Handler のメソッドの概要

メソッド	説明
destroy()（11-4 ページ）	ハンドラをクリーンアップします（1 回のみ）。
getName()（11-4 ページ）	ハンドラの名前を戻します。
getOptions()（11-4 ページ）	ハンドラ実装固有のオプションを戻します。
init()（11-5 ページ）	ハンドラを初期化します（1 回のみ）。

表 11-3 Handler のメソッドの概要（続き）

メソッド	説明
invoke()（11-5 ページ）	リクエスト・ハンドラを、指定された連鎖タイプの一部として起動します。
setName()（11-6 ページ）	ハンドラの名前を設定します。このメソッドは、init() の前にコールする必要があります。
setOptions()（11-6 ページ）	init が後で使用するハンドラのオプションを設定します。

destroy()

説明

ハンドラをクリーンアップします（1 回のみ）。このメソッドは、サーバーが停止する前に、SOAP サーバーによって 1 回のみ起動されます。これによって、ハンドラはグローバル状態をクリーンアップできます。廃棄できない場合は、SOAPException が発生します。

構文

```
public abstract void destroy();
```

getName()

説明

ハンドラの名前を戻します。

構文

```
public abstract String getName();
```

getOptions()

説明

ハンドラ実装固有のオプションを戻します。

構文

```
public abstract Properties getOptions();
```

init()

説明

ハンドラを初期化します（1 回のみ）。このメソッドは、SOAP サーバーがハンドラになんらかの起動操作を行う前にこのサーバーによって 1 回のみ起動され、ハンドラは任意のグローバル状態を設定できるようになります。このメソッドは、setOptions() で事前に設定されたオプションを使用します。ハンドラを初期化できない場合は、SOAPException が発生します。

構文

```
public abstract void init( SOAPServerContext ssc);
```

パラメータ	説明
ssc	情報メッセージ用のロガーを含む SOAP サーバー・コンテキスト。

invoke()

説明

リクエスト・ハンドラを、指定された連鎖タイプの一部として起動します。リクエスト・ハンドラまたはレスポンス・ハンドラの連鎖の実行は、いずれかのハンドラで SOAPException が発生した直後に終了することに注意してください。ただし、エラー連鎖内のすべてのハンドラは、いずれかのハンドラでの例外の発生にかかわらず起動されます。エラー・ハンドラで例外が発生した場合、例外は記録され、廃棄されます。ハンドラの起動に失敗した場合は、SOAPException が発生します。

構文

```
public abstract void invoke( int chainType,
                             RequestContext requestContext);
```

パラメータ	説明
chainType	次の連鎖タイプがサポートされています。 ■ Handler.REQUEST_TYPE: サービスの起動前にハンドラがリクエスト連鎖の一部として起動されている場合 ■ Handler.RESPONSE_TYPE: サービスの起動後にハンドラがレスポンス連鎖の一部として起動されている場合 ■ Handler.ERROR_TYPE: リクエスト連鎖、サービスの起動またはレスポンス連鎖のいずれかでエラーが発生した場合に、ハンドラがエラー連鎖の一部として起動されている場合
requestContext	関連するリクエスト・コンテキスト。

setName()

説明

ハンドラの名前を設定します。このメソッドは、init() の前にコールする必要があります。

構文

```
public abstract void setName( String name);
```

パラメータ	説明
name	ハンドラ・インスタンスの名前。

setOptions()

説明

init が後で使用するハンドラのオプションを設定します。このメソッドは、init() の前にコールする必要があります。

構文

```
public abstract void setOptions( Properties options);
```

パラメータ	説明
options	ハンドラ実装固有のオプション。

Provider インタフェース

Provider の説明

このインタフェースは、Java クラスやストアド・プロシージャなど、サービス・プロバイダのタイプごとにサポートが必要な機能を定義します。サービス認可およびパラメータのアンマーシャリングやマーシャリングは、プロバイダが行います。

プロバイダ（プロバイダ・インスタンスともいう）は、SOAP ハンドラにデプロイする必要があります。各プロバイダのデプロイでは、プロバイダ名、プロバイダを実装する Java クラス名（このインタフェースの実装である必要がある）および任意の数のプロバイダ固有のキーと値の組を定義する必要があります。指定されたプロバイダのデプロイ情報によって、SOAP ハンドラはこのインタフェースのみを介してプロバイダと対話します。

SOAP ハンドラは、デプロイされるプロバイダ・インスタンスごとに 1 つのインスタンスを作成します。各プロバイダの実装に 1 つ以上のインスタンスを持つことができます（ただし、お薦めはしません）。どのような場合も、プロバイダの各インスタンスがリクエストを同時に処理できる必要があります。

プロバイダの実装は、次の条件を満たす必要があります。

- 引数なしのコンストラクタを提供する。
- スレッド・セーフである。

Provider の構文

```
public interface Provider

Interface oracle.soap.server.Provider
```

Provider のメソッド

表 11-4 Provider のメソッドの概要

メソッド	説明
destroy()（11-8 ページ）	プロバイダ・インスタンスをクリーンアップします（1 回のみ）。
getId()（11-8 ページ）	プロバイダの名前を戻します。この名前は、SOAP ハンドラ内で一意です。
init()（11-8 ページ）	プロバイダ・インスタンスを初期化します（1 回のみ）。
invoke()（11-9 ページ）	指定されたサービスで、リクエストされたメソッドをコールします。この場合、SOAP リクエストはリクエスト・コンテキストで完全に記述されています。

destroy()

説明

プロバイダ・インスタンスをクリーンアップします（1 回のみ）。このメソッドは、ハンドラが停止する前に、SOAP ハンドラによって 1 回のみ起動されます。これによって、プロバイダはプロバイダのグローバルな状態をクリーンアップできます。廃棄できない場合は、SOAPException が発生します。

構文

```
public abstract void destroy();
```

getId()

説明

プロバイダの名前を戻します。この名前は、SOAP ハンドラ内で一意です。

構文

```
public abstract String getId();
```

init()

説明

プロバイダ・インスタンスを初期化します（1 回のみ）。このメソッドは、プロバイダがサポートするサービスに SOAP ハンドラがなんらかのリクエストを行う前に、ハンドラによって 1 回のみ起動され、プロバイダが任意のプロバイダのグローバル・コンテキストを設定できるようになります。サービスを初期化できないためにサービスを提供できない場合は、SOAPException が発生します。

構文

```
public abstract void init( ProviderDeploymentDescriptor pd,
                           SOAPServerContext ssc);
```

パラメータ	説明
pd	プロバイダのデプロイ情報を含むプロバイダ・ディスクリプタ。
ssc	情報メッセージ用のロガーを含む SOAP サーバー・コンテキスト。

invoke()

説明

指定されたサービスで、リクエストされたメソッドをコールします。この場合、SOAP リクエストはリクエスト・コンテキストで完全に記述されています。ユーザーが権限を所有していない、メソッドが存在しないなどの原因（数は不問）によってメソッドの起動中にエラーが発生した場合は、SOAPException が発生します。

構文

```
public abstract void invoke( RequestContext requestContext);
```

パラメータ	説明
requestContext	プロバイダがリクエストの処理に必要とするものすべてを含むリクエスト・コンテキスト。

ProviderManager インタフェース

ProviderManager の説明

このインタフェースは、プロバイダを管理するインタフェースを定義します。プロバイダ・マネージャは、プロバイダのデプロイ、プロバイダのアンデプロイおよびプロバイダのデプロイ情報へのアクセスを行うために SOAP エンジンによって使用されます。プロバイダ・マネージャは、デプロイ情報をキャッシュし、キャッシュをメンテナンスします。

HTTP サーバーは、プロバイダ・マネージャにセキュリティを提供します。プロバイダ・マネージャは、どのリクエストが受け入れられるのかを、URL によって構成できます。プロバイダ・マネージャに対する SOAP リクエストが他の URL に行われた場合、そのリクエストは拒否されます。この URL は SOAP サーブレットの別名にする必要があり、HTTP セキュリティを設定してこの URL にポスト可能なユーザーを制御できます。

ProviderManager の構文

```
public interface ProviderManager
```

```
Interface oracle.soap.server.ProviderManager
```

ProviderManager のメソッド

表 11-5 ProviderManager のメソッドの概要

メソッド	説明
deploy()（11-11 ページ）	指定されたプロバイダをデプロイします。
destroy()（11-11 ページ）	プロバイダ・マネージャをクリーンアップします。
getRequiredRequestURI()（11-11 ページ）	プロバイダ・マネージャがリクエストする URI を戻します。
init()（11-12 ページ）	プロバイダ・マネージャを初期化します。
list()（11-12 ページ）	デプロイ済のすべてのプロバイダについて、プロバイダ ID の配列を戻します。
query()（11-12 ページ）	指定されたプロバイダのデプロイメント・ディスクリプタを戻します。
setServiceManager()（11-13 ページ）	サービス・マネージャをプロバイダ・マネージャが使用できるようにします。
undeploy()（11-13 ページ）	指定されたプロバイダをアンデプロイします。

deploy()

説明

指定されたプロバイダをデプロイします。デプロイできない場合は、SOAPException が発生します。

構文

```
public abstract void deploy(ProviderDeploymentDescriptor providerId);
```

パラメータ	説明
providerId	デプロイするプロバイダの ID。

destroy()

説明

プロバイダ・マネージャをクリーンアップします。プロバイダ・マネージャをクリーンアップできない場合は、SOAPException が発生します。

構文

```
public abstract void destroy();
```

getRequiredRequestURI()

説明

プロバイダ・マネージャがリクエストする URI、またはすべての URI を使用できる場合は null を戻します。リクエストが受け入れられるためには、この URI にリクエストを行う必要があります。他の URI に対して行われたリクエストは拒否されます。

構文

```
public abstract String getRequiredRequestURI();
```

init()

説明

プロバイダ・マネージャを初期化します。デプロイ情報にアクセスできない場合は、SOAPException が発生します。

構文

```
public abstract void init(Properties options);
```

パラメータ	説明
options	デプロイ情報へのアクセスを設定するために必要なオプション。

list()

説明

デプロイ済のすべてのプロバイダについて、プロバイダ ID の配列を戻します。プロバイダ ID をリストできない場合は、SOAPException が発生します。

構文

```
public abstract String[] list();
```

query()

説明

指定されたプロバイダのデプロイメント・ディスクリプタを戻します。プロバイダが検出されない場合は、SOAPException が発生します。

構文

```
public abstract ProviderDeploymentDescriptor query( String providerId);
```

パラメータ	説明
providerId	プロバイダの ID。

setServiceManager()

説明

サービスのデプロイ情報を管理するために使用されているサービス・マネージャを、プロバイダ・マネージャが使用できるようにします。プロバイダの下でなんらかのサービスがデプロイされているかぎり、プロバイダ・マネージャは、サービス・マネージャを使用してそのプロバイダがアンデプロイされないようにすることができます。

構文

```
public abstract void setServiceManager( ServiceManager serviceManager);
```

パラメータ	説明
serviceManager	SOAP サーバー用のプロバイダのデプロイ情報を管理しているプロバイダ・マネージャ。

undeploy()

説明

指定されたプロバイダをアンデプロイし、アンデプロイ済プロバイダのデプロイ情報を含むディスクリプタを戻します。プロバイダが検出されない場合またはアンデプロイに失敗した場合は、`SOAPException` が発生します。

構文

```
public abstract ProviderDeploymentDescriptor undeploy( String providerId);
```

パラメータ	説明
providerId	アンデプロイするプロバイダの ID。

ServiceManager インタフェース

ServiceManager の説明

このインタフェースは、サービスを管理するインタフェースを定義します。サービス・マネージャは、サービスのデプロイ、サービスのアンデプロイおよびサービスのデプロイ情報へのアクセスを行うために SOAP エンジンによって使用されます。サービス・マネージャは、デプロイ情報をキャッシュし、キャッシュをメンテナンスします。

HTTP サーバーは、サービス・マネージャにセキュリティを提供します。サービス・マネージャは、どのリクエストが受け入れられるのかを、URL によって構成できます。サービス・マネージャに対する SOAP リクエストが他の URL に行われた場合、そのリクエストは拒否されます。この URL は SOAP サーブレットの別名にする必要があり、HTTP セキュリティを設定して指定された URL にポスト可能なユーザーを制御できます。

ServiceManager の構文

```
public interface ServiceManager

Interface oracle.soap.server.ServiceManager
```

ServiceManager のメソッド

表 11-6 ServiceManager のメソッドの概要

メソッド	説明
getRequiredRequestURI()（11-15 ページ）	サービス・マネージャがリクエストする URI を戻します。
deploy()（11-15 ページ）	指定されたサービスをデプロイします。
destroy()（11-15 ページ）	サービス・マネージャをクリーンアップします。
init()（11-16 ページ）	サービス・マネージャを初期化します。
list()（11-16 ページ）	プロバイダに関係なく、デプロイ済のすべてのサービスについて、プロバイダ ID の配列を戻します。
query()（11-17 ページ）	指定されたサービスのデプロイメント・ディスクリプタを戻します。
undeploy()（11-17 ページ）	指定されたサービスをアンデプロイし、そのディスクリプタを戻します。

getRequiredRequestURI()

説明

サービス・マネージャがリクエストする URI、またはすべての URI を使用できる場合は null を戻します。リクエストが受け入れられるためには、この URI にリクエストを行う必要があります。他の URI に対して行われたリクエストは拒否されます。

構文

```
public abstract String getRequiredRequestURI();
```

deploy()

説明

指定されたサービスをデプロイします。デプロイできない場合は、SOAPException が発生します。

構文

```
public abstract void deploy(ServiceDeploymentDescriptor sd);
```

パラメータ	説明
sd	デプロイするサービスのサービス・ディスクリプタ。

destroy()

説明

サービス・マネージャをクリーンアップします。サービス・マネージャをクリーンアップできない場合は、SOAPException が発生します。

構文

```
public abstract void destroy();
```

init()

説明

サービス・マネージャを初期化します。この実装は、プロバイダ・マネージャの null 値を処理できる必要があります。サービスのデプロイ情報にアクセスできない場合は、SOAPException が発生します。

構文

```
public abstract void init( Properties options,
                           ProviderManager providerManager);
```

パラメータ	説明
options	サービスのデプロイ情報へのアクセスを設定するために必要なオプション。
providerManager	SOAP サーバー用のプロバイダのデプロイ情報を管理しているプロバイダ・マネージャ（プロバイダ・マネージャが提供されていない場合は null）。サービス・マネージャは、プロバイダ・マネージャを使用して、新しいサービスのデプロイ時にそのプロバイダが存在するかどうかを確認する必要がある場合があります。

list()

説明

プロバイダに関係なく、デプロイ済のすべてのサービスについて、プロバイダ ID の配列を戻します。サービス ID をリストできない場合は、SOAPException が発生します。

構文

```
public abstract String[] list();
```

query()

説明

指定されたサービスのデプロイメント・ディスクリプタを戻します。サービスが検出されない場合は、SOAPException が発生します。

構文

```
public abstract ServiceDeploymentDescriptor query( String serviceId);
```

パラメータ	説明
serviceId	サービスの一意の URI。

undeploy()

説明

指定されたサービスをアンデプロイし、そのディスクリプタを戻します。サービスが検出されない場合またはアンデプロイに失敗した場合は、SOAPException が発生します。

構文

```
public abstract ServiceDeploymentDescriptor undeploy( String serviceId);
```

パラメータ	説明
serviceId	アンデプロイするサービスの URI。

ContainerContext クラス

ContainerContext の説明

このクラスは、SOAP サーバーが実行されているコンテナのコンテキストを定義します。実際のコンテンツは、サーブレット・エンジンの中など、サーバーが実行されている環境によって異なります。このクラスは、コンテナ固有のコンテンツのみを含みます。

ContainerContext の構文

```
public class ContainerContext extends Object

java.lang.Object
   |
   +----oracle.soap.server.ContainerContext
```

ContainerContext のフィールド

表 11-7 ContainerContext のフィールド

フィールド	構文	説明
SERVLET_CONTAINER	public static final String SERVLET_CONTAINER	Servlet コンテナ・タイプの値。

ContainerContext のメソッド

表 11-8 ContainerContext のメソッドの概要

メソッド	説明
ContainerContext()（11-19 ページ）	クラス・コンストラクタです。
getAttribute()（11-19 ページ）	指定された名前を持つ属性を戻します。
getAttributeNames()（11-20 ページ）	SOAP コンテキスト内で使用可能な属性名を含む列挙を戻します。
getContainerType()（11-20 ページ）	SOAP サーバーが実行されているコンテナ・タイプを戻します。
getHttpServlet()（11-20 ページ）	コンテナ・タイプが SERVLET_CONTAINER である場合に、SOAP リクエストを処理している HTTP サーブレットを戻します。

表 11-8 ContainerContext のメソッドの概要（続き）

メソッド	説明
removeAttribute()（11-21 ページ）	指定された名前を持つ属性をコンテキストから削除します。
setAttribute()（11-21 ページ）	SOAP コンテキスト内の指定された属性名にオブジェクトをバインドします。
setContainerType()（11-22 ページ）	コンテナ・タイプを設定します。
setHttpServlet()（11-22 ページ）	SERVLET_CONTAINER タイプのコンテナで実行中の SOAP サーバーに HTTP サーブレットを設定します。

ContainerContext()

説明

クラス・コンストラクタです。

構文

```
public ContainerContext();
```

getAttribute()

説明

指定された名前を持つ属性を戻します。その名前を持つ属性が存在しない場合は null を戻します。

構文

```
public Object getAttribute( String name);
```

パラメータ	説明
name	属性の名前を指定する文字列。

getAttributeNames()

説明

SOAP コンテキスト内で使用可能な属性名を含む列挙を戻します。

構文

```
public Enumeration getAttributeNames();
```

getContainerType()

説明

SOAP サーバーが実行されているコンテナ・タイプを戻します。

構文

```
public String getContainerType();
```

getHttpServlet()

説明

コンテナ・タイプが SERVLET_CONTAINER である場合に、SOAP リクエストを処理している HTTP サーブレットを戻します。サーブレットの属性が設定されていない場合は null を戻します。

構文

```
public HttpServlet getHttpServlet();
```

removeAttribute()

説明

指定された名前を持つ属性をコンテキストから削除します。その属性の削除後に getAttribute(java.lang.String) をコールして属性値を取得しようとすると、null が戻されます。

構文

```
public void removeAttribute( String name);
```

パラメータ	説明
name	削除する属性の名前を指定する文字列。

setAttribute()

説明

SOAP コンテキスト内の指定された属性名にオブジェクトをバインドします。指定された名前がすでに属性に使用されている場合は、古い属性を削除して、名前を新しい属性にバインドします。名前およびオブジェクトは、null にできません。

構文

```
public void setAttribute( String name,
                          Object object);
```

パラメータ	説明
name	属性の名前を指定する非 null 文字列。
object	バインドする属性を表す非 null オブジェクト。

setContainerType()

説明

コンテナ・タイプを設定します。

構文

```
public void setContainerType( String containerType);
```

パラメータ	説明
containerType	SOAP サーバーが実行されているコンテナ・タイプ。

setHttpServlet()

説明

SERVLET_CONTAINER タイプのコンテナで実行中の SOAP サーバーに HTTP サーブレットを設定します。

構文

```
public void setHttpServlet(HttpServlet servlet);
```

パラメータ	説明
servlet	SOAP リクエストを処理している HTTP サーブレット。

Logger クラス

Logger の説明

このクラスは、ロガーの実装の際にサポートされる必要がある機能を定義します。ロガーは、エラー・メッセージおよび情報メッセージを永続的に記録するために使用されます。

各ログ・リクエストでは重大性が指定されます。事前に指定された重大性以上である場合、情報を記録する必要があります。

重大性レベルは、低い順に次のようになります。

- SEVERITY_ERROR
- SEVERITY_STATUS
- SEVERITY_DEBUG

たとえば、重大性が SEVERITY_STATUS に設定されていると、重大性が SEVERITY_STATUS または SEVERITY_ERROR であるすべてのログ・リクエストが記録されます。

Logger の構文

```
Class oracle.soap.server.Logger

public abstract class Logger extends Object

java.lang.Object
   |
   +----oracle.soap.server.Logger
```

Logger のフィールド

表 11-9 Logger のフィールド

フィールド	構文	説明
SEVERITY_ERROR	public static final int SEVERITY_ERROR	エラー・メッセージの記録対象となる重大性レベル。
SEVERITY_STATUS	public static final int SEVERITY_STATUS	ステータス・メッセージの記録対象となる重大性レベル。
SEVERITY_DEBUG	public static final int SEVERITY_DEBUG	デバッグ目的で情報の記録対象となる重大性レベル。

表 11-9 Logger のフィールド（続き）

フィールド	構文	説明
SEVERITY_INVALID	protected static final int SEVERITY_INVALID	無効な重大性の設定。
SEVERITY_NAMES	public static String SEVERITY_NAMES[]	重大性レベルごとに出力可能な名前。重大性の値でインデックス付けされています。
DEFAULT_SEVERITY	public static final int DEFAULT_SEVERITY	実際に記録するログ・リクエストを判別する、デフォルトの重大性レベルの設定。デフォルトは SEVERITY_STATUS です。
OPTION_SEVERITY	public static final String OPTION_SEVERITY	ロガーの重大性を指定する構成オプション。
m_severity	protected int m_severity	ロガーの重大性の設定。

Logger のメソッド

表 11-10 Logger のメソッドの概要

メソッド	説明
Logger()（11-25 ページ）	クラス・コンストラクタです。
getSeverity()（11-25 ページ）	ロガーに対する現在の重大性の設定を戻します。
getSeverityName()（11-25 ページ）	指定された重大性に対応する重大性の名前を戻します。
getSeverityValue()（11-26 ページ）	指定された重大性の名前に対応する重大性の値を戻します。
init()（11-26 ページ）	構成パラメータを指定して、ロガーを初期化します（1 回のみ）。
isLoggable()（11-27 ページ）	指定された重大性レベルでメッセージを記録するかどうかを判別します。
log()（11-27 ページ）	メッセージを記録します。
setSeverity()（11-28 ページ）	現在の重大性を設定します。

Logger()

説明
クラス・コンストラクタです。

構文
```
public Logger();
```

getSeverity()

説明
ロガーに対する現在の重大性の設定を戻します。

構文
```
public int getSeverity();
```

getSeverityName()

説明
指定された重大性に対応する重大性の名前を戻します。

構文
```
protected final String getSeverityName( int severity);
```

パラメータ	説明
severity	重大性レベル（SEVERITY_xxx）。

getSeverityValue()

説明

指定された重大性の名前に対応する重大性の値を戻します。

構文

```
protected final int getSeverityValue( String severityName);
```

パラメータ	説明
severityName	重大性レベルの名前（エラーなど）。

init()

説明

構成パラメータを指定して、ロガーを初期化します（1 回のみ）。ロガーを初期化できない場合は、SOAPException が発生します。

構文

```
public abstract void init( Properties options,
                           ContainerContext context);
```

パラメータ	説明
options	ロガーの構成オプション。
context	SOAP サーバーが実行されているコンテナのコンテキスト。ロガーが使用できる情報が含まれています。

isLoggable()

説明

指定された重大性レベルでメッセージを記録するかどうかを判別します。指定された重大性レベルでメッセージを記録する場合は `true`、記録しない場合は `false` を戻します。

構文

```
public boolean isLoggable( int severity);
```

パラメータ	説明
severity	確認する重大性レベル。

log()

説明

メッセージを記録します。次の表に、オプションを示します。

構文	説明
public abstract void log(String msg, int severity);	指定された重大性の指定されたメッセージを記録します。
public abstract void log(String msg, Throwable t, int severity);	指定された重大性の指定されたメッセージおよび例外を記録します。
public abstract void log(Throwable t, int severity);	指定された重大性の指定された例外を記録します。

パラメータ	説明
msg	記録するメッセージ。
severity	情報の記録対象となる重大性。
t	ログに記録可能な例外。

setSeverity()

説明

現在の重大性を設定します。

構文

```
public void setSeverity(int severity);
```

パラメータ	説明
severity	ロガーに対する新しい重大性の設定。

ProviderDeploymentDescriptor クラス

ProviderDeploymentDescriptor の説明

このクラスは、特定のプロバイダに関するデプロイ情報を定義します。同じ実装を使用して異なるプロバイダをデプロイでき、プロバイダはプロバイダ・ディスクリプタでのみ区別できます。

ProviderDeploymentDescriptor の構文

```
public final class ProviderDeploymentDescriptor extends Object implements
Serializable

java.lang.Object
   |
   +----oracle.soap.server.ProviderDeploymentDescriptor
```

ProviderDeploymentDescriptor のメソッド

表 11-11 ProviderDeploymentDescriptor のメソッドの概要

メソッド	説明
ProviderDeploymentDescriptor()（11-30 ページ）	プロバイダ・ディスクリプタの新しいインスタンスを作成します。
fromXML()（11-30 ページ）	指定された XML 文書からプロバイダ・ディスクリプタを作成して戻します。
getClassname()（11-30 ページ）	プロバイダを実装するクラスの名前を戻します。
getId()（11-31 ページ）	プロバイダの一意の ID を戻します。
getOptions()（11-31 ページ）	プロバイダ固有のオプションを戻します。
getProviderType()（11-31 ページ）	プロバイダ・タイプを戻します。
setClassname()（11-31 ページ）	プロバイダを実装するクラスの名前を設定します。
setId()（11-32 ページ）	プロバイダ ID を設定します。
setOptions()（11-32 ページ）	オプションを設定します。
setProviderType()（11-32 ページ）	プロバイダ・タイプを設定します。

表 11-11 ProviderDeploymentDescriptor のメソッドの概要（続き）

メソッド	説明
toString()（11-33 ページ）	サービス・デプロイメント・ディスクリプタを文字列として書き出します。
toXML()（11-33 ページ）	サービス・デプロイメント・ディスクリプタを XML として書き出します。

ProviderDeploymentDescriptor()

説明

プロバイダ・ディスクリプタの新しいインスタンスを作成します。

構文

```
public ProviderDeploymentDescriptor();
```

fromXML()

説明

指定された XML 文書からプロバイダ・ディスクリプタを作成して戻します。

構文

```
public static ProviderDeploymentDescriptor fromXML( Element root);
```

パラメータ	説明
root	XML プロバイダ・ディスクリプタを表す文書のルート。

getClassname()

説明

プロバイダを実装するクラスの名前を戻します。

構文

```
public String getClassname();
```

getId()

説明
プロバイダの一意の ID を戻します。

構文
```
public String getId();
```

getOptions()

説明
プロバイダ固有のオプション、またはサービスに対するプロバイダ固有のオプションを表す値の組を戻します。

構文
```
public Hashtable getOptions();
```

getProviderType()

説明
プロバイダ・タイプを戻します。

構文
```
public String getProviderType();
```

setClassname()

説明
プロバイダを実装するクラスの名前を設定します。

構文
```
public void setClassname( String classname);
```

パラメータ	説明
classname	実装するクラスの名前。

setId()

説明
プロバイダ ID を設定します。

構文
```
public void setId( String id);
```

パラメータ	説明
id	一意のプロバイダ ID。

setOptions()

説明
オプションを設定します。

構文
```
public void setOptions( Hashtable options);
```

パラメータ	説明
options	サービスのプロバイダ実装固有のオプションを表す名前と値の組。

setProviderType()

説明
プロバイダ・タイプを設定します。

構文
```
public void setProviderType( String providerType);
```

パラメータ	説明
providerType	プロバイダ・タイプ。

toString()

説明

サービス・デプロイメント・ディスクリプタを文字列として書き出します。

構文

```
public String toString();
```

toXML()

説明

サービス・デプロイメント・ディスクリプタを XML として書き出します。

構文

```
public void toXML( Writer pr);
```

パラメータ	説明
pr	XML 出力用のライター。

RequestContext クラス

RequestContext の説明

このクラスは、SOAP リクエストに対するすべてのコンテキストを定義します。これには、プロバイダに渡される情報、および戻す前にプロバイダで設定する必要がある情報が含まれます。プロバイダにはリクエスト・エンベロープが指定されるため、リクエスト・パラメータのアンマーシャリングは、プロバイダが行います。同様に、レスポンス・エンベロープもプロバイダによって設定される必要がありますが、そのレスポンスをマーシャリングする必要があります。このレスポンスは、交換可能なハンドラが必要としています。次の情報は、SOAP エンジンによってプロバイダに提供されます。これによって、プロバイダはこの情報を `Provider.invoke()` で使用できます。

- `getEnvelope` - リクエストを含むエンベロープ。
- `getServiceDeploymentDescriptor` - メソッドがコールされているサービスのサービス・デプロイメント・ディスクリプタ。
- `getServiceId` - サービスの URI。
- `getUserContext` - サービスでメソッドをコールするユーザーを定義するセキュリティ・コンテキスト。
- `getMethodName` - サービスでコールされているメソッドの名前。

次の情報は、プロバイダが SOAP エンジンに指定する必要があります。

- `setResponseBytes` - マーシャリングされたレスポンス。これは、レスポンスが指定されると、レスポンス・エンベロープを作成し、そのエンベロープをマーシャリングすることによって作成できます。
- `setResponseEnvelope` - レスポンス・バイトと論理的に同等のレスポンス・エンベロープ。
- `getRequestEncodingStyle` - エラーが発生した場合にレスポンスに使用するエンコーディング・スタイル（設定されない場合は、デフォルトで、SOAP エンコーディングである Constants.NS_URI_SOAP_ENC）。プロバイダがこのエンコーディング・スタイルを必要としている場合は、例外の発生時に備えてできるだけ早くこの値を設定します。プロバイダは、リクエストまたはいずれかのパラメータと同じエンコーディングを使用できます。

RequestContext の構文

```
public class RequestContext extends Object

java.lang.Object
   |
   +----oracle.soap.server.RequestContext
```

RequestContext のメソッド

表 11-12 RequestContext のメソッドの概要

メソッド	説明
RequestContext()（11-36 ページ）	デフォルトのコンストラクタです。
getMethodName()（11-36 ページ）	SOAP リクエストに対して起動されているメソッドの名前を戻します。
getRequestEncodingStyle()（11-36 ページ）	リクエストに使用されたエンコーディング・スタイルを戻します。
getRequestEnvelope()（11-37 ページ）	実際の SOAP リクエストを表すエンベロープを戻します。
getResponseBytes()（11-37 ページ）	SOAP リクエスト用のレスポンス・ストリームを戻します。
getResponseEnvelope()（11-37 ページ）	SOAP レスポンスを表すエンベロープを戻します。
getResponseMap()（11-37 ページ）	SOAP レスポンスのシリアル化に必要なマッピング・レジストリを戻します。
getServiceDeploymentDescriptor()（11-38 ページ）	リクエストされたサービスのサービス・デプロイメント・ディスクリプタを戻します。
getServiceId()（11-38 ページ）	SOAP リクエストに対するサービス ID（URI）を戻します。
getUserContext()（11-38 ページ）	SOAP リクエストに対するユーザー・コンテキストを戻します。
setMethodName()（11-38 ページ）	SOAP リクエストに対するメソッド名を設定します。
setRequestEncodingStyle()（11-39 ページ）	リクエストに使用されたエンコーディング・スタイルを設定します。
setRequestEnvelope()（11-39 ページ）	実際の SOAP リクエストを表すエンベロープを設定します。
setResponseBytes()（11-39 ページ）	SOAP リクエスト用のレスポンス・ストリームを設定します。
setResponseEnvelope()（11-40 ページ）	SOAP レスポンスを表すエンベロープを設定します。
setResponseMap()（11-40 ページ）	SOAP レスポンス・エンベロープのシリアル化に必要なマッピング・レジストリを設定します。
setServiceDeploymentDescriptor（11-40 ページ）	リクエストされたサービスのサービス・デプロイメント・ディスクリプタを設定します。

表 11-12 RequestContext のメソッドの概要

メソッド	説明
setServiceId()（11-41 ページ）	SOAP リクエストに対するサービス ID（URI）を設定します。
setUserContext()（11-41 ページ）	SOAP リクエストに対するユーザー・コンテキストを設定します。

RequestContext()

説明
デフォルトのコンストラクタです。

構文
```
public RequestContext();
```

getMethodName()

説明
SOAP リクエストに対して起動されているメソッドの名前を戻します。

構文
```
public String getMethodName();
```

getRequestEncodingStyle()

説明
リクエストに使用されたエンコーディング・スタイルを戻します。

構文
```
public String getRequestEncodingStyle();
```

getRequestEnvelope()

説明

実際の SOAP リクエストを表すエンベロープを戻します。

構文

```
public Envelope getRequestEnvelope();
```

getResponseBytes()

説明

SOAP リクエスト用のレスポンス・ストリームを戻します。

構文

```
public ByteArrayOutputStream getResponseBytes();
```

getResponseEnvelope()

説明

SOAP レスポンスを表すエンベロープを戻します。

構文

```
public Envelope getResponseEnvelope();
```

getResponseMap()

説明

SOAP レスポンスのシリアル化に必要なマッピング・レジストリを戻します。

構文

```
public SOAPMappingRegistry getResponseMap();
```

getServiceDeploymentDescriptor()

説明

リクエストされたサービスのサービス・デプロイメント・ディスクリプタを戻します。プロバイダが Autonomous プロバイダである場合は null を戻します。

構文

```
public ServiceDeploymentDescriptor getServiceDeploymentDescriptor();
```

getServiceId()

説明

SOAP リクエストに対するサービス ID（URI）を戻します。

構文

```
public String getServiceId();
```

getUserContext()

説明

SOAP リクエストに対するユーザー・コンテキストを戻します。

構文

```
public UserContext getUserContext();
```

setMethodName()

説明

SOAP リクエストに対するメソッド名を設定します。メソッド名はエンベロープに含まれていますが、便宜上、サーバーがここにキャッシュできます。

構文

```
public void setMethodName( String methodName);
```

パラメータ	説明
methodName	サービスで起動されているメソッドの名前。

setRequestEncodingStyle()

説明

リクエストに使用されたエンコーディング・スタイルを設定します。

構文

```
public void setRequestEncodingStyle( String requestEncodingStyle);
```

パラメータ	説明
requestEncodingStyle	リクエスト・エンコーディング・スタイル。

setRequestEnvelope()

説明

実際の SOAP リクエストを表すエンベロープを設定します。

構文

```
public void setRequestEnvelope( Envelope envelope);
```

パラメータ	説明
envelope	SOAP エンベロープ。

setResponseBytes()

説明

SOAP リクエスト用のレスポンス・ストリームを設定します。

構文

```
public void setResponseBytes( ByteArrayOutputStream bytes);
```

パラメータ	説明
bytes	レスポンスを含むバイト配列の出力ストリーム。

setResponseEnvelope()

説明

SOAP レスポンスを表すエンベロープを設定します。

構文

```
public void setResponseEnvelope( Envelope envelope);
```

パラメータ	説明
envelope	SOAP レスポンス・エンベロープ。

setResponseMap()

説明

SOAP レスポンス・エンベロープのシリアル化に必要なマッピング・レジストリを設定します。

構文

```
public void setResponseMap( SOAPMappingRegistry smr);
```

setServiceDeploymentDescriptor

説明

リクエストされたサービスのサービス・デプロイメント・ディスクリプタを設定します。

構文

```
public void setServiceDeploymentDescriptor(
                  ServiceDeploymentDescriptor serviceDeploymentDescriptor);
```

パラメータ	説明
serviceDeploymentDescriptor	リクエストに対するサービス・デプロイメント・ディスクリプタ。

setServiceId()

説明

SOAP リクエストに対するサービス ID（URI）を設定します。

構文

```
public void setServiceId( String serviceId);
```

パラメータ	説明
serviceId	リクエストの送信先であるサービスの URI。

setUserContext()

説明

SOAP リクエストに対するユーザー・コンテキストを設定します。

構文

```
public void setUserContext( UserContext userContext);
```

パラメータ	説明
userContext	ユーザー・コンテキスト。

ServiceDeploymentDescriptor クラス

ServiceDeploymentDescriptor の説明

このクラスは、プロバイダのタイプに関係なく、SOAP サービスのデプロイ情報を定義します。このクラスは、任意の数の名前付きプロバイダ・オプションをサポートします。これによって、コードを変更しなくても、新しいタイプのプロバイダに対応するためにディスクリプタを拡張できます。

ServiceDeploymentDescriptor の構文

```
public final class ServiceDeploymentDescriptor extends Object implements
Serializable

java.lang.Object
   |
   +----oracle.soap.server.ServiceDeploymentDescriptor
```

ServiceDeploymentDescriptor のフィールド

表 11-13 ServiceDeploymentDescriptor のフィールド

フィールド	構文	説明
SERVICE_TYPE_RPC	public static final int SERVICE_TYPE_RPC	サービスが RPC ベースであることを示します。
SERVICE_TYPE_MESSAGE	public static final int SERVICE_TYPE_MESSAGE	サービスがメッセージ・ベースであることを示します。
SCOPE_REQUEST	public static final int SCOPE_REQUEST	フレッシュ・サービス・インスタンスを各リクエストに割り当てる必要があることを示します。
SCOPE_SESSION	public static final int SCOPE_SESSION	同一セッション内のすべてのリクエストが同一のサービス・インスタンスから提供されることを示します。
SCOPE_APPLICATION	public static final int SCOPE_APPLICATION	すべてのリクエストが同一のサービス・インスタンスから提供されることを示します。

ServiceDeploymentDescriptor のメソッド

表 11-14 ServiceDeploymentDescriptor のメソッドの概要

メソッド	説明
ServiceDeploymentDescriptor() (11-44 ページ)	新しいサービス・ディスクリプタを作成します。
buildFaultRouter() (11-45 ページ)	サービスの障害リスナーから作成された障害ルーターを戻します。
buildSOAPMappingRegistry() (11-45 ページ)	デプロイメント・ディスクリプタに登録されたすべての型マッピングから、XML シリアル化レジストリを生成します。
buildSqlClassMap() (11-45 ページ)	デプロイメント・ディスクリプタからの型マッピング情報を使用して、SQL 型から Java クラスへのマッピングを生成します。マップを生成できなかった場合は、SOAPException が発生します。
fromXML()（11-46 ページ）	サービス・デプロイメント・ディスクリプタに、指定された文書から情報（ディスクリプタの XML 表現）を移入します。
getDefaultSMRClass() (11-46 ページ)	デフォルトの SOAP マッピング・レジストリ・クラスを戻します。
getFaultListener() (11-46 ページ)	サービスに対する障害リスナーであるクラス名のリストを戻します。
getId()（11-46 ページ）	サービス ID（URI）を戻します。
getMethods()（11-47 ページ）	サービスによって提供されるメソッドのリストを戻します。
getProviderId()（11-47 ページ）	サービスのプロバイダ ID を戻します。
getProviderOptions() (11-47 ページ)	サービスのプロバイダ固有のオプションを表す名前と値の組を戻します。
getProviderType() (11-47 ページ)	プロバイダ・タイプを戻します。
getScope()（11-48 ページ）	適用範囲（SCOPE_xxx 定数のいずれか）を戻します。
getServiceType() (11-48 ページ)	サービス・タイプ（SERVICE_TYPE_xxx 定数のいずれか）を戻します。
getSqlMap()（11-48 ページ）	SQL 型から Java 型へのマッピングを戻します。
getTypeMappings() (11-48 ページ)	XML から Java への型マッピングを戻します。このマッピングでは、XML から Java への非シリアル化、および Java から XML へのシリアル化の方法が定義されます。

表 11-14 ServiceDeploymentDescriptor のメソッドの概要（続き）

メソッド	説明
isMethodValid()（11-49 ページ）	指定されたメソッドがサービスに有効であるかどうかを判別します。
setDefaultSMRClass()（11-49 ページ）	デフォルトの SOAP マッピング・レジストリ・クラスを設定します。
setFaultListener()（11-49 ページ）	障害リスナー・リストを設定します。
setId()（11-50 ページ）	サービス ID（有効な URI）を設定します。
setMethods()（11-50 ページ）	サービスによって提供されるメソッドのリストを設定します。
setProviderId()（11-50 ページ）	サービスのプロバイダの ID を設定します。
setProviderOptions()（11-51 ページ）	プロバイダ固有のオプションを設定します。
setProviderType()（11-51 ページ）	プロバイダ・タイプを設定します。
setScope()（11-51 ページ）	実行の適用範囲を設定します。
setServiceType()（11-52 ページ）	サービス・タイプを設定します。
setSqlMap()（11-52 ページ）	SQL 型から Java 型へのマッピングを設定します。
setTypeMappings()（11-52 ページ）	XML から Java への型マッピングを設定します。このマッピングでは、XML から Java への非シリアル化、および Java から XML へのシリアル化の方法が定義されます。
toXML()（11-53 ページ）	サービス・デプロイメント・ディスクリプタを XML として書き出します。
toString()（11-53 ページ）	ディスクリプタの出力可能な表現を戻します。

ServiceDeploymentDescriptor()

説明

新しいサービス・ディスクリプタを作成します。

構文

```
public ServiceDeploymentDescriptor();
```

buildFaultRouter()

説明

サービスの障害リスナーから作成された障害ルーターを戻します。

構文

```
public SOAPFaultRouter buildFaultRouter();
```

buildSOAPMappingRegistry()

説明

デプロイメント・ディスクリプタに登録されたすべての型マッピングから、XML シリアル化レジストリを生成します。

構文

```
public static SOAPMappingRegistry buildSOAPMappingRegistry(
                                        ServiceDeploymentDescriptor sdd);
```

パラメータ	説明
sdd	サービス・デプロイメント・ディスクリプタ。

buildSqlClassMap()

説明

デプロイメント・ディスクリプタからの型マッピング情報を使用して、SQL 型から Java クラスへのマッピングを生成します。マップを生成できなかった場合は、SOAPException が発生します。

構文

```
public static Hashtable buildSqlClassMap( ServiceDeploymentDescriptor sdd);
```

パラメータ	説明
sdd	使用するサービス・デプロイメント・ディスクリプタ。

fromXML()

説明

サービス・デプロイメント・ディスクリプタに、指定された文書から情報（ディスクリプタの XML 表現）を移入します。このサービス・デプロイメント・ディスクリプタを戻します。文書が無効である場合は、`IllegalArgumentException` が発生します。

構文

```
public static ServiceDeploymentDescriptor fromXML( Element root);
```

パラメータ	説明
root	サービス・ディスクリプタを表す XML 文書のルート。

getDefaultSMRClass()

説明

デフォルトの SOAP マッピング・レジストリ・クラスを戻します。

構文

```
public String getDefaultSMRClass();
```

getFaultListener()

説明

サービスに対する障害リスナーであるクラス名のリストを戻します。

構文

```
public String[] getFaultListener();
```

getId()

説明

サービス ID（URI）を戻します。

構文

```
public String getId();
```

getMethods()

説明
サービスによって提供されるメソッドのリストを戻します。

構文
```
public String[] getMethods();
```

getProviderId()

説明
サービスのプロバイダ ID を戻します。

構文
```
public String getProviderId();
```

getProviderOptions()

説明
サービスのプロバイダ固有のオプションを表す名前と値の組を戻します。

構文
```
public Hashtable getProviderOptions();
```

getProviderType()

説明
プロバイダ・タイプを戻します。

構文
```
public String getProviderType();
```

getScope()

説明

適用範囲（SCOPE_xxx 定数のいずれか）を戻します。

構文

```
public int getScope();
```

getServiceType()

説明

サービス・タイプ（SERVICE_TYPE_xxx 定数のいずれか）を戻します。

構文

```
public int getServiceType();
```

getSqlMap()

説明

SQL 型から Java 型へのマッピングを戻します。

構文

```
public Hashtable getSqlMap();
```

getTypeMappings()

説明

XML から Java への型マッピングを戻します。このマッピングでは、XML から Java への非シリアル化、および Java から XML へのシリアル化の方法が定義されます。

構文

```
public TypeMapping[] getTypeMappings();
```

isMethodValid()

説明

指定されたメソッドがサービスに有効であるかどうかを判別します。メソッドがサービスに対して有効である場合は true、有効でない場合は false を戻します。

構文

```
public boolean isMethodValid( String methodName);
```

setDefaultSMRClass()

説明

デフォルトの SOAP マッピング・レジストリ・クラスを設定します。

構文

```
public void setDefaultSMRClass( String defaultSMRClass);
```

パラメータ	説明
defaultSMRClass	デフォルトの SOAP マッピング・レジストリ・クラス。

setFaultListener()

説明

障害リスナー・リストを設定します。

構文

```
public void setFaultListener( String faultListener[]);
```

パラメータ	説明
faultListener	サービスに対する障害リスナーであるクラス名のリスト。

setId()

説明
サービス ID（有効な URI）を設定します。

構文
```
public void setId( String id);
```

パラメータ	説明
id	サービス URI。

setMethods()

説明
サービスによって提供されるメソッドのリストを設定します。

構文
```
public void setMethods( String methods[]);
```

パラメータ	説明
methods	提供されたメソッドのリスト。

setProviderId()

説明
サービスのプロバイダの ID を設定します。

構文
```
public void setProviderId( String providerId);
```

パラメータ	説明
providerId	サービスのプロバイダ ID。

setProviderOptions()

説明

プロバイダ固有のオプションを設定します。

構文

```
public void setProviderOptions( Hashtable providerOptions);
```

パラメータ	説明
providerOptions	サービスのプロバイダ固有のオプションを表す名前と値の組。

setProviderType()

説明

プロバイダ・タイプを設定します。

構文

```
public void setProviderType( String providerType);
```

パラメータ	説明
providerType	プロバイダ・タイプ（任意の文字列）。プロバイダ・タイプは、（プロバイダ固有のオプションの）XML サービス・ディスクリプタの検証に使用されます。

setScope()

説明

実行の適用範囲を設定します。

構文

```
public void setScope( int scope);
```

パラメータ	説明
scope	適用範囲（SCOPE_xxx 定数のいずれか）。

setServiceType()

説明
サービス・タイプを設定します。

構文
```
public void setServiceType( int serviceType);
```

パラメータ	説明
serviceType	サービス・タイプ（SERVICE_TYPE_xxx 定数のいずれか）。

setSqlMap()

説明
SQL 型から Java 型へのマッピングを設定します。

構文
```
public void setSqlMap( Hashtable sqlMap);
```

パラメータ	説明
sqlMap	SQL 型から Java クラスへのマッピング。

setTypeMappings()

説明
XML から Java への型マッピングを設定します。このマッピングでは、XML から Java への非シリアル化、および Java から XML へのシリアル化の方法が定義されます。

構文
```
public void setTypeMappings( TypeMapping typeMappings[]);
```

パラメータ	説明
typeMappings	型マッピング。

toXML()

説明

サービス・デプロイメント・ディスクリプタを XML として書き出します。

構文

```
public void toXML( Writer pr);
```

パラメータ	説明
pr	XML 出力用のライター。

toString()

説明

ディスクリプタの出力可能な表現を戻します。

構文

```
public String toString();
```

SOAPServerContext クラス

SOAPServerContext の説明

このクラスは、サーバーが実行されているコンテナのタイプに関係なく、SOAP サーバーのコンテキストを定義します。

SOAPServerContext の構文

```
public class SOAPServerContext extends Object

java.lang.Object
   |
   +----oracle.soap.server.SOAPServerContext
```

SOAPServerContext のメソッド

表 11-15 SOAPServerContext のメソッドの概要

メソッド	説明
SOAPServerContext()（11-55 ページ）	デフォルトのコンストラクタです。
getAttribute()（11-55 ページ）	指定された名前を持つ属性を戻します。その名前を持つ属性が存在しない場合は null を戻します。
getAttributeNames()（11-55 ページ）	SOAP コンテキスト内で使用可能な属性名を含む列挙を戻します。
getGlobalContext()（11-55 ページ）	コンテキスト・ノードを戻します。
getLogger()（11-56 ページ）	SOAP ロガーを戻します。
removeAttribute()（11-56 ページ）	指定された名前を持つ属性をコンテキストから削除します。
setAttribute()（11-56 ページ）	SOAP コンテキスト内の指定された属性名にオブジェクトをバインドします。
setGlobalContext ()（11-57 ページ）	グローバル・コンテキストを設定します。このコンテキストには、SOAP サーバー全体のオブジェクトが含まれます。
setLogger()（11-57 ページ）	ロガーを設定します。ロガーは、情報メッセージおよびデバッグ・メッセージをテキストベースで記録するために使用されます。

SOAPServerContext()

説明

デフォルトのコンストラクタです。

構文

```
public SOAPServerContext();
```

getAttribute()

説明

属性値を含むオブジェクトを戻します。その名前の属性が存在しない場合は null を戻します。

構文

```
public Object getAttribute( String name);
```

パラメータ	説明
name	取得する属性の名前を指定する文字列。

getAttributeNames()

説明

SOAP コンテキスト内で使用可能な属性名を含む列挙を戻します。

構文

```
public Enumeration getAttributeNames();
```

getGlobalContext()

説明

SOAP サーバー全体のオブジェクトを含むグローバル・コンテキストを戻します。属性が設定されていない場合は null を戻します。

構文

```
public Hashtable getGlobalContext();
```

getLogger()

説明

SOAP ロガーを戻します。ロガーは、情報メッセージおよびデバッグ・メッセージを記録するために使用されます。

構文

```
public Logger getLogger();
```

removeAttribute()

説明

指定された名前を持つ属性をコンテキストから削除します。その属性の削除後に getAttribute(java.lang.String) をコールして属性値を取得しようとすると、null が戻されます。

構文

```
public void removeAttribute( String name);
```

パラメータ	説明
name	削除する属性の名前を指定する文字列。

setAttribute()

説明

SOAP コンテキスト内の指定された属性名にオブジェクトをバインドします。指定された名前がすでに属性に使用されている場合は、古い属性を削除して、名前を新しい属性にバインドします。名前およびオブジェクトは、null にできません。

構文

```
public void setAttribute( String name,
                          Object object);
```

パラメータ	説明
name	属性の名前を指定する非 null 文字列。
object	バインドする属性を表す非 null オブジェクト。

setGlobalContext ()

説明

グローバル・コンテキストを設定します。このコンテキストには、SOAP サーバー全体のオブジェクトが含まれます。

構文

```
public void setGlobalContext( Hashtable globalContext);
```

パラメータ	説明
globalContext	グローバル・コンテキスト。

setLogger()

説明

ロガーを設定します。ロガーは、情報メッセージおよびデバッグ・メッセージをテキストベースで記録するために使用されます。

構文

```
public void setLogger( Logger logger);
```

パラメータ	説明
logger	SOAP ロガー。

UserContext クラス

UserContext の説明

このクラスは、SOAP サービス・リクエストのユーザー・コンテキストを定義します。いくつかの属性が事前に定義されており、それらに対する set メソッドおよび get メソッドが提供されています。また、プロバイダは、getAttribute および setAttribute を使用して、追加の属性を定義できます。

HttpServlet および HttpSession は実際にこのクラスに属すのではなく、JavaProvider で必要とされることに注意してください。

UserContext の構文

```
public class UserContext extends Object

java.lang.Object
   |
   +----oracle.soap.server.UserContext
```

UserContext のメソッド

表 11-16 UserContext のメソッドの概要

メソッド	説明
UserContext()（11-59 ページ）	デフォルトのコンストラクタです。
getAttribute()（11-60 ページ）	指定された名前を持つ属性を戻します。
getAttributeNames()（11-60 ページ）	SOAP コンテキスト内で使用可能な属性名を含む列挙を戻します。
getCertificate()（11-60 ページ）	SOAP リクエストを行うユーザー用のユーザー認証を戻します。
getHttpServlet()（11-61 ページ）	SOAP リクエストを処理している HTTP サーブレットを戻します。
getHttpSession()（11-61 ページ）	SOAP リクエストに対する HTTP セッションを戻します。
getRemoteAddress()（11-61 ページ）	リクエストを送信したリモート・クライアントのインターネット・プロトコル（IP）・アドレスを戻します。
getRemoteHost()（11-61 ページ）	リクエストを送信したリモート・クライアントのホスト名を戻します。
getRequestURI()（11-62 ページ）	リクエストの URI を戻します。

表 11-16 UserContext のメソッドの概要（続き）

メソッド	説明
getSecureChannel()（11-62 ページ）	チャネルが保護されているかどうかを示します。
getUsername()（11-62 ページ）	SOAP リクエストに対するプロトコル固有のユーザー名を戻します。
removeAttribute()（11-63 ページ）	指定された名前を持つ属性をコンテキストから削除します。
setAttribute()（11-63 ページ）	SOAP コンテキスト内の指定された属性名にオブジェクトをバインドします。
setCertificate()（11-64 ページ）	ユーザー認証を設定します。
setHttpServlet()（11-64 ページ）	HTTP サーブレットを設定します。
setHttpSession()（11-64 ページ）	HTTP セッションを設定します。
setRemoteAddress()（11-65 ページ）	クライアントのリモート IP アドレスを設定します。
setRemoteHost()（11-65 ページ）	SOAP リクエストを行うリモート・クライアントのホスト名を設定します。
setRequestURI()（11-65 ページ）	リクエストの URI を設定します。
setSecureChannel()（11-66 ページ）	チャネルが保護されているかどうかを示すインジケータを設定します。
setUsername()（11-66 ページ）	プロトコル固有のユーザー名を設定します。

UserContext()

説明

デフォルトのコンストラクタです。

構文

```
public UserContext();
```

getAttribute()

説明

指定された名前を持つ属性を戻します。その名前を持つ属性が存在しない場合は null を戻します。

構文

```
public Object getAttribute( String name);
```

パラメータ	説明
name	属性の名前を指定する文字列。

getAttributeNames()

説明

SOAP コンテキスト内で使用可能な属性名を含む列挙を戻します。

構文

```
public Enumeration getAttributeNames();
```

getCertificate()

説明

SOAP リクエストを行うユーザー用のユーザー認証を戻します。属性が設定されていない場合は null を戻します。

構文

```
public Object getCertificate();
```

getHttpServlet()

説明

SOAP リクエストを処理している HTTP サーブレットを戻します。サーブレットの属性が設定されていない場合は null を戻します。

構文

```
public HttpServlet getHttpServlet();
```

getHttpSession()

説明

SOAP リクエストに対する HTTP セッションを戻します。セッションの属性が設定されていない場合は null を戻します。

構文

```
getHttpSession public HttpSession getHttpSession();
```

getRemoteAddress()

説明

リクエストを送信したリモート・クライアントのインターネット・プロトコル（IP）・アドレスを戻します。

構文

```
public String getRemoteAddress();
```

getRemoteHost()

説明

リクエストを送信したリモート・クライアントのホスト名を戻します。

構文

```
public String getRemoteHost();
```

getRequestURI()

説明

リクエストの URI を戻します。

構文

```
public String getRequestURI();
```

getSecureChannel()

説明

チャネルが保護されているかどうかを示します。チャネルが保護されている場合は `true`、保護されていない場合は `false` を戻します。

構文

```
public boolean getSecureChannel();
```

getUsername()

説明

SOAP リクエストに対するプロトコル固有のユーザー名を戻します。属性が設定されていない場合は `null` を戻します。

構文

```
public String getUsername();
```

removeAttribute()

説明

指定された名前を持つ属性をコンテキストから削除します。その属性の削除後に getAttribute(java.lang.String) をコールして属性値を取得しようとすると、null が戻されます。

構文

```
public void removeAttribute( String name);
```

パラメータ	説明
name	属性の名前を指定する非 null 文字列。

setAttribute()

説明

SOAP コンテキスト内の指定された属性名にオブジェクトをバインドします。指定された名前がすでに属性に使用されている場合は、古い属性を削除して、名前を新しい属性にバインドします。名前およびオブジェクトは、null にできません。

構文

```
public void setAttribute( String name,
                          Object object);
```

パラメータ	説明
name	属性の名前を指定する非 null 文字列。
object	バインドする属性を表す非 null オブジェクト。

setCertificate()

説明
ユーザー認証を設定します。

構文
```
public void setCertificate( Object certificate);
```

パラメータ	説明
certificate	SOAP リクエストを行うユーザー用のユーザー認証。

setHttpServlet()

説明
HTTP サーブレットを設定します。

構文
```
public void setHttpServlet( HttpServlet servlet);
```

パラメータ	説明
servlet	SOAP リクエストを処理している HTTP サーブレット。

setHttpSession()

説明
HTTP セッションを設定します。

構文
```
public void setHttpSession( HttpSession session);
```

パラメータ	説明
servlet	SOAP リクエストに対する HTTP セッション。

setRemoteAddress()

説明
クライアントのリモート IP アドレスを設定します。

構文
```
public void setRemoteAddress( String remoteAddress);
```

パラメータ	説明
remoteAddress	SOAP リクエストを行うクライアントの IP アドレス。

setRemoteHost()

説明
SOAP リクエストを行うリモート・クライアントのホスト名を設定します。

構文
```
public void setRemoteHost( String remoteHost);
```

パラメータ	説明
remoteHost	SOAP リクエストを行うクライアントのホスト名。

setRequestURI()

説明
リクエストの URI を設定します。

構文
```
public void setRequestURI( String uri);
```

パラメータ	説明
uri	リクエストの URI。

setSecureChannel()

説明

チャネルが保護されているかどうかを示すインジケータを設定します。

構文

```
public void setSecureChannel( boolean secureChannel);
```

パラメータ	説明
secureChannel	チャネルが保護されている場合は `true`、保護されていない場合は `false`。

setUsername()

説明

プロトコル固有のユーザー名を設定します。

構文

```
public void setUsername( String username);
```

パラメータ	説明
username	SOAP リクエストに対するプロトコル固有のユーザー名。

oracle.soap.transport パッケージ

oracle.soap.transport の説明

このパッケージには、XDK for Java で Oracle SOAP をサポートする OracleSOAPTransport インタフェースが含まれています。

OracleSOAPTransport インタフェース

OracleSOAPTransport の説明

このインタフェースは、Oracle 固有のトランスポート拡張機能を定義します。

OracleSOAPTransport の構文

oracle.soap.transport.OracleSOAPTransport

```
public interface OracleSOAPTransport extends SOAPTransport
```

OracleSOAPTransport のメソッド

表 11-17 OracleSOAPTransport のメソッドの概要

メソッド	説明
close()（11-67 ページ）	トランスポートをクローズして、任意のクリーンアップを実行します。
getProperties()（11-68 ページ）	接続プロパティを戻します。
setProperties()（11-68 ページ）	接続プロパティを設定します。

close()

説明

トランスポートをクローズして、クリーンアップを実行します。

構文

```
public abstract void close();
```

getProperties()

説明

接続プロパティを戻します。

構文

```
public abstract Properties getProperties();
```

setProperties()

説明

接続プロパティを設定します。

構文

```
public abstract void setProperties( Properties prop);
```

パラメータ	説明
prop	接続プロパティ。

oracle.soap.transport.http パッケージ

oracle.soap.transport.http の説明

このパッケージには、OracleSOAPTransport を実装する OracleSOAPHTTPConnection クラスが含まれています。Oracle SOAP Client API は交換可能なトランスポートをサポートしているため、クライアントは簡単にトランスポートを変更できます。使用可能なトランスポートには HTTP および HTTPS（保護 HTTP）が含まれます。

OracleSOAPHTTPConnection クラス

OracleSOAPHTTPConnection の説明

このクラスは、OracleSOAPTransport を実装します。

OracleSOAPHTTPConnection の構文

```
public class OracleSOAPHTTPConnection extends Object

java.lang.Object
   |
   +----oracle.soap.transport.http.OracleSOAPHTTPConnection
```

OracleSOAPHTTPConnection のフィールド

表 11-18 OracleSOAPHTTPConnection のフィールド

フィールド	構文	説明
ALLOW_USER_INTERACTION	public static final String ALLOW_USER_INTERACTION	ユーザーの介入を設定するプロパティ。
AUTH_TYPE	public static final String AUTH_TYPE	HTTP 認証のタイプ（Basic または Digest）の定義に使用されるプロパティ。
CIPHERS	public static final String CIPHERS	HTTPS に使用される Cipher Suite の定義に使用されるプロパティ。Cipher Suite のリストは、コロンで区切られます。

表 11-18 OracleSOAPHTTPConnection のフィールド（続き）

フィールド	構文	説明
PASSWORD	public static final String PASSWORD	HTTP パスワードの定義に使用されるプロパティ。
PROXY_AUTH_TYPE	public static final String PROXY_AUTH_TYPE	プロキシ認証のタイプ（Basic または Digest）の定義に使用されるプロパティ。
PROXY_HOST	public static final String PROXY_HOST	プロキシ・ホストの定義に使用されるプロパティ。
PROXY_PASSWORD	public static final String PROXY_PASSWORD	プロキシ・パスワードの定義に使用されるプロパティ。
PROXY_PORT	public static final String PROXY_PORT	プロキシ・ポートの定義に使用されるプロパティ。
PROXY_USERNAME	public static final String PROXY_USERNAME	プロキシ・ユーザー名の定義に使用されるプロパティ。
STATUS_LINE	public static final String STATUS_LINE	HTTP ヘッダーからの HTTP のステータス行の取得（`getHeaders()`）に使用されるプロパティ。
USERNAME	public static final String USERNAME	HTTP ユーザー名の定義に使用されるプロパティ。
WALLET_LOCATION	public static final String WALLET_LOCATION	HTTPS に使用される Wallet の場所の定義に使用されるプロパティ。
WALLET_PASSWORD	public static final String WALLET_PASSWORD	HTTPS に使用される Wallet のパスワードの定義に使用されるプロパティ。

OracleSOAPHTTPConnection のメソッド

表 11-19 OracleSOAPHTTPConnection のメソッドの概要

メンバー	説明
OracleSOAPHTTPConnection()（11-71 ページ）	指定されたプロパティから OracleSOAPHTTPConnection の新しいインスタンスを作成します。
close()（11-72 ページ）	接続をクローズします。
finalize()（11-72 ページ）	接続をファイナライズします。
getHeaders()（11-72 ページ）	プロトコルによって生成されたヘッダーに対するすべてのヘッダーを含むハッシュテーブルを戻します。
getProperties()（11-72 ページ）	接続プロパティを戻します。
receive()（11-73 ページ）	受信したレスポンスの読込み元であるバッファ・リーダーを戻します。
send()（11-73 ページ）	指定された URL にエンベロープを転送するようにリクエストします。
setProperties()（11-74 ページ）	接続プロパティを設定します。

OracleSOAPHTTPConnection()

説明

指定されたプロパティから OracleSOAPHTTPConnection の新しいインスタンスを作成します。

構文

```
public OracleSOAPHTTPConnection( Properties prop);
```

パラメータ	説明
prop	接続プロパティ。

close()

説明

接続をクローズします。このメソッドのコール後、receive メソッドによって戻された BufferedReader をクローズできますが、これを使用しないでください。このメソッドをコールすると、ガベージ・コレクタを実行せずにリソースが解放されます。

構文

```
public void close();
```

finalize()

説明

接続をファイナライズします。

構文

```
public void finalize();
```

getHeaders()

説明

プロトコルによって生成されたヘッダーに対するすべてのヘッダーを含むハッシュテーブルを戻します。SOAP クライアントは、このメソッドを直接使用しないでください。かわりに、org.apache.soap.rpc.Call() を使用します。

構文

```
public Hashtable getHeaders();
```

getProperties()

説明

接続プロパティを戻します。

構文

```
public Properties getProperties();
```

receive()

説明

受信したレスポンスの読込み元であるバッファ・リーダーを戻します。レスポンスを受信しなかった場合は null を戻します。SOAP クライアントは、このメソッドを直接使用しないでください。かわりに、org.apache.soap.rpc.Call() を使用します。

構文

```
public BufferedReader receive();
```

send()

説明

指定された URL にエンベロープを転送するようにリクエストします。receive() ファンクションをコールすると、レスポンス（存在する場合）が取得されます。SOAP クライアントは、このメソッドを直接使用しないでください。かわりに、org.apache.soap.rpc.Call() を使用します。エラーが発生した場合は、SOAPException が発生し、適切な原因コードが表示されます。

構文

```
public void send( URL sendTo,
                  String action,
                  Hashtable headers,
                  Envelope env,
                  SOAPMappingRegistry smr,
                  int timeout);
```

パラメータ	説明
sendTo	エンベロープの送信先 URL。
action	SOAPAction ヘッダー・フィールドの値。
headers	プロトコル・ヘッダーとして処理する他のすべてのヘッダー・フィールド。
env	送信するエンベロープ。
smr	渡された XML と Java の間の型マッピング・レジストリ。
timeout	リクエスト SOAP コンテキスト。

setProperties()

説明

接続プロパティを設定します。

構文

```
public void setProperties( Properties prop);
```

パラメータ	説明
prop	接続プロパティ。

oracle.soap.util.xml パッケージ

oracle.soap.util.xml の説明

このパッケージには、XmlUtils クラスが含まれています。

XmlUtils クラス

XmlUtils の説明

このクラスは、OracleSOAPTransport に Oracle 固有のトランスポート拡張機能を実装します。このクラスの API によって、SOAP クライアントは SOAP サービスに対するリクエストを作成する XML 文書を生成し、SOAP リクエストを処理できます。Oracle SOAP は、有効な SOAP リクエストを送信するすべてのクライアントからのリクエストを処理します。

XmlUtils の構文

```
public class XmlUtils

java.lang.Object
   |
   +----oracle.soap.util.xml.XmlUtils
```

XmlUtils の構文

表 11-20 XmlUtils のメソッドの概要

メンバー	説明
XmlUtils()（11-76 ページ）	デフォルト・コンストラクタです。
extractServiceId()（11-76 ページ）	エンベロープからサービス ID を戻します。
extractMethodName()（11-76 ページ）	エンベロープからメソッド名を戻します。
parseXml()（11-77 ページ）	指定された XML ファイルを解析して、XML 文書を戻します。
createDocument()（11-77 ページ）	文書を作成します。

XmlUtils()

説明
デフォルトのコンストラクタです。

構文
```
public XmlUtils();
```

extractServiceId()

説明
エンベロープからサービス ID を戻します。これは、最初の本体エントリの名前空間 URI です。エンベロープからサービス URI を取得できない場合は、SOAPException が発生します。

構文
```
public static String extractServiceId(Envelope envelope);
```

パラメータ	説明
envelope	SOAP エンベロープ。

extractMethodName()

説明
エンベロープからメソッド名を戻します。これは、最初の本体エントリの名前です。エンベロープからメソッド名を取得できない場合は、SOAPException が発生します。

構文
```
public static String extractMethodName( Envelope envelope);
```

パラメータ	説明
envelope	SOAP エンベロープ。

parseXml()

説明

指定された XML ファイルを解析して、XML 文書を戻します。ファイルが検出されない場合、または解析エラーか I/O エラーが発生した場合は、SOAPException が発生します。次の表に、オプションを示します。

構文	説明
public static Document parseXml(String filename);	ファイル名を指定すると、指定した XML ファイルを解析して、XML 文書を戻します。
public static Document parseXml(Reader reader);	指定された XML ファイルを解析して、リーダーから XML 文書を戻します。
public static Document parseXml(InputStream is);	指定された XML ファイルを解析して、入力ストリームから XML 文書を戻します。

パラメータ	説明
filename	XML ファイルへのフルパス。
reader	XML のリーダー。
is	入力ストリーム・ソース。

createDocument()

説明

文書を作成します。文書を作成できない場合は、SOAPException が発生します。

構文

```
public static Document createDocument();
```

12

TransX Utility for Java

TransX Utility を使用すると、翻訳済のシード・データおよびメッセージを簡単にデータベースにロードできます。また、翻訳用の文字列を準備し、翻訳し、読み取ることによって、国際化コストが削減されます。TransX Utility によって、翻訳データのフォーマット・エラーが最小化され、データベースに事前に指定された場所に翻訳コンテンツが正確にロードされます。

TransX Utility を使用すると、翻訳済のメッセージおよびシード・データをロードする開発グループは、国際化要件を簡単に満たすことができます。データが事前定義されたフォーマットに変換されると、TransX Utility は、そのフォーマットを検証します。ファイルのエンコーディングでは、エンコーディングを定義する XML を利用しているため、翻訳済のデータは自動的にロードされます。これによって、データ・ファイルが XML 標準に準拠しているかぎり、不適切なエンコーディングによってロード時のエラーが発生することはありません。

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

TransX Utility コマンドライン・インタフェース

TransX Utility コマンドラインの構文

```
java oracle.xml.transx.loader [options] connect_string username password datasource
[datasource(s)]
java oracle.xml.transx.loader -v datasource [datasource(s)]
java oracle.xml.transx.loader -x connect_string username password table [column(s)]
java oracle.xml.transx.loader -s connect_string username password filename table
[column(s)]
```

TransX Utility コマンドライン・パラメータ

表 12-1 TransX Utility のコマンドライン・パラメータ

パラメータ	説明
connect_string	JDBC 接続文字列。接続文字列情報は、@ 記号を使用すると省略できます（たとえば、jdbc:oracle:thin:@）。
username	データベース・ユーザー名。
password	データベース・ユーザーのパスワード。
datasource	XML データ・ソース。
option	表 12-2「TransX Utility コマンドライン・オプション」に示すいずれかのオプション。

TransX Utility コマンドライン・オプションおよび例外

表 12-2 TransX Utility コマンドライン・オプション

オプション	説明
-u	既存の行を更新します。このオプションを指定すると、既存の行がスキップされずに更新されます。更新操作の対象から列を除外するには、useforupdate 属性を no に指定します。
-e	行がすでにデータベースに存在する場合に例外を発生させます。このオプションを指定すると、重複行が検出された場合に例外が発生します。デフォルトでは、重複行がスキップされます。データベースおよびデータセットの検索キー列の値が同じ場合、行が重複しているとみなされます。

表 12-2 TransX Utility コマンドライン・オプション（続き）

オプション	説明
-x	データベースのデータを事前に定義したフォーマットで出力します。これによって、TransX がアンロードを実行しますが、-s オプションとは異なり、stdout に出力します。オペレーティング・システムが介入すると、予期しないトランスコーディングによるデータの損失が発生する場合があるため、この出力先をファイルに指定することはお薦めしません。
-s	データベースのデータを事前に定義したフォーマットでファイルに保存します。これは、アンロードのオプションです。このオプションを指定すると、データベースへの問合せが実行され、その結果が事前に定義した XML 形式にフォーマットされて、指定したファイル名で格納されます。
-p	ロードする XML を出力します。XSU の正規フォーマットで挿入するデータセットを出力します。
-t	更新用の XML を出力します。XSU の正規フォーマットで更新するデータセットを出力します。
-o	検証を省略します（デフォルトでは、データセットの解析中に検証されます）。これによって、TransX がデフォルトで実行されるフォーマットの検証を省略します。
-v	データ・フォーマットを検証し、ロードせずに終了します。これによって、TransX が検証を実行し、終了します。
-w	空白を保持します。これによって、TransX が空白文字（\t、\r、\n、'' など）を重要なものとして処理します。デフォルトでは、文字列データ要素内の連続した空白文字は、1 つの空白文字に圧縮されます。

表 12-3 TransX Utility コマンドライン例外

例外	説明
-u、-e	相互に排他的です。
-v	データが後に続く唯一のオプションである必要があります。
-x	接続情報および SQL 問合せが後に続く唯一のオプションである必要があります。 すべての引数を省略すると、フロントエンドの使用情報が表に表示されます。

TransX Utility Application Program Interface

この項では、次のクラスで構成される TransX Utility Application Program Interface について説明します。

- loader クラス
- TransX インタフェース

loader クラス

loader の説明

このクラスは、TransX インタフェースをインスタンス化するメソッドを提供します。ロード操作はこのインタフェースを介して実行されます。

loader の構文

```
oracle.xml.transx.loader
```

loader のメソッド

表 12-4 loader のメソッドの概要

メソッド	説明
getLoader()（12-4 ページ）	TransX のインスタンスを取得します。
main()（12-5 ページ）	コマンドライン・インタフェースです。

getLoader()

説明

TransX のインスタンスを取得します。TransX インタフェースも参照してください。

構文

```
public static TransX getLoader();
```

main()

説明
コマンドライン・インタフェースの main ファンクションです。

構文
```
public static void main( String[] args);
```

パラメータ	説明
args	コマンドライン・パラメータ。

TransX インタフェース

TransX の説明
このインタフェースは、データのロード・ツール API です。

TransX の構文
```
public interface TransX
```

TransX のメソッド

表 12-5 TransX のメソッドの概要

メソッド	説明
close()（12-6 ページ）	使用済のリソースの再生を支援します。
load()（12-6 ページ）	ファイルまたは URL のデータセットをロードします。
open()（12-7 ページ）	データ・ロード・セッションを開始します。
setLoadingMode()（12-7 ページ）	複製に操作モードを設定します。
setPreserveWhitespace()（12-8 ページ）	空白を重要なものとして処理するようにローダーに指示します。
setValidationMode()（12-8 ページ）	検証モードを設定します。

表 12-5 TransX のメソッドの概要

メソッド	説明
unload()（12-9 ページ）	データセットをアンロードします。
validate()（12-9 ページ）	ファイルまたは URL のデータセットを検証します。

close()

説明

使用済のリソースの再生を支援します。このメソッドは、ロードの完了後にコールする必要があります。open() も参照してください。

java.sql.SQLException をレポートします。

構文

```
public void close();
```

load()

説明

ファイルのデータセットをロードします。事前に open() をコールすることによって、データベース接続を確立しておく必要があります。挿入または更新された行要素の合計数を戻します。java.lang.Exception をレポートします。open() も参照してください。次の表に、オプションを示します。

構文	説明
public int load(String file);	file パラメータで指定された XML ファイルをロードします。
public int load(URL url);	url が参照する XML 文書をロードします。

パラメータ	説明
file	ファイル名。
url	URL 文字列。

open()

説明

データ・ロード・セッションを開始します。このメソッドは、後続の load() のコールに使用される、指定された JDBC URL へのデータベース接続を確立します。

java.sql.SQLException をレポートします。load() も参照してください。

構文

```
public void open( String constr,
                  String user,
                  String pwd);
```

パラメータ	説明
constr	次の形式のデータベース URL。 jdbc:Oracle:<driver_type>:@[additional_parameters]
user	接続を確立したデータベース・ユーザー。
pwd	ユーザーのパスワード。

setLoadingMode()

説明

複製に操作モードを設定します。ロードするデータセットの値と同じキー列の値を持つ既存の行がデータベースに 1 つ以上存在する場合、ローダーは、このメソッドで指定した現行のロード・モードによって、次の操作を実行します。

- 重複行をスキップします（デフォルト）。
- 重複行を更新します。
- 例外をレポートします。

構文

```
public void setLoadingMode(int mode);
```

パラメータ	説明
mode	ロードのモード。次の定数は、Loading Mode クラスで定義されます。 SKIP_DUPLICATES（デフォルト） UPDATE_DUPLICATES EXCEPTION_ON_DUPLICATES

setPreserveWhitespace()

説明

空白を重要なものとして処理するようにローダーに指示します。デフォルトでは、フラグは false です。このコールを実行しなかった場合、または flag 引数を false に指定した場合、ローダーはデータセット内の空白文字の型を無視し、それらを空白文字としてロードします。データセット内の連続した空白文字は、1 つの空白文字として処理されます。

構文

```
public void setPreserveWhitespace( boolean flag);
```

パラメータ	説明
flag	例外の場合は true、例外でない場合は false。

setValidationMode()

説明

検証モードを設定します。デフォルトでは、フラグは true に設定されます。このコールを実行しなかった場合、または flag 引数を true に指定した場合、ローダーは load() をコールするたびに正規スキーマ定義に対してデータセット・フォーマットの検証を実行します。検証モードは、データセットが検証済である場合にのみ無効にする必要があります。validate() も参照してください。

構文

```
public void setValidationMode(boolean flag);
```

パラメータ	説明
flag	ローダーを検証するかどうかを判別します。

unload()

説明

データセットをアンロードします。次の表に、オプションを示します。データセットを XML で戻します。`java.lang.Exception` をレポートします。次の表に、オプションを示します。

構文	説明
public java.io.Reader unload(String table, String[] columns)*;	読込み可能なリーダーを戻します。
public void unload(String table, String[] columns, Writer out)*;	アンロード済のデータセットを指定されたライターに書き込みます。

パラメータ	説明
table	表名。
columns	列名（`null` の場合は * を意味します）。
out	書込み先のライター。

validate()

説明

ファイルまたは URL のデータセットを検証します。データセットの検証後、検証モードを安全に無効にできます。検証のためにインスタンス・ドキュメントを開く必要はありません。`setValidationMode()` も参照してください。正常に検証が実行された場合は `true`、実行されなかった場合は `false` を戻します。
`oracle.xml.parser.schema.XSDException` をレポートします。

構文

```
public boolean validate( String datasrc);
```

パラメータ	説明
datasrc	ファイル名または URL 文字列。

第II部

XDK for C パッケージ

第 II 部に含まれる章は、次のとおりです。

13

XML Parser for C

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』

Parser API

この XML プロセッサ（またはパーサー）の C 実装は、W3C の XML 仕様（改訂 REC-xml-19980210）に準拠しており、XML データを読み込むために XML プロセッサに必要な動作、およびアプリケーションに提供する必要がある情報を含んでいます。

コール順序

単一のドキュメントを解析する場合は、次の順にコールします。

```
xmlinit, xmlparsexxx, xmlterm
```

複数のドキュメントを解析し、最新のドキュメント・データのみを使用可能にする必要がある場合は、次の順にコールします。

```
xmlinit, xmlparsexxx, xmlclean, xmlparsexxx, xmlclean ... xmlterm
```

複数のドキュメントを解析し、すべてのドキュメント・データを使用可能にする必要がある場合は、次の順にコールします。

```
xmlinit, xmlparsexxx, xmlparsexxx ... xmlterm
```

メモリー

独自のメモリー割当てを行う場合は、メモリー・コールバック・ファンクションを使用できます。これらのファンクションを使用する場合は、そのすべてのファンクションを指定する必要があります。

SAX コールバックに渡されたパラメータ用または DOM 解析ツリーを使用して格納されたノードおよびデータ用に割り当てられたメモリーは、次のいずれかの操作が実行されるまで解放されません。

- xmlclean() がコールされる。
- xmlterm() がコールされる。

スレッド・セーフティ

初期化、解析、終了というコール順序の途中でスレッドが無効になると、予測できない動作および結果になる場合があります。

データ型

表 13-1 Parser API のデータ型の概要

データ型	構文	説明
boolean	typedef int boolean;	`true` または `false` のブール値。
oratext	typedef unsigned char oratext;	すべてのデータ・エンコーディングに使用される文字列ポインタ。 必要に応じてキャストします（UTF-16 の場合、`(ub2 *)`）。
string	typedef unsigned char String;	文字列ポインタ（C または C++）。
ub4	typedef unsigned int ub4;	32 ビット以上の符号なし整数。
uword	typedef unsigned int uword;	システム固有の符号なし整数。
xmlacctype	XMLACCESS_UNK /*An access method was specified an a URL, but it was unknown to the XML parser*/ XMLACCESS_FILE /*Filesystem I/O*/ XMLACCESS_HTTP /*HyperText Transport Protocol; the Web)*/ XMLACCESS_FTP /*File Transfer Protocol */ XMLACCESS_GOPHER /*Gopher*/ XMLACCESS_ORADB /*Direct Oracle database access*/ XMLACCESS_STREAM /*User-defined stream*/	XML 文書の XML アクセス・タイプ（HTTP、FTP、File など）の取得に使用可能なアクセス・メソッドの列挙。 詳細は、`xmlaccess()` ファンクションを参照してください。
xmlctx	typedef struct xmlctx xmlctx;	最上位の XML コンテキスト。 xmlctx のコンテンツは内部的に使用されるため、ユーザーからはアクセスできません。

表 13-1 Parser API のデータ型の概要（続き）

データ型	構文	説明

xmlmemcb	struct xmlmemcb
	{
	void *(*alloc)(void *ctx, size_t size);
	void (*free)(void *ctx, void *ptr);
	};
	typedef struct xmlmemcb xmlmemcb;
		メモリー・コールバック構造（オプション）。
		割当てを初期化する必要はありません。`calloc` ではなく、`malloc` と同様に動作します。

xmlnode	typedef struct xmlnode xmlnode;	**注意：**xmlnode のコンテンツは内部的に使用されるため、ユーザーからはアクセスできません。
xmlntype	ELEMENT_NODE = 1 /* element */ ATTRIBUTE_NODE= 2 /* attribute */ TEXT_NODE = 3 /* char data not escaped by CDATA */ CDATA_SECTION_NODE= 4 /* char data escaped by CDATA */ ENTITY_REFERENCE_NODE = 5 /* entity reference */ ENTITY_NODE = 6 /* entity */ PROCESSING_INSTRUCTION_NODE = 7 /* processing instruction */ COMMENT_NODE= 8 /* comment */ DOCUMENT_NODE = 9 /* document */ DOCUMENT_TYPE_NODE= 10 /* DTD */	ノードのタイプの列挙。 解析ツリーのノードのタイプです。13-69 ページの「getNodeType()」を参照してください。名前および値は DOM 仕様に適合します。

表 13-1 Parser API のデータ型の概要（続き）

データ型	構文	説明
	OCUMENT_FRAGMENT_NODE= 11 /* document fragment */ DNOTATION_NODE = 12 /* notation */	
xmlsaxc	struct xmlsaxcb { sword (*startDocument) (void *ctx); sword (*endDocument) (void *ctx); sword (*startElement)(void *ctx, const oratext *name, const struct xmlattrs *attrs); sword (*endElement)(void *ctx, const oratext *name); sword (*characters)(void *ctx, const oratext *ch, size_t len); sword (*ignorableWhitespace)(void *ctx, const oratext *ch, size_t len); sword (*processingInstruction)(void *ctx, const oratext *target, const oratext *data); sword (*notationDecl)(void *ctx, const oratext *name, const oratext *publicId, const oratext *systemId); sword (*unparsedEntityDecl)(void *ctx, const oratext *name, const oratext *publicId, const oratext *systemId, const oratext *notationName); sword (*comment)(void *ctx, const oratext *data);	SAX コールバック構造（SAX のみ）。

表 13-1 Parser API のデータ型の概要（続き）

データ型	構文	説明
	sword (*elementDecl)(void *ctx, const oratext *name, const oratext *content); sword (*attributeDecl)(void *ctx, const oratext *elem, const oratext *attr, const oratext *body); sword (*xmlDecl)(void *ctx, const oratext *version, boolean encoding); /* Following fields are reserved for future use.*/ void (*empty1)(); void (*empty2)(); void (*empty3)(); void (*empty4)(); }; typedef struct xmlsaxcb xmlsaxcb;	

Parser API のファンクションおよびメソッド

表 13-2 Parser API のファンクションおよびメソッドの概要

ファンクションまたはメソッド	説明
freeElements()（13-8 ページ）	割り当てられた要素ノードのリストを解放します。
getEncoding()（13-8 ページ）	ドキュメントのエンコーディングを戻します。
isSingleChar()（13-9 ページ）	ドキュメント・データがシングルバイトかマルチバイトかを判別します。
isStandalone()（13-9 ページ）	ドキュメントが単独かどうかを判別します。
isUnicode()（13-10 ページ）	ドキュメント・データが Unicode かどうかを判別します。
saveString()（13-10 ページ）	メモリーを割り当て、null で終了するシングルバイトまたはマルチバイトの文字列を XML 文字列プールに保存します。
saveString2()（13-11 ページ）	メモリーを割り当て、null で終了する Unicode 文字列を XML 文字列プールに保存します。
printBuffer()（13-11 ページ）	XML ツリーの表現をバッファに出力します。
printSize()（13-12 ページ）	XML ツリーの出力表現のサイズを戻します。
printStream()（13-13 ページ）	XML ツリーの出力表現をファイル・ストリームに書き込みます。
setDocOrder()（13-13 ページ）	現行のドキュメントの各ノードに対してドキュメント順序を設定します。
xmlaccess()（13-14 ページ）	指定されたアクセス・メソッドに対して I/O コールバック・ファンクションを設定します。
xmlclean()（13-16 ページ）	前回の解析中に使用されたすべてのメモリーを解放します。
xmlinit()（13-16 ページ）	XML パーサーを初期化します。
xmlinitenc()（13-18 ページ）	DOM データのエンコーディングを指定して、XML パーサーを初期化します。
xmlLocation()（13-20 ページ）	解析中に現在の位置を戻します。
xmlparse()（13-20 ページ）	URI を解析します。
xmlparsebuf()（13-22 ページ）	バッファを解析します。
xmlparsedtd()（13-23 ページ）	外部 DTD を解析します。
xmlparsefile()（13-25 ページ）	ファイルを解析します。

表 13-2 Parser API のファンクションおよびメソッドの概要（続き）

ファンクションまたはメソッド	説明
xmlparsestream()（13-27 ページ）	ユーザー定義のストリームを解析します。
xmlterm()（13-29 ページ）	XML パーサーを終了します。
xmlwhere()（13-29 ページ）	エラー位置の情報を戻します。

freeElements()

説明

割り当てられた要素ノードのリストを解放します。主に、getElementsByTagName() によって作成されたリストの解放に使用されます。

構文

```
void freeElements( xmlctx *ctx,
                   xmlnodes *list);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
list	(IN)	解放するノードのリスト。

getEncoding()

説明

DOM/SAX データ・エンコーディングの IANA/MIME 名（「ASCII」、「ISO-8859-1」、「UTF-8」、「UTF-16」など）を戻します。isSingleChar()（データがシングルバイトかマルチバイトかを判別）または isUnicode()（データが Unicode（UTF-16）かどうかを判別）も参照してください。データのエンコーディングは、開始時にユーザーが指定します。

構文

```
oratext *getEncoding( xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。

isSingleChar()

説明

現行のコンテキストに対するデータを、シングルバイト文字（ASCII、ISO-8859、EBCDIC など）またはマルチバイト文字（UTF-8 または Unicode）のどちらでエンコーディングするかを指定するフラグを戻します。getEncoding() を参照してください。このファンクションは、データのエンコーディング名を戻します。

構文

```
boolean isSingleChar( xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。

isStandalone()

説明

ドキュメントのスタンドアロン・フラグの値を戻します。このファンクションは、XML 宣言で指定されているとおり、ドキュメントのスタンドアロン・フラグのブール値を戻します。

構文

```
boolean isStandalone( xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。

isUnicode()

説明

Unicode（UCS2）エンコーディングのフラグを戻します。このファンクションは、isSingleChar() に類似しています。

構文

```
boolean isUnicode(xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。

saveString()

説明

メモリーを割り当て、null で終了するシングルバイトまたはマルチバイトの文字列を XML 文字列プールに保存します。この方法で文字列を保存すると、文字列は、最もコンパクトな方法で 1 つのプールに一列に並んで格納されるため、個々に解放することはできません。xmlclean() または xmlterm() をコールしてプール全体を解放した場合にのみ、メモリーが再利用されます。Unicode 文字列を保存するには、saveString2() を使用します。

構文

```
oratext *saveString( xmlctx *ctx,
                     oratext *str);
```

パラメータ	IN/OUT	説明
ctx	(IN)	LPX コンテキスト。
str	(IN)	シングルバイトまたはマルチバイトの文字列に対するポインタ。

saveString2()

説明

メモリーを割り当て、null で終了する Unicode 文字列を XML 文字列プールに保存します。Unicode 文字列は、1 つではなく 2 つの null バイトで終了することに注意してください。この方法で文字列を保存すると、文字列は、最もコンパクトな方法で 1 つのプールに一列に並んで格納されるため、個々に解放することはできません。xmlclean() または xmlterm() をコールしてプール全体を解放した場合にのみ、メモリーが再利用されます。シングルバイトまたはマルチバイトの文字列を保存するには、saveString() を使用します。

構文

```
ub2 *saveString2( xmlctx *ctx,
                  ub2 *ustr);
```

パラメータ	IN/OUT	説明
ctx	(IN)	LPX コンテキスト。
ustr	(IN)	Unicode 文字列に対するポインタ。

printBuffer()

説明

指定されたノードをルートとした XML ツリーの出力表現を作成し、宛先バッファに出力します。インデントは、レベルおよびステップで制御されます。ステップは、新しいレベルをインデントする空白の数です。レベルは、開始レベル（最上位は 0（ゼロ））です。

構文

```
void printBuffer( oratext *buffer,
                  size_t bufsiz,
                  xmlnode *node,
                  uword step,
                  uword level);
```

パラメータ	IN/OUT	説明
buffer	(IN)	出力先の宛先バッファ。
bufsiz	(IN)	宛先バッファのサイズ。
node	(IN)	出力する XML ツリーのルート・ノード。
step	(IN)	新しいレベルをインデントする空白の数。
level	(IN)	インデントの開始レベル。

printSize()

説明

指定されたノードをルートとした XML ツリーの出力表現のサイズを戻します。printBuffer() の場合と同様に、インデントは、レベルおよびステップで制御されます。ステップは、新しいレベルをインデントする空白の数です。レベルは、開始レベル（最上位は 0（ゼロ））です。このファンクションを使用して、printBuffer() に必要なバッファのサイズを事前に計算できます。

構文

```
size_t printSize( xmlnode *node,
                  uword step,
                  uword level);
```

パラメータ	IN/OUT	説明
node	(IN)	出力する XML ツリーのルート・ノード。
step	(IN)	新しいレベルをインデントする空白の数。
level	(IN)	インデントの開始レベル。

printStream()

説明

（指定されたノードをルートとした）XML ツリーの出力表現を、標準入出力ストリーム（FILE*）に書き込みます。このファンクションは、バッファではなくストリームに出力することを除き、printBuffer() と同じです。インデントは、レベルおよびステップで制御されます。ステップは、新しいレベルをインデントする空白の数です。レベルは、開始レベル（最上位は 0（ゼロ））です。

構文

```
void printStream( FILE *stream,
                  xmlnode *node,
                  uword step,
                  uword level);
```

パラメータ	IN/OUT	説明
stream	(IN)	書込み先の出力ストリーム。
node	(IN)	出力する XML ツリーのルート・ノード。
step	(IN)	新しいレベルをインデントする空白の数。
level	(IN)	インデントの開始レベル。

setDocOrder()

説明

現行のドキュメントの各ノードに対してドキュメント順序を設定します。XSLT 処理を行うには、最終ドキュメントに対してこのファンクションをコールする必要があります。このファンクションは、XSLT プロセッサによって自動的にコールされるため、ユーザーがこのファンクションをコールする必要はありません。

構文

```
ub4 setDocOrder(xmlctx *ctx, ub4 start_id);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
start_id	(IN)	割り当てる初期 ID 番号。

xmlaccess()

説明

指定されたアクセス・メソッドに対して I/O コールバック・ファンクションを設定します。

構文

```
uword xmlaccess( xmlctx *ctx,
                 xmlacctype access,
                 XML_OPENF((*openf)),
                 XML_CLOSEF((*closef)),
                 XML_READF((*readf)));
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
access	(IN)	アクセス・メソッドの列挙（XMLACCESS_xxx）。
openf	(IN)	入力ソースをオープンするコールバック・ファンクション。
closef	(IN)	入力ソースをクローズするコールバック・ファンクション。
readf	(IN)	入力ソースを読み込むコールバック・ファンクション。

コメント

指定されたアクセス・メソッドに対して I/O コールバック・ファンクションを設定します。ほとんどのメソッドに組込みコールバック・ファンクションが含まれているため、ユーザーが指定する必要はありません。ただし、XMLACCESS_STREAM の場合は、ユーザー自身がストリーム・コールバック・ファンクションを設定する必要があります。

次の 3 つのコールバック・ファンクションを起動すると、入力ソースのオープン、クローズおよび読込みが行われます。これらのファンクションは、ファンクション・プロトタイプ・マクロ（XML_OPENF、XML_CLOSEF および XML_READF）を使用して宣言しておく必要があります。

XML_OPENF は、オープン用ファンクションで、入力ソースをオープンするために 1 回コールされます。xmlihdl 共用体で永続ハンドルを設定する必要があります。永続ハンドルには、汎用ポインタ（void *）または整数（UNIX ファイルまたはソケット・ハンドル）のいずれかを使用できます。このファンクションは、正常に実行されると XMLERR_OK を戻します。

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
ih	(OUT)	オープンされたハンドルが置かれる場所。
length	(OUT)	既知の場合は、入力データの合計バイト数（不明の場合は 0（ゼロ））。
parts	(IN)	構成要素に分割された URL（不透明ポインタ）。
path	(IN)	オープンされる完全な URL。

XML_CLOSEF は、クローズ・ファンクションで、オープンされたソースをクローズし、リソースを解放します。

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
ih	(IN)	オープンされたハンドル。

XML_READF は、リーダー・ファンクションで、オープンされたソースからバッファにデータを読み込み、読み込んだバイト数を戻します。

- 0（ゼロ）以下の場合は、EOI 条件が示されます。
- 0 を超える場合は、EOI フラグによって最終データかどうかが判別されます。

EOI が戻された場合は、対応するクローズ・ファンクションが自動的にコールされます。

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
dest	(OUT)	データの読込み先の宛先バッファ。
destsize	(IN)	宛先バッファのサイズ。
eoi	(OUT)	情報の終わりが検出されたかどうかを判別します。
ih	(IN)	読込みを行う入力ハンドル共用体。
nraw	(OUT)	読み込まれるバイト数。
path	(IN)	エラー・メッセージで使用するために提供されるソースの URL。

xmlclean()

説明

XML パーサー内でメモリーを再利用します。ただし、システムには解放しません。xmlterm() によって、最後にすべてのメモリーがシステムに解放されます。解析と解析の間で xmlclean() がコールされない場合は、前回のドキュメントによって使用されたデータが割り当てられ、このデータに対するポインタが有効状態のままになります。そのため、複数のドキュメントのデータに同時にアクセスできます。ただし、DOM で操作できるのは現行のドキュメントのみです。

1 つのコンテキスト内で、一度に 1 つのドキュメントのデータのみにアクセスできるようにするには、新しく解析を開始する前に clear() をコールする必要があります。

構文

```
void xmlclean( xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。

xmlinit()

説明

XML パーサーを初期化します。このファンクションは、解析を行う前にコールする必要があります。

- このコールの C バージョンは、正常に初期化された場合は XML コンテキストを戻し、エラーが発生した場合はユーザーの err 引数を設定します。通常、エラー・コードが 0（ゼロ）の場合は正常に初期化されたことを示し、0（ゼロ）以外は問題が発生したことを示します。
- このファンクションは、1 つ以上の XML ファイルの処理を開始する前に 1 回のみコールします。xmlterm() は、XML ファイルのすべての処理が完了した後にコールします。
- C の場合、err 以外のすべての引数を null にできます。C++ の場合、すべての引数はデフォルト値を持ち、不要な場合は省略できます。
- デフォルトでは、データ・エンコーディングは UTF-8 です。ドキュメントがシングルバイトの場合は、パフォーマンスを向上させるようにエンコーディングを設定する必要があります。
- メッセージは、msghdlr が指定されないかぎり、stderr に出力されます。

- デフォルトでは、DOM ツリーが構築されます。SAX モードの場合は、saxcb コールバックを設定する必要があります。不要な SAX コールバック・ファンクションは、すべて null に設定できます。
- 独自のメモリー割当てを行う場合は、memcb で指定されるメモリー・コールバック・ファンクションを使用できます。alloc() および free() ファンクションの両方を指定する必要があります。
- msgctx、saxcbctx および memcbctx パラメータは、それぞれメッセージ・ハンドラ、SAX ファンクションまたはメモリー・ファンクションのユーザー定義のコールバック・ルーチンに情報を渡すために定義および設定できます。ユーザー定義のコールバック・ファンクションに追加情報を渡す必要がない場合は、これらのパラメータを null に設定する必要があります。
- lang パラメータは、エラー・メッセージの言語を判別します。デフォルトは「American」です。
- 正常に初期化された場合は XML コンテキストを戻し、エラーが発生した場合はユーザーの err 引数を設定します。エラー・コードが 0（ゼロ）の場合は正常に初期化されたとを示し、0（ゼロ）以外は問題が発生したことを示します。

構文

```
xmlctx *xmlinit( uword *err,
                 const oratext *incoding,
                 XML_MSGHDLRF((*msghdlr)),
                 void *msgctx,
                 const xmlsaxcb *saxcb,
                 void *saxcbctx,
                 const xmlmemcb *memcb,
                 void *memcbctx,
                 const oratext *lang);
```

パラメータ	IN/OUT	説明
err	(OUT)	数値エラー・コード（エラーが発生した場合）。
incoding	(IN)	デフォルトの入力エンコーディング。
msghdlr	(IN)	エラー・メッセージ・ハンドラのファンクション。
msgctx	(IN)	エラー・メッセージ・ハンドラに対するユーザー・コンテキスト。
saxcb	(IN)	ファンクション・ポインタを含む SAX コールバック構造。
saxcbctx	(IN)	SAX コールバックに対するユーザー・コンテキスト。
memcb	(IN)	メモリー・ファンクションのコールバック構造。

パラメータ	IN/OUT	説明
memcbctx	(IN)	メモリー・ファンクションのコールバックに対するユーザー・コンテキスト。
lang	(IN)	エラー・メッセージの言語。

エラー・コード	説明
XMLERR_BAD_ENCODING	エンコーディングが不明です。エンコーディングの IANA/MIME 名を使用して、グローバリゼーション・サポート・データが存在することを確認してください。
XMLERR_INVALID_LANG	エラー・メッセージに指定されている言語が不明です。
XMLERR_INVALID_MEMCB	メモリー・コールバック構造（memcb）は指定されていますが、alloc および free ファンクション・ポインタが指定されていません。
XMLERR_LEH_INIT	LEH（catch/throw）パッケージを初期化できません。内部エラーです。オラクル社カスタマ・サポート・センターに連絡してください。
XMLERR_NLS_INIT	National Language Support（NLS）パッケージを初期化できません。インストールまたは構成に問題がある可能性があります。

xmlinitenc()

説明

DOM データ・エンコーディングを指定して、XML パーサーを初期化します。このファンクションは、解析を開始する前にコールする必要があります。SAX の `xmlinit()` と同じですが、データ・エンコーディングを指定できます。

構文

```
xmlctx *xmlinitenc( uword *err,
                    const oratext *incoding,
                    const oratext *outcoding,
                    XML_MSGHDLRF((*msghdlr)),
                    void *msgctx,
                    const xmlsaxcb *saxcb,
                    void *saxcbctx,
                    const xmlmemcb *memcb,
                    void *memcbctx,
                    const oratext *lang);
```

パラメータ	IN/OUT	説明
err	(OUT)	数値エラー・コード（エラーが発生した場合）。
incoding	(IN)	デフォルトの入力エンコーディング。
outcoding	(IN)	出力（DIM/SAX データ）キャラクタ・セットのエンコーディング。
msghdlr	(IN)	エラー・メッセージ・ハンドラのファンクション。
msgctx	(IN)	エラー・メッセージ・ハンドラに対するユーザー・コンテキスト。
saxcb	(IN)	ファンクション・ポインタを含む SAX コールバック構造。
saxcbctx	(IN)	SAX コールバックに対するユーザー・コンテキスト。
memcb	(IN)	メモリー・ファンクションのコールバック構造。
memcbctx	(IN)	メモリー・ファンクションのコールバックに対するユーザー・コンテキスト。
lang	(IN)	エラー・メッセージの言語。

エラー・コード	説明
XMLERR_BAD_ ENCODING	エンコーディングが不明です。エンコーディングの IANA/MIME 名を使用して、グローバリゼーション・サポート・データが存在することを確認してください。
XMLERR_INVALID_LANG	エラー・メッセージに指定されている言語が不明です。
XMLERR_INVALID_ MEMCB	メモリー・コールバック構造（memcb）は指定されていますが、alloc および free ファンクション・ポインタが指定されていません。
XMLERR_LEH_INIT	LEH（catch/throw）パッケージを初期化できません。内部エラーです。オラクル社カスタマ・サポート・センターに連絡してください。
XMLERR_NLS_INIT	National Language Support パッケージを初期化できません。インストールまたは構成に問題がある可能性があります。

xmlLocation()

説明

解析中に現在のソースの位置を戻します。このファンクションは、いつでもコールできます。ただし、解析中にコールしなかった場合は、path および line の両方に対して 0（ゼロ）が戻されます。

構文

```
uword xmlLocation( xmlctx *ctx,
                   ub4 *line,
                   oratext **path);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。
line	(OUT)	現在の行番号。
path	(OUT)	現在のソースのパスまたは URL。

xmlparse()

説明

URI によって指定された入力ドキュメントに対して XML パーサーを起動します。XML パーサーは、最初に、xmlinit() または xmlinitenc() のコールによって正常に初期化されている必要があります。パーサー・オプションは、flags マスクに OR 計算されたフラグ・ビットとして指定します。

デフォルトの入力エンコーディングは incoding に指定できます。これによって、xmlinit() に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。

構文

```
uword xmlparse( xmlctx *ctx,
                const oratext *uri,
                const oratext *incoding,
                ub4 flags);
```

パラメータ	IN/OUT	説明
ctx	(IN/OUT)	XML パーサー・コンテキスト。
uri	(IN)	XML 文書の URI。
incoding	(IN)	デフォルトの入力エンコーディング。
flags	(IN)	パーサー・オプションのフラグ・ビットのマスク。

パーサー・フラグ・オプション	説明
XML_DTD_ONLY	外部サブセットを解析します。これは、`xmlparsedtd()` をコールする場合と同様に、完全な XML 文書ではなく外部サブセット（DTD）を解析します。主に、Class Generator によって使用され、完全な文書を必要とすることなく DTD からクラスを生成できます。
XML_FLAG_DISCARD_ WHITESPACE	不要な空白（行の終わりなど）を削除します。デフォルトでは、空白がレポートされ、無視可能な空白が示されます。このオプションによって、要素の終了タグと次の要素の開始タグの間のすべての空白が削除されます。
XML_FLAG_STOP_ON_ WARNING	警告が発生すると検証を停止します。このフラグが設定されていないかぎり、検証で発生した問題は警告（致命的ではない）とみなされます。このフラグを設定すると、最初の警告後に検証が停止します。
XML_FLAG_VALIDATE	検証を開始します。デフォルトでは、整形式であるかどうかのみが確認されます。
XML_FORCE_INCODING	入力ドキュメントを強制的にエンコーディング「incoding」で解析します。デフォルトの入力エンコーディングを incoding に指定できます。これによって、xmlinit に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。`XML_FLAG_FORCE_INCODING` を設定すると、ドキュメントは「incoding」として解析されます。
XML_WARN_DUPLICATE_ ENTITY	複製エンティティ宣言が検出された場合、警告が発生します。通常、複製エンティティ宣言は、警告を発生させることなく無視されます。このフラグを設定すると、警告が発生します。

xmlparsebuf()

説明

バッファに対して XML パーサーを起動します。XML パーサーは、最初に、xmlinit() または xmlinitenc() のコールによって正常に初期化されている必要があります。パーサー・オプションは、flags マスクに OR 計算されたフラグ・ビットとして指定します。

デフォルトの入力エンコーディングは incoding に指定できます。これによって、xmlinit() に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。

構文

```
uword xmlparsebuf( xmlctx *ctx,
                   const oratext *buffer,
                   size_t len,
                   const oratext *incoding,
                   ub4 flags);
```

パラメータ	IN/OUT	説明
ctx	(IN/OUT)	XML パーサー・コンテキスト。
buffer	(IN)	入力バッファ名。
len	(IN)	バッファ長。
incoding	(IN)	デフォルトの入力エンコーディング。
flags	(IN)	パーサー・オプションのフラグ・ビットのマスク。

パーサー・フラグ・オプション	説明
XML_DTD_ONLY	外部サブセットを解析します。これは、xmlparsedtd() をコールする場合と同様に、完全な XML 文書ではなく外部サブセット（DTD）を解析します。主に、Class Generator によって使用され、完全な文書を必要とすることなく DTD からクラスを生成できます。

パーサー・フラグ・オプション	説明
XML_FLAG_DISCARD_WHITESPACE	不要な空白（行の終わりなど）を削除します。デフォルトでは、空白がレポートされ、無視可能な空白が示されます。このオプションによって、要素の終了タグと次の要素の開始タグの間のすべての空白が削除されます。
XML_FLAG_STOP_ON_WARNING	警告が発生すると検証を停止します。このフラグが設定されていないかぎり、検証で発生した問題は警告（致命的ではない）とみなされます。このフラグを設定すると、最初の警告後に検証が停止します。
XML_FLAG_VALIDATE	検証を開始します。デフォルトでは、整形式であるかどうかのみが確認されます。
XML_FORCE_INCODING	入力ドキュメントを強制的にエンコーディング「incoding」で解析します。デフォルトの入力エンコーディングを incoding に指定できます。これによって、xmlinit に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。 `XML_FLAG_FORCE_INCODING` を設定すると、ドキュメントは「incoding」として解析されます。
XML_WARN_DUPLICATE_ENTITY	複製エンティティ宣言が検出された場合、警告が発生します。通常、複製エンティティ宣言は、警告を発生させることなく無視されます。このフラグを設定すると、警告が発生します。

xmlparsedtd()

説明

完全なドキュメントではなく、外部 DTD ファイルに対して XML パーサーを起動します。主に Class Generator によって使用され、完全なドキュメントを必要とすることなく、DTD からクラスを生成します。XML パーサーは、最初に、`xmlinit()` または `xmlinitenc()` のコールによって正常に初期化されている必要があります。パーサー・オプションは、`flags` マスクに OR 計算されたフラグ・ビットとして指定します。

デフォルトの入力エンコーディングは `incoding` に指定できます。これによって、`xmlinit()` に指定された `incoding` がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、`incoding` であると想定されます。IANA/MIME エンコーディング名には「`UTF-8`」、「`ASCII`」などを使用する必要があります。

構文

```
uword xmlparsedtd( xmlctx *ctx,
                   const oratext *filename,
                   oratext *name,
                   const oratext *incoding,
                   ub4 flags);
```

パラメータ	IN/OUT	説明
ctx	(IN/OUT)	XML パーサー・コンテキスト。
filename	(IN)	外部サブセットのファイル名。
name	(IN)	DTD 名。
incoding	(IN)	デフォルトの入力エンコーディング。
flags	(IN)	パーサー・オプションのフラグ・ビットのマスク。

パーサー・フラグ・オプション	説明
XML_DTD_ONLY	外部サブセットを解析します。これは、xmlparsedtd をコールする場合と同様に、完全な XML 文書ではなく外部サブセット（DTD）を解析します。主に、Class Generator によって使用され、完全な文書を必要とすることなく DTD からクラスを生成できます。
XML_FLAG_DISCARD_WHITESPACE	不要な空白（行の終わりなど）を削除します。デフォルトでは、空白がレポートされ、無視可能な空白が示されます。このオプションによって、要素の終了タグと次の要素の開始タグの間のすべての空白が削除されます。
XML_FLAG_STOP_ON_WARNING	警告が発生すると検証を停止します。このフラグが設定されていないかぎり、検証で発生した問題は警告（致命的ではない）とみなされます。このフラグを設定すると、最初の警告後に検証が停止します。
XML_FLAG_VALIDATE	検証を開始します。デフォルトでは、整形式であるかどうかのみが確認されます。

パーサー・フラグ・オプション	説明
XML_FORCE_INCODING	入力ドキュメントを強制的にエンコーディング「incoding」で解析します。デフォルトの入力エンコーディングをincoding に指定できます。これによって、xmlinit に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。XML_FLAG_FORCE_INCODING を設定すると、ドキュメントは「incoding」として解析されます。
XML_WARN_DUPLICATE_ENTITY	複製エンティティ宣言が検出された場合、警告が発生します。通常、複製エンティティ宣言は、警告を発生させることなく無視されます。このフラグを設定すると、警告が発生します。

xmlparsefile()

説明

ファイル・システム内のドキュメントに対して XML パーサーを起動します。XML パーサーは、最初に、xmlinit() または xmlinitenc() のコールによって正常に初期化されている必要があります。パーサー・オプションは、flags マスクに OR 計算されたフラグ・ビットとして指定します。

デフォルトの入力エンコーディングは incoding に指定できます。これによって、xmlinit() に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。

構文

```
uword xmlparsefile( xmlctx *ctx,
                    const oratext *path,
                    const oratext *incoding,
                    ub4 flags);
```

パラメータ	IN/OUT	説明
ctx	(IN/OUT)	XML パーサー・コンテキスト。
path	(IN)	ドキュメントのファイル・システム・パス。
incoding	(IN)	デフォルトの入力エンコーディング。
flags	(IN)	パーサー・オプションのフラグ・ビットのマスク。

パーサー・フラグ・オプション	説明
XML_DTD_ONLY	外部サブセットを解析します。これは、xmlparsedtd をコールする場合と同様に、完全な XML 文書ではなく外部サブセット（DTD）を解析します。主に、Class Generator によって使用され、完全な文書を必要とすることなく DTD からクラスを生成できます。
XML_FLAG_DISCARD_WHITESPACE	不要な空白（行の終わりなど）を削除します。デフォルトでは、空白がレポートされ、無視可能な空白が示されます。このオプションによって、要素の終了タグと次の要素の開始タグの間のすべての空白が削除されます。
XML_FLAG_STOP_ON_WARNING	警告が発生すると検証を停止します。このフラグが設定されていないかぎり、検証で発生した問題は警告（致命的ではない）とみなされます。このフラグを設定すると、最初の警告後に検証が停止します。
XML_FLAG_VALIDATE	検証を開始します。デフォルトでは、整形式であるかどうかのみが確認されます。
XML_FORCE_INCODING	入力ドキュメントを強制的にエンコーディング「incoding」で解析します。デフォルトの入力エンコーディングを incoding に指定できます。これによって、xmlinit に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。`XML_FLAG_FORCE_INCODING` を設定すると、ドキュメントは「incoding」として解析されます。
XML_WARN_DUPLICATE_ENTITY	複製エンティティ宣言が検出された場合、警告が発生します。通常、複製エンティティ宣言は、警告を発生させることなく無視されます。このフラグを設定すると、警告が発生します。

xmlparsestream()

説明

ユーザー定義のストリームに対して XML パーサーを起動します。XML パーサーは、最初に、xmlinit() または xmlinitenc() のコールによって正常に初期化されている必要があります。パーサー・オプションは、flags マスクに OR 計算されたフラグ・ビットとして指定され、パーサーのデフォルト動作をオーバーライドします。

デフォルトの入力エンコーディングは incoding に指定できます。これによって、xmlinit() に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。

このファンクションのコンテキストでは、ストリームはユーザー定義のエンティティです。stream はストリーム / コンテキスト・ポインタで、I/O コールバック・ファンクションに渡されます。パーサーは、ストリームを直接参照しません。最初に、アクセス・メソッド XMLACCESS_STREAM の I/O コールバック・ファンクションを設定する必要があります。各コールバック・ファンクションでは、ストリーム（またはストリーム・コンテキスト）・ポインタを ihdl 構造の ptr_xmlihdl メモリーとして使用できます。その意味および使用方法は、ユーザーが定義します。

構文

```
uword xmlparsestream( xmlctx *ctx,
                      const void *stream,
                      const oratext *incoding,
                      ub4 flags);
```

パラメータ	IN/OUT	説明
ctx	(IN/OUT)	XML パーサー・コンテキスト。
stream	(IN)	ユーザー定義のストリーム・オブジェクトに対するポインタ。
incoding	(IN)	デフォルトの入力エンコーディング。
flags	(IN)	パーサー・オプションのフラグ・ビットのマスク。

パーサー・フラグ・オプション	説明
XML_DTD_ONLY	外部サブセットを解析します。これは、xmlparsedtd をコールする場合と同様に、完全な XML 文書ではなく外部サブセット（DTD）を解析します。主に、Class Generator によって使用され、完全な文書を必要とすることなく DTD からクラスを生成できます。
XML_FLAG_DISCARD_WHITESPACE	不要な空白（行の終わりなど）を削除します。デフォルトでは、空白がレポートされ、無視可能な空白が示されます。このオプションによって、要素の終了タグと次の要素の開始タグの間のすべての空白が削除されます。
XML_FLAG_STOP_ON_WARNING	警告が発生すると検証を停止します。このフラグが設定されていないかぎり、検証で発生した問題は警告（致命的ではない）とみなされます。このフラグを設定すると、最初の警告後に検証が停止します。
XML_FLAG_VALIDATE	検証を開始します。デフォルトでは、整形式であるかどうかのみが確認されます。
XML_FORCE_INCODING	入力ドキュメントを強制的にエンコーディング「incoding」で解析します。デフォルトの入力エンコーディングを incoding に指定できます。これによって、xmlinit に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。`XML_FLAG_FORCE_INCODING` を設定すると、ドキュメントは「incoding」として解析されます。
XML_WARN_DUPLICATE_ENTITY	複製エンティティ宣言が検出された場合、警告が発生します。通常、複製エンティティ宣言は、警告を発生させることなく無視されます。このフラグを設定すると、警告が発生します。

xmlterm()

説明

XML パーサーを終了します。このファンクションは、xmlinit へのコールとメイン・プログラムの終了の間にコールする必要があります。

構文

```
uword xmlterm(xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN/OUT)	XML パーサー・コンテキスト。

コメント

このファンクションは、XML コンテキストを終了します。パーサーが使用したすべてのメモリーが、システムに戻されます。コンテキストは再利用できません。さらに解析を行う場合は、新しいコンテキストを作成する必要があります。XML パーサーは、xmlinit を再度コールして新しいコンテキストを取得しないかぎり、コールできません。メモリーをシステムに戻すことなく内部的に再利用し、コンテキストを継続して使用可能にする xmlclean() ファンクションと比較してください。

xmlwhere()

説明

最後または現在のエラーのエラー位置情報を戻します。現在のエラーが発生したソースの位置を戻します。このファンクションは、エラー・ハンドラ内からコールする必要があります。ソースの位置はスタックです。たとえば、インデックス 0（ゼロ）は最も低いエラー発生レベルです。インデックス 1 は囲みエンティティです。スタック全体を示すには、インデックスを、false が戻された場合に停止して、0 から N にループする必要があります。

構文

```
boolean xmlwhere( xmlctx *ctx,
                  ub4 *line,
                  oratext **path,
                  uword idx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。
line	(OUT)	エラーが発生した行番号。
path	(OUT)	エラーが発生したソースのパスまたは URL。
idx	(IN)	スタック内のエラー番号（0（ゼロ）から開始）。

W3C の SAX API

W3C の SAX API の説明

SAX はイベントベースの XML 解析用の標準インタフェースで、XML-DEV メーリング・リストのメンバーが共同開発したものです。

SAX を使用するには、ファンクション・ポインタを使用して xmlsaxcb 構造を初期化し、xmlinit コールに渡します。ユーザー定義のコンテキスト構造体に対するポインタを含めることもできます。そのコンテキスト・ポインタは、各 SAX ファンクションに渡されます。

W3C の SAX API のコールバック構造

```
typedef struct
{
   sword (*startDocument)(void *ctx);
   sword (*endDocument)(void *ctx);
   sword (*startElement)(void *ctx, const oratext *name,
                         const struct xmlnodes *attrs);
   sword (*endElement)(void *ctx, const oratext *name);
   sword (*characters)(void *ctx, const oratext *ch, size_t len);
   sword (*ignorableWhitespace)(void *ctx, const oratext *ch,
                         size_t len);
   sword (*processingInstruction)(void *ctx, const oratext *target
                         const oratext *data);
   sword (*notationDecl)(void *ctx, const oratext *name,
                         const oratext *publicId,
                         const oratext *systemId);
   sword (*unparsedEntityDecl)(void *ctx, const oratext *name,
                         const oratext *publicId,
                         const oratext *systemId,
                         const oratext *notationName);
   sword (*nsStartElement)(void *ctx, const oratext *qname,
                         const oratext *local, const oratext *nsp,
                         const struct xmlnodes *attrs);
   sword (*comment)(void *ctx, const oratext *data);
   sword (*elementDecl)(void *ctx, const oratext *name,
                         const oratext *content);
   sword (*attributeDecl)(void *ctx, const oratext *elem,
                         const oratext *attr, const oratext *body);
   sword (*xmlDecl)(void *ctx, const oratext *version, boolean encoding);
} xmlsaxcb;
```

W3C の SAX API のデータ構造

表 13-3 データ構造の名前空間 API

データ構造	宣言	説明
oratext*	typedef unsigned char oratext;	データに対するポインタです。Unicode を含む任意のエンコーディングです。必要に応じてキャストします。
sword	typedef signed int sword;	SAX のユーザー・コールバック・ファンクションの値を戻します。0（ゼロ）は成功を示します。
xmlnodes*	typedef struct xmlnodes xmlnodes;	ノード（要素の属性）のリストに対するポインタです。 **注意：**`xmlnodes*` は不透明型で、直接参照できません。かわりに、DOM ファンクション getAttributeIndex() を使用します。

コールバック・ファンクション

表 13-4 SAX コールバック・ファンクションの概要

ファンクション	説明
characters()（13-33 ページ）	要素内の文字データの通知を受け取ります。
endDocument()（13-33 ページ）	ドキュメントの終わりの通知を受け取ります。
endElement()（13-34 ページ）	要素の終わりの通知を受け取ります。
ignorableWhitespace()（13-34 ページ）	要素の内容にある無視可能な空白の通知を受け取ります。
notationDecl()（13-35 ページ）	表記法宣言の通知を受け取ります。
processingInstruction()（13-35 ページ）	処理命令の通知を受け取ります。
startDocument()（13-36 ページ）	ドキュメントの始まりの通知を受け取ります。
startElement()（13-36 ページ）	要素の始まりの通知を受け取ります。
unparsedEntityDecl()（13-37 ページ）	解析対象外のエンティティ宣言の通知を受け取ります。

表 13-5 Oracle SAX 拡張コールバック・ファンクションの概要

ファンクション	説明
attributeDecl()（13-37 ページ）	DTD 内の要素の属性宣言に関する通知を受け取ります。
comment()（13-38 ページ）	XML 文書内のコメントに関する通知を受け取ります。
elementDecl()（13-38 ページ）	DTD 内の要素宣言に関する通知を受け取ります。
ignorableWhitespace()（13-34 ページ）	要素の名前空間の始まりの通知を受け取ります。

characters()

説明

要素内の文字データの通知を受け取ります。

構文

```
sword (*characters)( void *ctx,
                     const oratext *ch,
                     size_t len);
```

パラメータ	IN/OUT	説明
ctx	(IN)	ユーザー・コンテキスト。
ch	(IN)	文字データに対するポインタ。
len	(IN)	文字データの長さ。

endDocument()

説明

ドキュメントの終わりの通知を受け取ります。

構文

```
sword (*endDocument)( void *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	クライアント・コンテキスト。

endElement()

説明

要素の終わりの通知を受け取ります。

構文

```
sword (*endElement)( void *ctx,
                     const oratext *name);
```

パラメータ	IN/OUT	説明
ctx	(IN)	クライアント・コンテキスト。
name	(IN)	要素型名。

ignorableWhitespace()

説明

要素の内容にある無視可能な空白の通知を受け取ります。

構文

```
sword (*ignorableWhitespace)( void *ctx,
                              const oratext *ch,
                              size_t len);
```

パラメータ	IN/OUT	説明
ctx	(IN)	ユーザー・コンテキスト。
ch	(IN)	文字データに対するポインタ。
len	(IN)	文字データの長さ。

notationDecl()

説明

表記法宣言の通知を受け取ります。

構文

```
sword (*notationDecl)( void *ctx,
                       const oratext *name,
                       const oratext *publicId,
                       const oratext *systemId);
```

パラメータ	IN/OUT	説明
ctx	(IN)	ユーザー・コンテキスト。
name	(IN)	表記法名。
publicID	(IN)	表記法の公開識別子（存在しない場合は null）。
systemId	(IN)	表記法のシステム識別子。

processingInstruction()

説明

処理命令の通知を受け取ります。

構文

```
sword (*processingInstruction)( void *ctx,
                                const oratext *target,
                                const oratext *data);
```

パラメータ	IN/OUT	説明
ctx	(IN)	ユーザー・コンテキスト。
target	(IN)	処理命令のターゲット。
data	(IN)	処理命令のデータ（データが存在しない場合は null）。

startDocument()

説明

ドキュメントの始まりの通知を受け取ります。

構文

```
sword (*startDocument)( void *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	ユーザー・コンテキスト。

startElement()

説明

要素の始まりの名前空間を認識しない通知を受け取ります。

構文

```
sword (*startElement)( void *ctx,
                       const oratext *name,
                       const struct xmlnodes *attrs);
```

パラメータ	IN/OUT	説明
ctx	(IN)	ユーザー・コンテキスト。
name	(IN)	要素名。
attrs	(IN)	指定された属性またはデフォルトの属性。

unparsedEntityDecl()

説明

解析対象外のエンティティ宣言の通知を受け取ります。

構文

```
sword (*unparsedEntityDecl)( void *ctx,
                             const oratext *name,
                             const oratext *publicId,
                             const oratext *systemId,
                             const oratext *notationName);
```

パラメータ	IN/OUT	説明
ctx	(IN)	ユーザー・コンテキスト。
name	(IN)	エンティティ名。
publicId	(IN)	エンティティの公開識別子（存在しない場合は null）。
systemId	(IN)	エンティティのシステム識別子。
notationName	(IN)	対応する表記法名。

attributeDecl()

説明

要素の属性宣言に関する通知を受け取ります。

構文

```
sword (*attributeDecl)( void *ctx,
                        const oratext *elem,
                        const oratext *attr,
                        const oratext *body);
```

パラメータ	IN/OUT	説明
ctx	(IN)	ユーザー・コンテキスト。
elem	(IN)	要素名。
attr	(IN)	属性名。
body	(IN)	属性宣言の本体。

comment()

説明

コメントに関する通知を受け取ります。

構文

```
sword (*comment)( void *ctx,
                  const oratext *data);
```

パラメータ	IN/OUT	説明
ctx	(IN)	クライアント・コンテキスト。
data	(IN)	コメントの本体。

elementDecl()

説明

要素宣言に関する通知を受け取ります。

構文

```
sword (*elementDecl)( void *ctx,
                      const oratext *name,
                      const oratext *content);
```

パラメータ	IN/OUT	説明
ctx	(IN)	ユーザー・コンテキスト。
name	(IN)	要素名。
content	(IN)	要素のコンテンツ・モデル。

nsStartElement()

説明

要素の名前空間の始まりの通知を受け取ります。

構文

```
sword (*nsStartElement)( void *ctx,
                         const oratext *qname,
                         const oratext *local,
                         const oratext *namespace,
                         const struct xmlnodes *attrs);
```

パラメータ	IN/OUT	説明
ctx	(IN)	クライアント・コンテキスト。
qname	(IN)	要素の完全修飾名。
local	(IN)	要素のローカル名。
namespace	(IN)	要素の名前空間（URI）。
attrs	(IN)	指定された属性またはデフォルトの属性。

W3C の DOM API

W3C の DOM API の説明

DOM 標準はオブジェクト指向であるため、C に適用するには、次の変更が必要です。

- 再利用されるファンクション名の拡張が必要です。たとえば、属性クラスの `getValue()` には一意の名前 `getAttrValue()` を指定し、`getNodeValue()` によって確立されるパターンと一致させます。
- DOM を拡張するために、いくつかのファンクションを追加する必要があります。たとえば、ノードの子ノード数を戻すように定義されたファンクションは存在しないため、`numChildNodes()` が新しく作成されます。

この C DOM インタフェースの実装は、REC-DOM-Level-1-19981001 に準拠します。

W3C の DOM API のデータ構造

表 13-6 W3C の DOM API のデータ構造

データ構造	宣言	説明
boolean	typedef int boolean;	ブール値（`true` または `false`）。
oratext	typedef unsigned char oratext;	文字列ポインタ。
xmlctx	typedef struct xmlctx xmlctx;	最上位の XML パーサー・コンテキスト。 **注意 :** xmlctx のコンテンツは内部的に使用されるため、ユーザーからはアクセスできません。
xmlnode	typedef struct xmlnode xmlnode;	ドキュメント・ノード。 **注意 :** xmlnode のコンテンツは内部的に使用されるため、ユーザーからはアクセスできません。
xmlnodes	typedef struct xmlnodes xmlnodes;	ノードの配列。 **注意 :** xmlnodes のコンテンツは内部的に使用されるため、ユーザーからはアクセスできません。

表 13-6 W3C の DOM API のデータ構造（続き）

データ構造	宣言	説明
xmlntype	ELEMENT_NODE = 1 /* element */ ATTRIBUTE_NODE = 2 /* attribute */ TEXT_NODE = 3 /* char data not escaped by CDATA */ CDATA_SECTION_NODE = 4 /* char data escaped by CDATA */ ENTITY_REFERENCE_NODE = 5 /* entity reference */ ENTITY_NODE = 6 /* entity */ PROCESSING_INSTRUCTION_ NODE = 7 /* processing instruction */ COMMENT_NODE= 8 /* comment */ DOCUMENT_NODE = /* document */ DOCUMENT_TYPE_NODE = 10 /* DTD */ DOCUMENT_FRAGMENT_NODE = 11 /* document fragment */ NOTATION_NODE = 12 /* notation */	ノードのタイプの列挙。 **注意：**名前および値は DOM 仕様に適合します。 解析ツリーのノードのタイプ。`getNodeType()` を参照してください。

W3C の DOM API のファンクション

表 13-7 W3C の DOM API のファンクションの概要

ファンクション	説明
appendChild()（13-46 ページ）	親ノードの子ノード・リストに子ノードを追加します。
appendData()（13-46 ページ）	ノードの現行データに文字データを追加します。
cloneNode()（13-47 ページ）	指定されたノードと同一の新しいノードを作成します。
createAttribute()（13-48 ページ）	要素ノードに名前空間を認識しない属性を作成します。
createAttributeNS()（13-48 ページ）	要素ノードに名前空間を認識する属性を作成します。
createCDATASection()（13-49 ページ）	CDATA セクション・ノードを作成します。
createComment()（13-49 ページ）	コメント・ノードを作成します。
createDocument()（13-50 ページ）	名前空間を認識しないドキュメント・ノードを作成します。
createDocumentFragment()（13-50 ページ）	ドキュメント・フラグメント・ノードを作成します。
createDocumentNS()（13-51 ページ）	名前空間を認識するドキュメント・ノードを作成します。
createDocumentType()（13-51 ページ）	新しい DTD ノードを作成します。
createElement()（13-52 ページ）	名前空間を認識しない要素ノードを作成します。
createElementNS()（13-52 ページ）	名前空間を認識する要素ノードを作成します。
createEntityReference()（13-53 ページ）	実体参照ノードを作成します。
createProcessingInstruction()（13-53 ページ）	処理命令ノードを作成します。
createTextNode()（13-54 ページ）	Text ノードを作成します。
deleteData()（13-54 ページ）	ノードの文字データから部分文字列を削除します。
getAttribute()（13-55 ページ）	配列から、インデックスを指定して、1 つの属性を戻します。
getAttributeIndex()（13-55 ページ）	インデックスを指定して、要素の属性を戻します。
getAttrName()（13-56 ページ）	属性名を戻します。

表 13-7 W3C の DOM API のファンクションの概要（続き）

ファンクション	説明
getAttributeNode()（13-56 ページ）	名前を指定して、要素の属性ノードを戻します。
getAttributes()（13-57 ページ）	要素の属性の配列を戻します。
getAttrSpecified()（13-57 ページ）	属性の指定されたフラグの値を戻します。
getAttrValue()（13-58 ページ）	属性の値を戻します。
getCharData()（13-58 ページ）	Text ノードの文字データを戻します。
getCharLength()（13-58 ページ）	Text ノードの文字データの長さを戻します。
getChildNode()（13-59 ページ）	子のインデックスを指定して、ノードの配列からノードを戻します。
getChildNodes()（13-59 ページ）	ノードの子ノードの配列を戻します。
getContentModel()（13-60 ページ）	DTD から要素のコンテンツ・モデルを戻します。
getDocOrder()（13-60 ページ）	ノードのドキュメント順序のカーディナルを戻します。
getDocType()（13-61 ページ）	現行の DTD を戻します。
getDocTypeEntities()（13-61 ページ）	DTD の汎用エンティティのリストを戻します。
getDocTypeName()（13-61 ページ）	DTD 名を戻します。
getDocTypeNotations()（13-62 ページ）	DTD の表記法のリストを戻します。
getDocument()（13-62 ページ）	最上位のドキュメント・ノードを戻します。
getDocumentElement()（13-62 ページ）	最上位またはルートの要素ノードを戻します。
getElementByID()（13-63 ページ）	指定された ID を持つ要素ノードを戻します。
getElementsByTagName()（13-63 ページ）	一致する名前を持つ要素のリストを戻します。
getElementsByTagNameNS()（13-64 ページ）	一致する名前を持つ要素のリストを、名前空間の情報とともに戻します。
getEntityNotation()（13-64 ページ）	エンティティの NDATA を戻します。
getEntityPubID()（13-65 ページ）	エンティティの公開識別子を戻します。

表 13-7 W3C の DOM API のファンクションの概要（続き）

ファンクション	説明
getEntitySysID()（13-65 ページ）	エンティティのシステム識別子を戻します。
getFirstChild()（13-65 ページ）	ノードの最初の子ノードを戻します。
getImplementation()（13-66 ページ）	DOM インプリメンテーション構造を戻します。
getLastChild()（13-66 ページ）	ノードの最後の子ノードを戻します。
getNamedItem()（13-67 ページ）	NodeList から名前付きノードを戻します。
getNextSibling()（13-67 ページ）	ノードの直後の兄弟関係を戻します。
getNodeMapLength()（13-68 ページ）	NodeMap 内のエントリ数を戻します（DOM の getLength）。
getNodeName()（13-68 ページ）	ノード名を戻します。
getNodeType()（13-69 ページ）	ノードの型コード（列挙）を戻します。
getNodeValue()（13-69 ページ）	ノードの値（文字データ）を戻します。
getNotationPubID()（13-70 ページ）	表記法の公開識別子を戻します（DOM の getPublicId）。
getNotationSysID()（13-70 ページ）	表記法のシステム識別子を戻します（DOM の getSystemId）。
getOwnerDocument()（13-71 ページ）	指定されたノードを含むドキュメント・ノードを戻します。
getParentNode()（13-71 ページ）	ノードの親ノードを戻します。
getPIData()（13-72 ページ）	処理命令のデータを戻します（DOM の getData）。
getPITarget()（13-72 ページ）	処理命令のターゲットを戻します（DOM の getTarget）。
getPreviousSibling()（13-72 ページ）	ノードの直前の兄弟関係を戻します。
getTagName()（13-73 ページ）	ノードのタグ名を戻します。現在、これは名前と同じです。
hasAttributes()（13-73 ページ）	要素ノードに属性が含まれているかどうかを判別します（DOM の拡張機能）。
hasChildNodes()（13-74 ページ）	ノードに子ノードが含まれているかどうかを判別します。
hasFeature()（13-74 ページ）	DOM インプリメンテーションが固有の機能をサポートするかどうかを判別します。
importNode()（13-75 ページ）	ドキュメントに、別のドキュメントからノードをコピーします。

表 13-7 W3C の DOM API のファンクションの概要（続き）

ファンクション	説明
insertBefore()（13-75 ページ）	指定された参照ノードの前に新しい子ノードを挿入します。
insertData()（13-76 ページ）	ノードの既存のデータに新しい文字データを挿入します。
isStandalone()（13-76 ページ）	ドキュメントが単独かどうかを判別します（DOM の拡張機能）。
nodeValid()（13-77 ページ）	現行の DTD に対してノードを検証します（DOM の拡張機能）。
normalize()（13-77 ページ）	隣接する Text ノードをマージしてノードを正規化します。
numAttributes()（13-78 ページ）	要素ノードの属性数を戻します（DOM の拡張機能）。
numChildNodes()（13-78 ページ）	ノードの子ノードの数を戻します（DOM の拡張機能）。
prefixToURI()（13-78 ページ）	名前空間の接頭辞およびノードを指定して、一致する URL を戻します。
removeAttribute()（13-79 ページ）	名前を指定して、要素の属性を削除します。
removeAttributeNode()（13-79 ページ）	ポインタを指定して、要素の属性を削除します。
removeChild()（13-80 ページ）	ノードをその親ノードの子ノード・リストから削除します。
removeNamedItem()（13-80 ページ）	名前を指定して、ノードを NodeList から削除します。
replaceChild()（13-81 ページ）	ノードを他のノードで置換します。
replaceData()（13-81 ページ）	ノードの文字データの部分文字列を他の文字列で置換します。
saveString()（13-82 ページ）	XML メモリー・プールに文字列を保存します。
saveString2()（13-82 ページ）	
setAttribute()（13-83 ページ）	属性名および属性値を指定して、新しい属性を要素ノードに対して設定（追加または置換）します。
setAttributeNode()（13-83 ページ）	新しい属性に対するポインタを指定して、その新しい属性を要素ノードに対して設定（追加または置換）します。
setAttrValue()（13-84 ページ）	属性の値を設定します。
setCharData()（13-84 ページ）	Text ノードまたは CDATA ノードの値を設定します。

表 13-7 W3C の DOM API のファンクションの概要（続き）

ファンクション	説明
setNamedItem()（13-85 ページ）	親ノードの子ノード・リストに新しいノードを設定（追加または置換）します。
setNodeValue()（13-85 ページ）	ノードの値（文字データ）を設定します。
setPIData()（13-86 ページ）	処理命令のデータを設定します（DOM の setData）。
splitText()（13-86 ページ）	ノードの文字データを 2 分割します。
substringData()（13-87 ページ）	ノードの文字データの部分文字列を戻します。

appendChild()

説明

指定された親ノードの子ノード・リストの最後に新しいノードを追加し、追加したノードを戻します。

構文

```
xmlnode *appendChild( xmlctx *ctx,
                      xmlnode *parent,
                      xmlnode *newnode);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
parent	(IN)	親ノード。
newnode	(IN)	追加する新しいノード。

appendData()

説明

指定された文字列を Text ノードまたは CDATA ノードの文字データに追加します。

構文

```
void appendData( xmlctx *ctx,
                 xmlnode *node,
                 const oratext *arg);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
node	(IN)	ノードに対するポインタ。
arg	(IN)	追加する新しいデータ。

cloneNode()

説明

ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。複製ノードは親を持ちません（parentNode() は null を戻します）。

要素を複製すると、デフォルトの属性を表すために XML プロセッサによって生成された属性を含む、すべての属性およびその値がコピーされます。ただし、このメソッドでは、ノードに含まれるテキストは、子の Text ノードに含まれているため、ディープ・クローンでないかぎりコピーされません。他のタイプのノードを複製すると、単純にそのノードのコピーが戻されます。

ディープ・クローンでは、ノードの子ノードは単に指定されるのみでなく、再帰的に複製されます。

構文

```
xmlnode *cloneNode( xmlctx *ctx,
                    const xmlnode *old,
                    boolean deep);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
old	(IN)	複製する古いノード。
deep	(IN)	再帰フラグ。

createAttribute()

説明

指定された名前および値を持つ新しい属性ノードを作成します。この新しいノードは連結されていないため、setAttributeNode() を使用して要素ノードに追加する必要があります。

構文

```
xmlnode *createAttribute( xmlctx *ctx,
                          const oratext *name,
                          const oratext *value);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
name	(IN)	新しい属性の名前。
value	(IN)	新しい属性の値。

createAttributeNS()

説明

名前空間の情報とともに、指定された名前および値を持つ新しい属性ノードを作成します。この新しいノードは連結されていないため、setAttributeNode() を使用して要素ノードに追加する必要があります。

構文

```
xmlnode *createAttributeNS( xmlctx *ctx,
                            const oratext *uri,
                            const oratext *qname,
                            const oratext *value);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
uri	(IN)	新しい属性の名前空間 URI。
qname	(IN)	新しい属性の修飾名。
value	(IN)	新しい属性の値。

createCDATASection()

説明

新しい CDATA ノードを作成します。

構文

```
xmlnode *createCDATASection( xmlctx *ctx,
                             const oratext *data);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
data	(IN)	CDATA 本体。

createComment()

説明

新しいコメント・ノードを作成します。

構文

```
xmlnode *createComment( xmlctx *ctx,
                        const oratext *data);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
data	(IN)	コメントのテキスト。

createDocument()

説明

メモリーに新しいドキュメントを作成します。元のファンクション createDocument() は、DOM 2.0 CORE で標準化されています。互換性のために、以前のファンクションで元の使用方法を保持し、新しい CORE ファンクションを createDocumentNS() と呼びます。XML 文書は、常に DOCUMENT_NODE 型のノードがルートになります。このファンクションによって、ルート・ノードが作成され、コンテキスト内に設定されます。存在できる現行のドキュメントは 1 つのみであるため、ドキュメント・ノードも 1 つのみです。ノードがすでに存在する場合、このファンクションは処理を行わず、null を戻します。

構文

```
xmlnode* createDocument( xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。

createDocumentFragment()

説明

新しいドキュメント・フラグメント・ノードを作成します。ドキュメント・フラグメントは、軽量なドキュメント・オブジェクトです。これには 1 つ以上の子ノードが含まれますが、ドキュメント全体のオーバーヘッドはありません。ドキュメント・フラグメントは、一部の操作（挿入など）で単純なノードのかわりに使用できます。この場合、操作はフラグメント・ノード自体ではなく、フラグメントのすべての子ノードに対して行われます。

構文

```
xmlnode *createDocumentFragment( xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。

createDocumentNS()

説明

メモリーに新しいドキュメントを作成します。元のファンクション createDocument() は、DOM 2.0 CORE で標準化されています。互換性のために、以前のファンクションで元の使用方法を保持し、新しい CORE ファンクションを createDocumentNS と呼びます。メモリーに新しいドキュメントを作成します。XML 文書は、常に DOCUMENT_NODE 型のノードがルートになります。このファンクションによって、ルート・ノードが作成され、コンテキスト内に設定されます。存在できる現行のドキュメントは 1 つのみであるため、ドキュメント・ノードも 1 つのみです。ノードがすでに存在する場合、このファンクションは処理を行わず、null を戻します。DTD を指定すると、その ownerDocument() 属性が、作成されるドキュメントに設定されます。

構文

```
xmlnode* createDocumentNS( xmldomimp *imp,
                           oratext *uri,
                           oratext *qname,
                           xmlnode *dtd);
```

パラメータ	IN/OUT	説明
imp	(IN)	XML DOM インプリメンテーション。 getImplementation() を参照してください。
uri	(IN)	新しいドキュメントの名前空間 URI。
qname	(IN)	新しいドキュメントの名前空間修飾名（DOCUMENT_NODE の名前）。
dtd	(IN)	ドキュメントに対応する DTD。

createDocumentType()

説明

新しいドキュメント・タイプ（DTD）のノードを作成します。

構文

```
xmlnode* createDocumentType( xmldomimp *imp,
                             oratext *qname,
                             oratext *pubid,
                             oratext *sysid);
```

パラメータ	IN/OUT	説明
imp	(IN)	XML DOM インプリメンテーション。 getImplementation() を参照してください。
pubid	(IN)	外部サブセットの公開識別子。
qname	(IN)	新しいドキュメント・タイプの名前空間修飾名 (DOCUMENT_NODE の名前)。
sysid	(IN)	外部サブセットのシステム識別子。

createElement()

説明

新しい要素ノードを作成します。

構文

```
xmlnode *createElement( xmlctx *ctx,
                        const oratext *elname);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
elname	(IN)	新しい要素の名前。

createElementNS()

説明

名前空間の情報を含む要素ノードを作成します。

構文

```
xmlnode *createElementNS( xmlctx *ctx,
                          const oratext *uri,
                          const oratext *qname);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
uri	(IN)	新しい要素の名前空間 URI。
qname	(IN)	新しい要素の修飾名。

createEntityReference()

説明

新しい実体参照ノードを作成します。

構文

```
xmlnode *createEntityReference( xmlctx *ctx,
                                const oratext *name);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
name	(IN)	参照するエンティティの名前。

createProcessingInstruction()

説明

指定されたターゲットおよびコンテンツを持つ新しい処理命令ノードを作成します。

構文

```
xmlnode *createProcessingInstruction( xmlctx *ctx,
                                      const oratext *target,
                                      const oratext *data);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
target	(IN)	処理命令のターゲット。
data	(IN)	処理命令の定義。

createTextNode()

説明

指定されたコンテンツを持つ新しい Text ノードを作成します。

構文

```
xmlnode *createTextNode( xmlctx *ctx,
                         const oratext *data);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
data	(IN)	ノードのデータ。

deleteData()

説明

ノードの文字データから部分文字列を削除します。

構文

```
void deleteData( xmlctx *ctx,
                 xmlnode *node,
                 ub4 offset,
                 ub4 count);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
node	(IN)	ノードに対するポインタ。
offset	(IN)	部分文字列の開始オフセット（0（ゼロ）から開始）。
count	(IN)	部分文字列の長さ。

getAttribute()

説明

インデックス（0（ゼロ）から開始）を指定して、属性の配列から 1 つの属性を戻します。属性の名前または値（あるいはその両方）は、getAttrName() および getAttrValue() を使用してフェッチします。エラーが発生すると、null を戻します。

構文

```
const oratext *getAttribute( const xmlnode *node,
                             const oratext *name);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。
name	(IN)	属性名。

getAttributeIndex()

説明

インデックス（0（ゼロ）から開始）を指定して、属性の配列から 1 つの属性を戻します。属性の名前または値（あるいはその両方）は、getAttrName() および getAttrValue() を使用してフェッチします。エラーが発生すると、null を戻します。

構文

```
xmlnode *getAttributeIndex( const xmlnodes *attrs,
                            size_t index);
```

パラメータ	IN/OUT	説明
attrs	(IN)	属性ノードの構造に対するポインタ（getAttribute() によって戻される）。
index	(IN)	属性番号（0（ゼロ）から開始）。

getAttrName()

説明

属性に対するポインタを指定して、その属性の名前を戻します。DOM 仕様では、これは getName() という名前のメソッドです。

構文

```
const oratext *getAttrName( const xmlnode *attr);
```

パラメータ	IN/OUT	説明
attr	(IN)	属性に対するポインタ（getAttribute() を参照）。

getAttributeNode()

説明

指定された名前の要素ノードの属性に対するポインタを戻します。該当する属性が存在しない場合は、null を戻します。

構文

```
xmlnode *getAttributeNode( const xmlnode *elem,
                           const oratext *name);
```

パラメータ	IN/OUT	説明
elem	(IN)	要素ノードに対するポインタ。
name	(IN)	属性名。

getAttributes()

説明

指定されたノードのすべての属性の配列を戻します。その後、このポインタを getAttribute に渡して個々の属性ポインタをフェッチするか、または numAttributes に渡して属性の合計数を戻すことができます。属性を定義していない場合は、null が戻されます。

構文

```
xmlnodes *getAttributes( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	属性を戻すノード。

getAttrSpecified()

説明

属性の specified フラグを戻します。元のドキュメントで、または DOM を介してこの属性に値が明示的に指定されている場合は true を、指定されていない場合は false を戻します。ノードが属性でない場合は、false を戻します。DOM 仕様では、これは getSpecified() という名前のメソッドです。

構文

```
boolean getAttrSpecified( const xmlnode *attr);
```

パラメータ	IN/OUT	説明
attr	(IN)	属性に対するポインタ（getAttribute() を参照）。

getAttrValue()

説明

属性に対するポインタを指定して、その属性の値（定義）を戻します。DOM 仕様では、これは getValue() という名前のメソッドです。

構文

```
const oratext *getAttrValue( const xmlnode *attr);
```

パラメータ	IN/OUT	説明
attr	(IN)	属性に対するポインタ（getAttribute() を参照）。

getCharData()

説明

Text ノードまたは CDATA ノードの文字データを戻します。DOM 仕様では、これは getData() という名前のメソッドです。

構文

```
const oratext *getCharData( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	Text ノードに対するポインタ。

getCharLength()

説明

Text ノードまたは CDATA ノードの文字データの長さを戻します。DOM 仕様では、これは getLength() という名前のメソッドです。

構文

```
size_t getCharLength( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	Text ノードに対するポインタ。

getChildNode()

説明

ノードの配列内にある *n* 番目のノードを戻します。該当するノードが存在しない場合は、null を戻します。これは、DOM 仕様にはない、新しく作成されたファンクションですが、DOM のパターンと一致する名前が付けられています。

構文

```
xmlnode* getChildNode( const xmlnodes *nodes,
                       size_t index);
```

パラメータ	IN/OUT	説明
nodes	(IN)	ノードの配列（getChildNodes() を参照）。
index	(IN)	子ノード番号（0（ゼロ）から開始）。

getChildNodes()

説明

指定されたノードの子ノードの配列を戻します。その後、このポインタを getChildNode() に渡して個々の子ノードをフェッチできます。

構文

```
xmlnodes* getChildNodes( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	子ノードを戻すノード。

getContentModel()

説明

現行の DTD から、指定された要素のコンテンツ・モデルを戻します。コンテンツ・モデルは XML ノードで構成されるため、解析対象ドキュメントと同じファンクションを使用して検索できます。

構文

```
xmlnode *getContentModel( xmlnode *dtd,
                          oratext *name);
```

パラメータ	IN/OUT	説明
dtd	(IN)	DTD に対するポインタ。
name	(IN)	要素名。

getDocOrder()

説明

ノードのドキュメント順序のカーディナルを戻します。`setDocOrder()` がコールされていない場合は、すべてのノードの順序が 0（ゼロ）になります。このファンクションは、主に XSLT プロセッサによって使用されます。

構文

```
ub4 getDocOrder( xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ドキュメント順序を戻すノード。

getDocType()

説明

現行ドキュメントの（不透明）DTD に対するポインタを戻します。

構文

```
xmlnode* getDocType( xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサーのコンテンツ。

getDocTypeEntities()

説明

指定された DTD に対して定義されている（汎用）エンティティの配列を戻します。

構文

```
xmlnodes *getDocTypeEntities( xmlnode* dtd);
```

パラメータ	IN/OUT	説明
dtd	(IN)	DTD に対するポインタ。

getDocTypeName()

説明

指定された DTD の名前を戻します。

構文

```
const oratext *getDocTypeName( xmlnode* dtd);
```

パラメータ	IN/OUT	説明
dtd	(IN)	DTD に対するポインタ。

getDocTypeNotations()

説明

指定された DTD に対して定義されている表記法の配列を戻します。

構文

```
xmlnodes *getDocTypeNotations( xmlnode* dtd);
```

パラメータ	IN/OUT	説明
dtd	(IN)	DTD に対するポインタ。

getDocument()

説明

解析対象ドキュメントのルート・ノードを戻します。このルート・ノードは、常に DOCUMENT_NODE 型です。ドキュメント・ノードの子であるルート要素ノードを戻す getDocumentElement() ファンクションと比較してください。

構文

```
xmlnode* getDocument( xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。

getDocumentElement()

説明

解析対象ドキュメントのルート要素（ノード）を戻します。このノードがドキュメント全体のルートになります。最上位のドキュメント・ノード（ルート要素ノードの親ノード）を戻す getDocument と比較してください。

構文

```
xmlnode* getDocumentElement( xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。

getElementByID()

説明

指定された ID を持つ要素ノードを戻します。ID が定義されていない（または他の問題が発生した）場合は、null を戻します。

構文

```
xmlnode *getElementByID( xmlctx *ctx,
                         oratext *id);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。
id	(IN)	要素 ID。

getElementsByTagName()

説明

指定されたタグ名を持つ（指定されたノードをルートとするツリー内の）すべての要素のリストを、ツリーの先行順走査で検出した順に戻します。ルートが null の場合は、ドキュメント全体を検索します。特殊値「*」を指定すると、すべてのタグに一致します。

構文

```
xmlnodes *getElementsByTagName( xmlctx *ctx,
                                xmlnode *root,
                                const oratext *name);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。
root	(IN)	ツリーのルート・ノード。
name	(IN)	要素のタグ名。

getElementsByTagNameNS()

説明

指定されたタグ名を持つ（指定されたノードをルートとするツリー内の）すべての要素のリストを、ツリーの先行順走査で検出した順に戻します。ルートが null の場合は、ドキュメント全体を検索します。特殊値「*」を指定すると、すべてのタグに一致します。

構文

```
xmlnodes *getElementsByTagNameNS( xmlctx *ctx,
                                  xmlnode *root,
                                  const oratext *uri,
                                  const oratext *local);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML パーサー・コンテキスト。
root	(IN)	ツリーのルート・ノード。
uri	(IN)	要素の名前空間 URI。
local	(IN)	要素のローカル名。

getEntityNotation()

説明

エンティティ・ノードの NDATA（表記法）を戻します。DOM 仕様では、これは getNotationName() という名前のメソッドです。

構文

```
const oratext *getEntityNotation( const xmlnode *ent);
```

パラメータ	IN/OUT	説明
ent	(IN)	エンティティに対するポインタ。

getEntityPubID()

説明

エンティティ・ノードの公開識別子を戻します。DOM 仕様では、これは getPublicId() という名前のメソッドです。

構文

```
const oratext *getEntityPubID( const xmlnode *ent);
```

パラメータ	IN/OUT	説明
ent	(IN)	エンティティに対するポインタ。

getEntitySysID()

説明

エンティティ・ノードのシステム識別子を戻します。DOM 仕様では、これは getSystemId() という名前のメソッドです。

構文

```
const oratext *getEntitySysID( const xmlnode *ent);
```

パラメータ	IN/OUT	説明
ent	(IN)	エンティティに対するポインタ。

getFirstChild()

説明

指定されたノードの最初の子ノードを戻します。ノードに子ノードが存在しない場合は、null を戻します。

構文

```
xmlnode *getFirstChild( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

getImplementation()

説明

実装の DOM インプリメンテーション構造に対するポインタを戻します。該当する情報がない場合は、null を戻します。

構文

```
xmldomimp* getImplementation( xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。

getLastChild()

説明

指定されたノードの最後の子ノードを戻します。ノードに子ノードが存在しない場合は、null を戻します。

構文

```
xmlnode *getLastChild( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

getNamedItem()

説明

ノードの配列から、指定されたノードを戻します。また、ユーザーのインデックス（提供されている場合）をそのノードの子ノード番号（0（ゼロ）から開始）に設定します。

構文

```
xmlnode *getNamedItem( const xmlnodes *nodes,
                       const oratext *name,
                       size_t *index);
```

パラメータ	IN/OUT	説明
nodes	(IN)	ノードの配列。
name	(IN)	フェッチするノードの名前。
index	(OUT)	検出されたノードのインデックス。

getNextSibling()

説明

指定されたノードの直後の兄弟関係（親ノードの次の子ノード）に対するポインタを戻します。ノードが最後の子ノードの場合は、null を戻します。

構文

```
xmlnode *getNextSibling( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

getNodeMapLength()

説明

ノードの配列（getChildNodes() によって戻される）を指定して、マップ内のノードの数を戻します。DOM 仕様では、これは getLength という名前のメンバー・ファンクションです。

構文

```
size_t getNodeMapLength( const xmlnodes *nodes);
```

パラメータ	IN/OUT	説明
nodes	(IN)	ノードの配列。

getNodeName()

説明

指定されたノードの名前を戻します。ノードに名前が付いていない場合は、null を戻します。現在は、「タグ名」と「名前」は同義であることに注意してください。

構文

```
const oratext *getNodeName( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

getNodeType()

説明

ノードの型コードを戻します。

構文

```
xmlntype getNodeType( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

getNodeValue()

説明

ノードの値（対応する文字データ）を戻します。ノードにデータが存在しない場合は、null を戻します。

構文

```
const oratext* getNodeValue( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

getNotationPubID()

説明

表記法ノードの公開識別子を戻します。DOM 仕様では、これは getPublicId() という名前のメソッドです。

構文

```
const oratext *getNotationPubID( const xmlnode *note);
```

パラメータ	IN/OUT	説明
note	(IN)	ノードに対するポインタ。

getNotationSysID()

説明

表記法ノードのシステム識別子を戻します。DOM 仕様では、これは getSystemId() という名前のメソッドです。

構文

```
const oratext *getNotationSysID( const xmlnode *note);
```

パラメータ	IN/OUT	説明
note	(IN)	ノードに対するポインタ。

getOwnerDocument()

説明

指定されたノードを含むドキュメント・ノードを戻します。常に、DOCUMENT_NODE 型のノードが XML 文書のルートになります。ドキュメント内の任意のノードに対して getOwnerDocument() をコールすると、そのドキュメント・ノードが戻されます。

構文

```
xmlnode *getOwnerDocument( xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

getParentNode()

説明

指定されたノードの親ノードを戻します。ノードが最上位のノードの場合は、null を戻します。

構文

```
xmlnode *getParentNode( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

getPIData()

説明

処理命令のデータ文字列を戻します。DOM 仕様では、これは getData() という名前のメソッドです。

構文

```
const oratext *getPIData( const xmlnode *pi);
```

パラメータ	IN/OUT	説明
pi	(IN)	処理命令ノードに対するポインタ。

getPITarget()

説明

処理命令のターゲット文字列を戻します。DOM 仕様では、これは getTarget() という名前のメソッドです。

構文

```
const oratext *getPITarget( const xmlnode *pi);
```

パラメータ	IN/OUT	説明
pi	(IN)	処理命令ノードに対するポインタ。

getPreviousSibling()

説明

指定されたノードの直前の兄弟関係（このノードの前に存在する同レベルのノード）を戻します。ノードが最初の子ノードの場合は、null を戻します。

構文

```
xmlnode *getPreviousSibling( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

getTagName()

説明

ノードのタグ名を戻します。現在、これはノードの名前と同義です。「getNodeName()」を参照してください。W3C のワーキング・グループでは、DOM 仕様について、「Node インタフェースには汎用の nodeName 属性があるにもかかわらず、Element インタフェースには tagName 属性がある。これらの 2 つの属性には同じ値が含まれている必要があるが、ワーキング・グループでは、DOM API が様々なユーザー層のニーズを満たす必要があることを考慮し、両方をサポートする価値がある」と考えています。

構文

```
const oratext *getTagName( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

hasAttributes()

説明

指定されたノードに定義済の属性が含まれているかどうかを判別し、含まれている場合は `true` を、含まれていない場合は `false` を戻します。これは、`hasChildNodes()` から開始されたパターンに従って名前が付けられた DOM 拡張機能です。

構文

```
boolean hasAttributes( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

hasChildNodes()

説明

指定されたノードに子ノードが含まれているかどうかを判別し、含まれている場合は true を、含まれていない場合は false を戻します。getChildNodes() がポインタ（子ノードが含まれている場合）または null（子ノードが含まれていない場合）のどちらを戻すかをテストすると、これと同じ結果を得ることができます。

構文

```
boolean hasChildNodes( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。

hasFeature()

説明

DOM インプリメンテーションが固有の機能およびバージョンを実装するかどうかをテストします。feature は、テストする機能のパッケージ名です。DOM レベル 1 の場合、正当な値は「HTML」および「XML」です（大 / 小文字は区別されません）。version は、テストするパッケージ名のバージョン番号を示します。DOM レベル 1 の場合は文字列「1.0」です。バージョンを指定していない場合は、その機能のすべてのバージョンがサポートされ、true が戻されます。

構文

```
boolean hasFeature( xmldomimp *imp,
                    const oratext *feature,
                    const oratext *version);
```

パラメータ	IN/OUT	説明
imp	(IN)	XML DOM インプリメンテーション構造。
feature	(IN)	テストする機能のパッケージ名。
version	(IN)	テストするパッケージ名のバージョン番号。

importNode()

説明

ドキュメントに、別のドキュメントからノードをインポートします。戻されたノードが親ノードを持たない場合は、null になります。ソース・ノードが変更されていない場合、または元のドキュメントから削除されていない場合は、ソース・ノードの新しいコピーを作成します。deep を true に指定した場合は、ノードの下位サブツリーを再帰的にインポートします。false に指定した場合は、ノードのみをインポートします。

追加情報が nodeType に応じてコピーされます。このとき、XML ソースのフラグメントが 1 つのドキュメントから別のドキュメントにコピーされた場合に行われる動作のミラー化が試行され、2 つのドキュメントが異なる DTD を持つ場合があることが認識されます。ノードのタイプごとに実行される固有のアクションについては、DOM 2.0 仕様を参照してください。

構文

```
xmlnode *importNode( xmlctx *ctx,
                     xmlnode *import,
                     boolean deep);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
import	(IN)	インポートするノード。
deep	(IN)	サブツリーを再帰的にインポートするかどうかを判別します。

insertBefore()

説明

指定された親ノードの子ノード・リスト内にある既存の参照ノードの前に新しいノードを挿入します。参照ノードが null の場合は、リストの最後に新しいノードを追加します。新しいノードがドキュメント・フラグメントの場合は、フラグメント自体ではなく、その子ノードが同じ順序で挿入されます。これは、新しいノードがツリー内にすでに存在する場合は、最初に削除されます。

構文

```
xmlnode *insertBefore( xmlctx *ctx,
                       xmlnode *parent,
                       xmlnode *newChild,
                       xmlnode *refChild);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
parent	(IN)	新しいノードの挿入先の親ノード。
newChild	(IN)	挿入する新しい子ノード。
refChild	(IN)	前に新しいノードが挿入される参照ノード。

insertData()

説明

指定されたオフセットでノードの文字データに文字列を挿入します。

構文

```
void insertData( xmlctx *ctx,
                 xmlnode *node,
                 ub4 offset,
                 const oratext *arg);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
node	(IN)	ノードに対するポインタ。
offset	(IN)	挿入点（0（ゼロ）から開始）。
arg	(IN)	挿入する新しい文字列。

isStandalone()

説明

ドキュメントの <?xml?> 処理命令で指定されたとおり、スタンドアロン・フラグの値を戻します。これは、DOM 仕様にはない、新しく作成されたファンクションですが、DOM のパターンと一致する名前が付けられています。

構文

```
boolean isStandalone( xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。

nodeValid()

説明

ノードを DTD に対して検証します。検証結果が正常な場合は 0（ゼロ）を、正常でない場合は 0（ゼロ）以外のエラー・コード（メッセージ・ファイルで検索可能）を戻します。このファンクションは、API または Class Generator（あるいはその両方）を使用して独自のドキュメントを構成するアプリケーション用に提供されています。通常、パーサーがドキュメントを検証するため、ユーザーは nodeValid を明示的にコールする必要はありません。

構文

```
uword nodeValid( xmlctx *ctx,
                 xmlnode *node);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
node	(IN)	ノードに対するポインタ。

normalize()

説明

隣接する Text ノードをマージして、要素を正規化します。隣接する Text ノードは通常の解析中は存在せず、DOM を介して追加のノードが挿入された場合にのみ存在します。

構文

```
void normalize( xmlctx *ctx,
                xmlnode *elem);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
elem	(IN)	要素ノードに対するポインタ。

numAttributes()

説明

属性の配列（getAttributes によって戻される）に含まれる定義済の属性の数を戻します。これは、DOM 仕様にはない、新しく作成されたファンクションですが、DOM のパターンに従って名前が付けられています。

構文

```
size_t numAttributes( const xmlnodes *attrs);
```

パラメータ	IN/OUT	説明
attrs	(IN)	属性の配列。

numChildNodes()

説明

ノードの配列（getChildNodes によって戻される）に含まれる子ノードの数を戻します。これは、DOM 仕様にはない、新しく作成されたファンクションですが、DOM のパターンに従って名前が付けられています。

構文

```
size_t numChildNodes(const xmlnodes *nodes);
```

パラメータ	IN/OUT	説明
nodes	(IN)	不透明型ノード構造に対するポインタ。

prefixToURI()

説明

指定された名前空間の接頭辞およびノードと一致する URL を戻します。指定されたノードが一致する接頭辞を持たない場合は、その親ノードが試行されます。次に、その親ノードの親ノードが試行され、ルート・ノードまで繰り返し試行されます。接頭辞が定義されていない場合は、null を戻します。

構文

```
oratext *prefixToURI( xmlnode *node,
                      oratext *prefix);
```

パラメータ	IN/OUT	説明
node	(IN)	開始ノード。
prefix	(IN)	一致させる接頭辞。

removeAttribute()

説明

指定された属性名の属性を要素ノードから削除します。削除された属性がデフォルト値を持つ場合は、すぐに置換されます。

構文

```
void removeAttribute( xmlnode *elem,
                      const oratext *name);
```

パラメータ	IN/OUT	説明
elem	(IN)	要素ノードに対するポインタ。
name	(IN)	削除する属性の名前。

removeAttributeNode()

説明

属性に対するポインタを指定して、その属性を要素から削除します。属性を正常に削除すると、その属性ノードを戻します。エラーが発生すると、null を戻します。

構文

```
xmlnode *removeAttributeNode( xmlnode *elem,
                              xmlnode *attr);
```

パラメータ	IN/OUT	説明
elem	(IN)	要素ノードに対するポインタ。
attr	(IN)	削除する属性ノード。

removeChild()

説明

指定されたノードをその親ノードから削除します。

構文

```
xmlnode *removeChild( xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	削除する古いノード。

removeNamedItem()

説明

指定されたノードをノードの配列から削除します。

構文

```
xmlnode *removeNamedItem( xmlnodes *nodes,
                          const oratext *name);
```

パラメータ	IN/OUT	説明
nodes	(IN)	ノードのリスト。
name	(IN)	削除するノードの名前。

replaceChild()

説明

既存の子ノードを新しいノードで置換し、古いノードを戻します。新しいノードがツリー内にすでに存在する場合は、最初に削除されます。

構文

```
xmlnode *replaceChild( xmlctx *ctx,
                       xmlnode *oldnode,
                       xmlnode *newnode);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
oldnode	(IN)	置換される古いノード。
newnode	(IN)	新しく置換するノード。

replaceData()

説明

指定された文字オフセットから始まる、指定された長さの部分文字列を置換文字列と置き換えます。

構文

```
void replaceData( xmlctx *ctx,
                  xmlnode *node,
                  ub4 offset,
                  ub4 count,
                  const oratext *arg);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
node	(IN)	ノードに対するポインタ。
offset	(IN)	置換する部分文字列の開始オフセット（0（ゼロ）から開始）。
count	(IN)	古い部分文字列の長さ。

パラメータ	IN/OUT	説明
arg	(IN)	置換テキスト。

saveString()

説明

XML メモリー・プールに文字列を保存します。メモリーは、xmlclean または xmlterm のコール後に解放されます。

構文

```
oratext *saveString( xmlctx *ctx,
                     oratext *str);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
str	(IN)	保存する文字列。

saveString2()

説明

メモリーを割り当て、null で終了する Unicode 文字列を XML 文字列プールに保存します。Unicode 文字列は、1 つではなく 2 つの null バイトで終了することに注意してください。この方法で文字列を保存すると、文字列は、最もコンパクトな方法で 1 つのプールに一列に並んで格納されるため、個々に解放することはできません。xmlclean() または xmlterm() をコールしてプール全体を解放した場合にのみ、メモリーが再利用されます。シングルバイトまたはマルチバイトの文字列を保存するには、saveString() を使用します。

構文

```
ub2 *saveString2( xmlctx *ctx,
                  ub2 *ustr);
```

パラメータ	IN/OUT	説明
ctx	(IN)	LPX コンテキスト。
ustr	(IN)	Unicode 文字列に対するポインタ。

setAttribute()

説明

要素ノードの新しい属性を作成します。指定された属性名を持つ属性がすでに存在する場合は、単純にその値が置換されます。

構文

```
xmlnode *setAttribute( xmlctx *ctx,
                       xmlnode *elem,
                       const oratext *name,
                       const oratext *value);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
elem	(IN)	要素ノードに対するポインタ。
name	(IN)	新しい属性の名前。
value	(IN)	新しい属性の値。

setAttributeNode()

説明

指定された要素に新しい属性を追加します。指定された属性名を持つ属性がすでに存在する場合は、その属性が置換され、ユーザーの古いポインタ（提供されている場合）が古い属性に設定されます。新しい属性である場合、その属性が追加され、古いポインタが null に設定されます。正常に実行されたことを示す真理値を戻します。

構文

```
boolean setAttributeNode( xmlctx *ctx,
                          xmlnode *elem,
                          xmlnode *newNode,
                          xmlnode **oldNode);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
elem	(IN)	要素ノードに対するポインタ。

パラメータ	IN/OUT	説明
newNode	(IN)	新しい属性に対するポインタ。
oldNode	(OUT)	古い属性の戻りポインタ。

setAttrValue()

説明

属性値を設定します。

構文

```
void setAttrValue( xmlnode *attr,
                   const oratext *data);
```

パラメータ	IN/OUT	説明
attr	(IN)	値を設定する必要がある属性。
data	(IN)	属性の新しい値。

setCharData()

説明

Text ノードまたは CDATA ノードの値を設定します。

構文

```
void setCharData( xmlnode *node,
                  const oratext *data);
```

パラメータ	IN/OUT	説明
node	(IN)	データを設定する必要があるノード。
data	(IN)	ノードの新しいテキスト。

setNamedItem()

説明

親ノードのマップに新しい子ノードを設定します。同じ名前を持つ古いノードが存在する場合は、その古いノードを置換（およびユーザー・ポインタが提供されている場合は、それを古いノードに設定）します。該当する名前のノードが存在しない場合は、ノードをマップに追加し、ポインタを null に設定します。

構文

```
boolean setNamedItem( xmlctx *ctx,
                      xmlnode *nodes,
                      xmlnode *node,
                      xmlnode **old);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
nodes	(IN)	ノード・マップ。
node	(IN)	追加する新しいノード。
old	(OUT)	置換されるノードに対するポインタ。

setNodeValue()

説明

ノードに対応する値（文字データ）を設定します。

構文

```
void setNodeValue( xmlnode *node,
                   const oratext *data);
```

パラメータ	IN/OUT	説明
node	(IN)	ノードに対するポインタ。
data	(IN)	ノードの新しいデータ。

setPIData()

説明

処理命令データを設定します（setNodeValue と同じ）。データを null に設定することはできません。DOM 仕様では、これは setData() という名前のメソッドです。

構文

```
void setPIData( xmlnode *pi,
                const oratext *data);
```

パラメータ	IN/OUT	説明
pi	(IN)	処理命令ノードに対するポインタ。
data	(IN)	処理命令の新しいデータ。

splitText()

説明

Text ノードを、指定されたオフセットで 2 つの Text ノードに分割し、その両方を兄弟としてツリー内に保持します。元のノードには、オフセット点までのすべての内容が含まれます。また、元のノードの直後に兄弟関係として挿入される新しいノードには、オフセット点以降のすべての古い内容が含まれます。

構文

```
xmlnode *splitText( xmlctx *ctx,
                    xmlnode *old,
                    uword offset);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
old	(IN)	分割する元のノード。
offset	(IN)	分割点のオフセット。

substringData()

説明

ノードの文字データの部分文字列を戻します。

構文

```
const oratext *substringData( xmlctx *ctx,
                              const xmlnode *node,
                              ub4 offset,
                              ub4 count);
```

パラメータ	IN/OUT	説明
ctx	(IN)	XML コンテキスト。
node	(IN)	ノードに対するポインタ。
offset	(IN)	部分文字列の開始オフセット。
count	(IN)	部分文字列の長さ。

名前空間 API

名前空間 API の説明

名前空間 API は、DOM への拡張機能であるインタフェースを提供し、ドキュメントの名前空間に関連する情報を提供します。

- XML 名前空間は、eXtensible Markup Language 文書で使用される要素名および属性名を、URI 参照によって識別される名前空間と対応付けることによって修飾するための単純な方法を提供します。1 つの XML 文書には、複数のソフトウェア・モジュールに対して定義され、それらのモジュールによって使用される要素および属性（ここではマークアップ・ボキャブラリという）が含まれている場合があります。この目的の 1 つはモジュール性です。一般的なマークアップ・ボキャブラリが存在し、それに対して使用できる有効なソフトウェアがある場合は、マークアップを新しく作成するより再利用する方が効率的です。
- このような複数のマークアップ・ボキャブラリを含むドキュメントによって、認識問題および競合問題が発生します。ソフトウェア・モジュールは、他のソフトウェア・パッケージのマークアップが同じ要素型または属性名を使用するために競合が発生した場合でも、設計上処理する必要があるタグおよび属性を認識できる必要があります。
- これらの考慮点から、ドキュメントの構造体は汎用名を持ち、その汎用名は構造体を含むドキュメント以外でも有効である必要があります。これを実行するメカニズムが、この XML 名前空間の C 実装によって提供されます。
- XML 名前空間の名前は、修飾名として表示される必要があります。修飾名には、名前を名前空間接頭辞とローカル部分に分割するコロンが 1 つ含まれています。接頭辞は、URI 参照にマップされ、名前空間を選択します。汎用的に管理される URI 名前空間とドキュメント独自の名前空間を組み合せると、汎用的に一意の識別子が作成されます。接頭辞の有効範囲を判別したり、デフォルトを設定するためのメカニズムが提供されています。
- URI 参照は、名前では許可されていない文字を含む場合があるため、名前空間接頭辞として直接使用することはできません。そのため、名前空間接頭辞は URI 参照のプロキシとして機能します。W3C の名前空間仕様で記述されている属性に基づく構文が、名前空間接頭辞と URI 参照の対応付けを宣言するために使用されます。
- この名前空間インタフェースの C 実装は、改訂 REC-xml-names-19990114 の XML 名前空間標準に準拠します。

名前空間 API のファンクション

表 13-8 名前空間 API のファンクションの概要

ファンクション	説明
getAttrLocal()（13-89 ページ）	属性のローカル名を戻します。
getAttrNamespace()（13-90 ページ）	属性の名前空間（URI）を戻します。
getAttrPrefix()（13-90 ページ）	属性の接頭辞を戻します。
getAttrQualifiedName()（13-90 ページ）	属性の完全修飾名を戻します。
getNodeLocal()（13-91 ページ）	ノードのローカル名を戻します。
getNodeNamespace()（13-91 ページ）	ノードの名前空間（URI）を戻します。
getNodePrefix()（13-91 ページ）	ノードの接頭辞を戻します。
getNodeQualifiedName()（13-92 ページ）	ノードの修飾名を戻します。

getAttrLocal()

説明

属性のローカル名を戻します。

構文

```
const oratext *getAttrLocal( const xmlnode *attr);
```

パラメータ	IN/OUT	説明
attr	(IN)	不透明属性構造に対するポインタ（getAttribute() を参照）。

getAttrNamespace()

説明

属性の名前空間を戻します。

構文

```
const oratext *getAttrNamespace( const xmlnode *attr);
```

パラメータ	IN/OUT	説明
attr	(IN)	不透明属性構造に対するポインタ（getAttribute() を参照）。

getAttrPrefix()

説明

属性の接頭辞を戻します。

構文

```
const oratext *getAttrPrefix( const xmlnode *attr);
```

パラメータ	IN/OUT	説明
attr	(IN)	不透明属性構造に対するポインタ（getAttribute() を参照）。

getAttrQualifiedName()

説明

属性の完全修飾名を戻します。

構文

```
const oratext *getAttrQualifiedName( const xmlnode *attr);
```

パラメータ	IN/OUT	説明
attr	(IN)	不透明属性構造に対するポインタ（getAttribute() を参照）。

getNodeLocal()

説明

ノードのローカル名を戻します。

構文

```
const oratext *getNodeLocal( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	ローカル名の取得元ノード。

getNodeNamespace()

説明

ノードの名前空間を戻します。

構文

```
const oratext *getNodeNamespace( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	名前空間の取得元ノード。

getNodePrefix()

説明

ノードの接頭辞を戻します。

構文

```
const oratext *getNodePrefix( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	接頭辞の取得元ノード。

getNodeQualifiedName()

説明

ノードの完全修飾名を戻します。

構文

```
const oratext *getNodeQualifiedName( const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	(IN)	名前の取得元ノード。

14

XSLT Processor for C

XSL Language Processors は、XML 文書を読み込み、異なるスタイルの他の XML 文書に変換するために使用されます。XSLT プロセッサの C 実装は、XSL Transformations 標準（1999 年 11 月 16 日公開のバージョン 1.0）に準拠しており、XSLT 仕様で指定されている、XSLT プロセッサに必要な動作を実装します。

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』

XSLT API のデータ構造および型

データ構造

表 14-1 データ型の概要

データ型	定義	説明
oratext	typedef unsigned char oratext;	すべてのデータ・エンコーディングに使用される文字列ポインタ。必要に応じてキャストします（UTF-16 の場合、(ub *)）。
ub4	typedef unsigned int ub4;	32 ビット以上の符号なし整数。
uword	typedef unsigned int uword;	システム固有の符号なし整数。
xmlctx	typedef struct xmlctx xmlctx;	最上位の XML コンテキスト。
xmlnode	typedef struct xmlnode xmlnode;	ドキュメント・ノード。
xmlsaxcb	struct xmlsaxcb { sword (*startDocument)(void *ctx); sword (*endDocument)(void *ctx); sword (*startElement)(void *ctx, const oratext *name, const struct xmlattrs *attrs); sword (*endElement)(void *ctx, const oratext *name); sword (*characters)(void *ctx, const oratext *ch, size_t len); sword (*ignorableWhitespace) (void *ctx, const oratext *ch, size_t len); sword (*processingInstruction) (void *ctx, const oratext *target, const oratext *data);	SAX コールバック構造（SAX のみ）。

表 14-1 データ型の概要（続き）

データ型	定義	説明
	sword (*notationDecl)(void *ctx, const oratext *name, const oratext *publicId, const oratext *systemId); sword (*unparsedEntityDecl) (void *ctx, const oratext *name, const oratext *publicId, const oratext *systemId, const oratext *notationName); sword (*nsStartElement)(void *ctx, const oratext *qname, const oratext *local, const oratext *namespace, const struct xmlattrs *attrs); sword (*comment)(void *ctx, const oratext *data); sword (*elementDecl)(void *ctx, const oratext *name, const oratext *content); sword (*attributeDecl)(void *ctx, const oratext *elem, const oratext *attr, const oratext *body); /* The following 5 fields are reserved for future use. */ void (*empty1)(); void (*empty2)(); void (*empty3)(); void (*empty4)(); void (*empty5)(); }; typedef struct xmlsaxcb xmlsaxcb;	

表 14-1 データ型の概要（続き）

データ型	定義	説明

xmlstream	struct xmlstream;
	typedef struct xmlstream xmlstream;
	
	#define XMLSTREAM_OPENF(func) uword
	func(xmlstream *stream, ub4 rsvd)
	#define XMLSTREAM_WRITEF(func) uword
	func(xmlstream *stream, ub4 rsvd)
	#define XMLSTREAM_CLOSEF(func) uword
	func(xmlstream *stream, ub4 rsvd)
	
	struct xmlstream {
	XMLSTREAM_OPENF((*open));
	XMLSTREAM_WRITEF((*write));
	XMLSTREAM_CLOSEF((*close));
	
	/* User can associate any data to lpxstream structure
	using this member so that he can access that inside
	the callbacks which will always have xmlstream as
	the firstargument. */
	
	void *userdata;
	};
		ストリームベースの出力。

xpnset	typedef struct xpnset xpnset;	XPath ノード・セット。
xpnsetele	typedef struct xpnsetele xpnsetele;	XPath ノード・セット要素。
xpobj	typedef struct xpobj xpobj;	XPath オブジェクト。

表 14-1 データ型の概要（続き）

データ型	定義	説明
xpobjtyp	typedef enum xpobjtyp (XPOBJTYP_BOOL, /* boolean */ XPOBJTYP_NUM, /* number */ XPOBJTYP_STR, /* string */ XPOBJTYP_NSET, /* node set */ XPOBJTYP_RTFRAG /* restult tree fragment */ } xpobjtyp;	XPath オブジェクト型。
xslctx	typedef struct xslctx xslctx;	最上位の XSL コンテキスト。
xsloutputmethod	typedef enum { XSLMETHOD_UNKNOWN = 0, XSLMETHOD_TEXT, XSLMETHOD_HTML, } xsloutputmethod;	XSL 処理の出力メソッド。

XSLT API のファンクション

表 14-2 XSLT ファンクションの概要

ファンクション	説明
xslprocess()（14-7 ページ）	XML 文書ソースで XSL スタイルシートを処理します。
xslprocessex()（14-8 ページ）	XML 文書ソースで XSL スタイルシートを処理します。
xslprocessxml()（14-9 ページ）	XML 文書ソースで XSL スタイルシートを処理します。
xslprocessdocfrag()（14-10 ページ）	XML 文書ソースで XSL スタイルシートを処理します。
xslinit()（14-11 ページ）	XSL コンテキストを作成および初期化します。
xslgetbaseuri()（14-11 ページ）	XSL コンテキストに対応するベース URI を取得します。
xslgetoutputdomctx()（14-12 ページ）	XSL コンテキストに対応する出力 DOM を取得します。
xslgetoutputsax()（14-12 ページ）	XSL コンテキストに対応する SAX オブジェクトを取得します。

表 14-2 XSLT ファンクションの概要（続き）

ファンクション	説明
xslgetoutputstream()（14-13 ページ）	XSL コンテキストに対応する出力ストリームを取得します。
xslgetresultdocfrag()（14-13 ページ）	XSL コンテキストに対応する結果のドキュメント・フラグメントを取得します。
xslgetxslctx()（14-13 ページ）	XSL コンテキストに対応する XML スタイルシートの XML コンテキストを取得します。
xslsettextparam()（14-14 ページ）	最上位のパラメータ（変数）を設定します。
xslgettextparam()（14-14 ページ）	最上位のパラメータ（変数）の値を戻します。
xslsetoutputdomctx()（14-15 ページ）	XML コンテキストを設定して、処理結果を（DOM として）格納します。
xslsetoutputencoding()（14-15 ページ）	XSLT 出力エンコーディングを設定します。
xslsetoutputmethod()（14-16 ページ）	XSLT 出力メソッドを設定します。
xslsetoutputsax()（14-17 ページ）	出力用に生成される SAX ベースのイベントを設定します。
xslsetoutputsaxctx()（14-17 ページ）	出力用に生成される SAX ベースのイベントを設定します。
xslsetoutputstream()（14-18 ページ）	テキスト・ストリームベースの出力を生成するためのコールバックを指定するストリーム・オブジェクトを設定します。
xslresetallparams()（14-18 ページ）	追加されたすべての最上位のパラメータをリセットします。
xslterm()（14-19 ページ）	XSLT コンテキスト・オブジェクトを終了します。

表 14-3 XPath ファンクションの概要

ファンクション	説明
xpmakexpathctx()（14-19 ページ）	XPath コンテキストを作成します。
xpfreexpathctx()（14-20 ページ）	XPath 式を解析します。
xpparsexpathexpr()（14-20 ページ）	XPath 式を評価します。
xpevalxpathexpr()（14-21 ページ）	XPath 式の評価結果の値を取得します。
xpgetxpobjtyp()（14-21 ページ）	XPath 式の評価結果の値を取得します。
xpgetbooleanval()（14-22 ページ）	XPath 式の評価結果の値を取得します。

表 14-3 XPath ファンクションの概要（続き）

ファンクション	説明
xpgetnumval()（14-22 ページ）	XPath 式の評価結果の値を取得します。
xpgetstrval()（14-22 ページ）	XPath 式の評価結果の値を取得します。
xpgetnsetval()（14-23 ページ）	XPath 式の評価結果の値を取得します。
xpgetrtfragval()（14-23 ページ）	XPath 式の評価結果の値を取得します。
xpgetfirstnsetelem()（14-23 ページ）	XPath 式の評価結果の値を取得します。
xpgetnextnsetelem()（14-24 ページ）	XPath 式の評価結果の値を取得します。
xpgetnsetelemnode()（14-24 ページ）	XPath 式の評価結果の値を取得します。

xslprocess()

説明

XML 文書ソースで XSL スタイルシートを処理します。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。

構文

```
uword xslprocess( xmlctx *docctx,
                  xmlctx *xslctx,
                  xmlctx *resctx,
                  xmlnode **result);
```

引数	IN/OUT	説明
docctx	(IN/OUT)	XML 文書コンテキスト。
xslctx	(IN)	XSL スタイルシート・コンテキスト。
resctx	(IN)	結果のドキュメント・フラグメント・コンテキスト。
result	(IN/OUT)	結果のドキュメント・フラグメント・ノード。

xslprocessex()

説明

XML 文書ソースで XSL スタイルシートを処理します。解析を開始する前に、設定する最上位のパラメータのリストを指定できます。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。

構文

```
uword xslprocessex( xmlctx *docctx,
                    xmlctx *sctx,
                    xmlctx *resctx,
                    size_t nparams,
                    oratext *params[],
                    oratext *vals[],
                    oratext *baseuri,
                    xmlnode **result);
```

引数	IN/OUT	説明
docctx	(IN/OUT)	XML 文書コンテキスト。
sctx	(IN)	XSL スタイルシート・コンテキスト。
resctx	(IN)	結果のドキュメント・フラグメント・コンテキスト。
nparams	(IN)	渡される名前と値の組の数。
params	(IN)	パラメータの名前の配列（数の場合は nparams である必要があります)。
vals	(IN)	パラメータの値の配列（数の場合は nparams である必要があります)。
baseuri	(IN)	スタイルシートのベース URI を指定する文字列。
result	(IN/OUT)	結果のドキュメント・フラグメント・ノード。

xslprocessxml()

説明

XML 文書ソースで XSL スタイルシートを処理します。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。

この XSLT API は、ユーザーが次の操作を行った後に使用されます。

- xslinit() を使用して、XSLT 処理コンテキストを作成した後。
- xslinit() を使用して、ベース URI を設定した後。
- xslsettextparam() を使用して、すべての最上位のパラメータを設定した後。
- xslsetoutput() を使用して、出力メソッドを選択した後。

このファンクションは、XSL のスタイルシート、パラメータおよび出力メソッドに従って、(docctx によって) 指定された XML ファイルを処理します。このファンクションをコールすると、選択した出力スキームによって、次のいずれかが戻されます。

- **ストリームベースの出力 :** ストリーム・コールバックを使用して、出力テキストがリダイレクトされます。
- **SAX ベースの出力 :** XSLT によって送信されたすべての SAX イベントが、ユーザーに戻されます。
- **DOM ベースの出力 :** xslgetresultdocgrag() ファンクションをコールすると、xslsetoutputdomctx() にコールして設定した resctx() によって作成された最終ドキュメント・フラグメントにアクセスできます。

構文

```
uword xslprocessxml( xslctx *xslSSctx,
                     xmlctx *docctx,
                     boolean normalize,
                     ub4 resvd);
```

引数	IN/OUT	説明
xslSSctx	(IN)	XSL 処理コンテキスト。
docctx	(IN/OUT)	XML 文書コンテキスト。
normalize	(IN)	true の場合は、入力 DOM ツリーを正規化します。
resvd		予約済。null に設定する必要があります。

xslprocessdocfrag()

説明

XML 文書ソースで XSL スタイルシートを処理します。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。

この XSLT API は、ユーザーが次の操作を行った後に使用されます。

- xslinit() を使用して、XSLT 処理コンテキストを作成した後。
- xslinit() を使用して、ベース URI を設定した後。
- xslsettextparam() を使用して、すべての最上位のパラメータを設定した後。
- xslsetoutput() を使用して、出力メソッドを選択した後。

このファンクションは、XSL のスタイルシート、パラメータおよび出力メソッドに従って、（docctx によって）指定された XML ファイルを処理します。このファンクションをコールすると、選択した出力スキームによって、次のいずれかが戻されます。

- **ストリームベースの出力 :** ストリーム・コールバックを使用して出力テキストがリダイレクトされます。
- **SAX ベースの出力 :** XSLT によって送信されたすべての SAX イベントが、ユーザーに戻されます。
- **DOM ベースの出力 :** xslgetresultdocgrag() ファンクションをコールすると、xslsetoutputdomctx() にコールして設定した resctx() によって作成された最終ドキュメント・フラグメントにアクセスできます。

構文

```
uword xslprocessxmldocfrag( xslctx *xslSSctx,
                            xmlctx *docctx,
                            xmlnode *docFrag,
                            boolean normalize,
                            ub4 resvd);
```

引数	IN/OUT	説明
xslSSctx	(IN)	XSL 処理コンテキスト。
docctx	(IN)	XML 文書コンテキスト。
docFrag	(IN)	入力ドキュメント・フラグメント・ノード。
normalize	(IN)	true の場合は、入力 DOM ツリーを正規化します。

引数	IN/OUT	説明
resvd		予約済。null に設定する必要があります。

xslinit()

説明

XSL コンテキストを作成および初期化します。正常に終了した場合は XSL コンテキスト・オブジェクトに対するポインタ、エラーが発生した場合は null を戻します。

構文

```
xslctx *xslinit ( uword *err,
                  xmlctx *xslctx,
                  const oratext *baseURI,
                  ub4 resvd);
```

引数	IN/OUT	説明
err	(OUT)	LPXERR_OK（正常に終了した場合）またはエラー・コード（エラーが発生した場合）。
xslctx	(IN)	入力スタイルシートの XSL コンテキスト。
baseURI	(IN)	include および import 文を処理するためのベース URI。
resvd		将来の使用を予約済。null に設定する必要があります。

xslgetbaseuri()

説明

XSL コンテキストに対応するベース URI を取得します。正常に終了した場合はベース URI に対するポインタ、エラーが発生した場合は null を戻します。

構文

```
const oratext* xslgetbaseuri ( xslctx *xslSSctx);
```

引数	IN/OUT	説明
xslSSctx	(IN)	問い合せる XSL 処理コンテキスト。

xslgetoutputdomctx()

説明

XSL コンテキストに対応する出力 DOM を取得します。正常に終了した場合は出力 DOM コンテキスト・オブジェクトに対するポインタ、エラーが発生した場合は null を戻します。

構文

```
xmlctx *xslgetoutputdomctx( xslctx *xslSSctx);
```

引数	IN/OUT	説明
xslSSctx	(IN)	問い合せる XSL 処理コンテキスト。

xslgetoutputsax()

説明

XSL コンテキストに対応する SAX オブジェクトを取得します。正常に終了した場合は出力 SAX オブジェクトに対するポインタ、エラーが発生した場合は null を戻します。

構文

```
xmlsaxcb *xslgetoutputsax( xslctx *xslSSctx);
```

引数	IN/OUT	説明
xslSSctx	(IN)	問い合せる XSL 処理コンテキスト。

xslgetoutputstream()

説明

XSL コンテキストに対応する出力ストリームを取得します。正常に終了した場合は出力ストリーム・オブジェクトに対するポインタ、エラーが発生した場合は null を戻します。

構文

```
xmlstream *xslgetoutputstream( xslctx *xslSSctx);
```

引数	IN/OUT	説明
xslSSctx	(IN)	問い合せる XSL 処理コンテキスト。

xslgetresultdocfrag()

説明

XSL コンテキストに対応する結果のドキュメント・フラグメントを取得します。正常に終了した場合はドキュメント・フラグメント・ノードに対するポインタ、エラーが発生した場合は null を戻します。

構文

```
xmlnode *xslgetresultdocfrag( xslctx *xslSSctx);
```

引数	IN/OUT	説明
xslSSctx	(IN)	問い合せる XSL 処理コンテキスト。

xslgetxslctx()

説明

XSL コンテキストに対応する XML スタイルシートの XML コンテキストを取得します。正常に終了した場合は XSL スタイルシートの XML コンテキスト・オブジェクトに対するポインタ、エラーが発生した場合は null を戻します。

構文

```
xmlctx *xslgetxslctx( xslctx *xslSSctx);
```

引数	IN/OUT	説明
xslSSctx	(IN)	問い合せる XSL 処理コンテキスト。

xslsettextparam()

説明

最上位のパラメータ（変数）を設定します。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。

構文

```
uword xslsettextparam ( xslctx *xslSSctx,
                        const oratext *param_name,
                        const oratext *param_value);
```

引数	IN/OUT	説明
xslSSctx	(IN)	問い合せる XSL 処理コンテキスト。
param_name	(IN)	値を設定するパラメータ（変数）の名前。
param_value	(IN)	パラメータの値。

xslgettextparam()

説明

最上位のパラメータ（変数）の値を戻します。正常に終了した場合はパラメータ（変数）の値に対するポインタ、エラーが発生した場合は null を戻します。

構文

```
const oratext *xslgettextparam( xslctx *xslSSctx,
                                const oratext *param_name);
```

引数	IN/OUT	説明
xslSSctx	(IN)	XSL 処理コンテキスト。
param_name	(IN)	値を取得するパラメータ（変数）の名前。

xslsetoutputdomctx()

説明

XML コンテキストを設定して、処理結果を（DOM として）格納します。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。

xslsetoutputdomctx() をコールすると、以前 xslsetoutputdomctx()、xslsetoutputstream() または xslsetoutputsax() をコールして設定した出力メソッドをオーバーライドできます。XML 入力の後続の処理は、resctx によって指定された xmlctx にドキュメント・フラグメントの形式で格納されます。

xslgetoutputdomctx() を使用すると、現在設定されている resctx を取得できます。DOM コンテキストが設定されていないか、あるいは xslsetoutputdomctx() が xslsetoutputstream() または xslsetoutputsax() への後続のコールによってオーバーライドされている場合、xslgetoutputdomctx() は、現在の出力ノードが DOM 出力モードではないことを示す null コンテキストを戻します。

構文

```
uword xslsetoutputdomctx( xslctx *xslSSctx,
                          xmlctx *resctx);
```

引数	IN/OUT	説明
xslSSctx	(IN)	XSL 処理コンテキスト。
resctx	(IN)	出力ドキュメント・フラグメントが格納される XML コンテキスト・オブジェクト。このコンテキストを使用して、XML ツリーの後続の処理時に処理イベントが生成されるため、このコンテキストに対するポインタは最後まで有効な状態に保持される必要があります。

xslsetoutputencoding()

説明

XSLT 出力エンコーディングを設定します。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。このファンクションは、XSL スタイルシートの設定をオーバーライドします。

このファンクションのコールを使用して、XSL 処理ディレクティブ xsl:output encoding ="..." と同等の出力メソッドを設定します。XSL スタイルシートに 1 つ以上の類似した文が含まれている場合、このファンクションは、XSL スタイルシートに存在する古い文の効果をオーバーライドします。

構文

```
uword xslsetoutputencoding( xslctx *xslSSctx,
                            oratext *outcoding);
```

引数	IN/OUT	説明
xslSSctx	(IN)	XSL 処理コンテキスト。
outcoding	(IN)	出力エンコーディングのエンコーディング名。

xslsetoutputmethod()

説明

XSLT 出力メソッドを設定します。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。このファンクションは、XSL スタイルシートの設定をオーバーライドします。

このファンクションのコールを使用して、XSL 処理ディレクティブ xsl:output method = "text/xml/html" と同等の出力メソッドを設定します。XSL スタイルシートに 1 つ以上の類似した文が含まれている場合、このファンクションは、XSL スタイルシートに存在する古い文の効果をオーバーライドします。

構文

```
uword xslsetoutputmethod( xslctx *xslSSctx,
                          xsloutputmethod method,
                          ub4 resvd);
```

引数	IN/OUT	説明
xslSSctx	(IN)	XSL 処理コンテキスト。
method	(IN)	選択する出力メソッド。
resvd		将来の使用を予約済。null に設定する必要があります。

xslsetoutputsax()

説明

出力用に生成される SAX ベースのイベントを設定します。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。

xslsetoutputsazctx() をコールすると、SAX コールバックによって使用される SAX コンテンツを設定できます。設定しない場合は、SAX コンテンツとして SAX コールバックに null が渡されます。

> **注意：** SAX コールバックは、コールバックの設定のみで起動できます。

構文

```
uword xslsetoutputsax( xslctx *xslSSctx,
                       xmlsaxcb *s);
```

引数	IN/OUT	説明
xslSSctx	(IN)	XSL 処理コンテキスト。
s	(IN)	SAX イベントの生成に使用される SAX コールバック。

xslsetoutputsaxctx()

説明

出力用に生成される SAX ベースのイベントを設定します。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。

xslsetoutputsazctx() をコールすると、SAX コールバックによって使用される SAX コンテンツを設定できます。設定しない場合は、SAX コンテンツとして SAX コールバックに null が渡されます。

> **注意：** SAX コールバックは、コールバックの設定のみで起動できます。

構文

```
uword xslsetoutputsaxctx( xslctx *xslSSctx,
                          void *saxctx);
```

引数	IN/OUT	説明
xslSSctx	(IN)	XSL 処理コンテキスト。
saxctx	(IN)	SAX コンテキスト。

xslsetoutputstream()

説明

テキスト・ストリームベースの出力を生成するためのコールバックを指定するストリーム・オブジェクトを設定します。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。

構文

```
uword xslsetoutputstream( xslctx *xslSSctx,
                          xmlstream *stream);
```

引数	IN/OUT	説明
stream	(IN)	使用するコールバックを指定するストリーム構造。
xslSSctx	(IN)	XSL 処理コンテキスト。

xslresetallparams()

説明

追加されたすべての最上位のパラメータをリセットします。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。

構文

```
uword xslresetallparams( xslctx *xslSSctx);
```

引数	IN/OUT	説明
xslSSctx	(IN)	XSL 処理コンテキスト。

xslterm()

説明

XSLT コンテキスト・オブジェクトを終了します。正常に終了した場合は LPXERR_OK、エラーが発生した場合は LPXERR_FAIL を戻します。

構文

```
uword xslterm( xslctx *xslSSctx);
```

引数	IN/OUT	説明
xslSSctx	(IN)	XSL 処理コンテキスト。

xpmakexpathctx()

説明

XPath コンテキストを作成します。XPath コンテキスト・オブジェクトを戻します。このコールは常に正常に実行されます。

構文

```
xpctx *xpmakexpathctx( xmlctx *ctx,
                       xmlnode *xslnode,
                       xmlnode *xml_node,
                       oratext *baseURI,
                       size_t nctxels,
                       xmlnode **ctxnodes);
```

引数	IN/OUT	説明
ctx	(IN)	XSL コンテキスト。null に指定できます。
xslnode	(IN)	名前空間の拡張に使用される XSL ノード。null に指定できます。
xml_node	(IN)	コンテキスト・ノード。解析のために null に設定します。
baseURI	(IN)	解析のためのベース URI。
nctxels	(IN)	現行のノード・セット内のノードの数。
ctxnodes	(IN)	現行のノード・セット。

xpfreexpathctx()

説明

XPath コンテキストを解放します。

構文

```
void xpfreexpathctx( xpctx *xctx);
```

引数	IN/OUT	説明
xctx	(IN)	解放する XPath コンテキスト。

xpparsexpathexpr()

説明

XPath 式を解析します。正常に終了した場合は式ツリー、エラーが発生した場合は null を戻します。

構文

```
xpexpr *xpparsexpathexpr( xpctx *xctx,
                          oratext *expr,
                          sword *err);
```

引数	IN/OUT	説明
xctx	(IN/OUT)	XPath コンテキスト。
expr	(IN)	文字列形式の式。
err	(OUT)	エラー・コード。

xpevalxpathexpr()

説明

XPath 式を評価します。正常に終了した場合は結果オブジェクト、エラーが発生した場合は null を戻します。xpgetxpobjtyp() を使用して、次のいずれかの型の結果オブジェクトが取得されます。

- XPOBJTYP_BOOL
- XPOBJTYP_NUM
- XPOBJTYP_STR
- XPOBJTYP_NSET
- XPOBJTYP_RTFRAG

構文

```
xpobj *xpevalxpathexpr( xpctx *xctx,
                        xpexpr *exprtree,
                        sword *err);
```

引数	IN/OUT	説明
xctx	(IN/OUT)	XPath コンテキスト。
exprtree	(IN)	ツリー形式の式。
err	(OUT)	エラー・コード。

xpgetxpobjtyp()

説明

XPath 式の評価結果の値を取得します。正常に終了した場合は式の値、エラーが発生した場合は null を戻します。

構文

```
xpobjtyp xpgetxpobjtyp( xpobj *xobj);
```

引数	IN/OUT	説明
xobj	(IN)	式オブジェクト。

xpgetbooleanval()

説明

XPath 式の評価結果の値を取得します。正常に終了した場合は式の値、エラーが発生した場合は null を戻します。

構文

```
boolean xpgetbooleanval (xpobj *xobj);
```

引数	IN/OUT	説明
xobj	(IN)	式オブジェクト。

xpgetnumval()

説明

XPath 式の評価結果の値を取得します。正常に終了した場合は式の値、エラーが発生した場合は null を戻します。

構文

```
double xpgetnumval( xpobj *xobj);
```

引数	IN/OUT	説明
xobj	(IN)	式オブジェクト。

xpgetstrval()

説明

XPath 式の評価結果の値を取得します。正常に終了した場合は式の値、エラーが発生した場合は null を戻します。

構文

```
oratext* xpgetstrval( xpobj *xobj);
```

引数	IN/OUT	説明
xobj	(IN)	式オブジェクト。

xpgetnsetval()

説明

XPath 式の評価結果の値を取得します。正常に終了した場合は式の値、エラーが発生した場合は null を戻します。

構文

```
xpnset* xpgetnsetval( xpobj *xobj);
```

引数	IN/OUT	説明
xobj	(IN)	式オブジェクト。

xpgetrtfragval()

説明

XPath 式の評価結果の値を取得します。正常に終了した場合は式の値、エラーが発生した場合は null を戻します。

構文

```
xmlnode* xpgetrtfragval(xpobj *xobj);
```

引数	IN/OUT	説明
xobj	(IN)	式オブジェクト。

xpgetfirstnsetelem()

説明

XPath 式の評価結果の値を取得します。正常に終了した場合は式の値、エラーが発生した場合は null を戻します。

構文

```
xpnsetele* xpgetfirstnsetelem( xpnset *nset);
```

引数	IN/OUT	説明
nset	(IN)	ノード・セット。

xpgetnextnsetelem()

説明

XPath 式の評価結果の値を取得します。正常に終了した場合は式の値、エラーが発生した場合は null を戻します。

構文

```
xpnsetele* xpgetnextnsetelem( xpnsetele *nsetele);
```

引数	IN/OUT	説明
nsetele	(IN)	ノード・セット要素。

xpgetnsetelemnode()

説明

XPath 式の評価結果の値を取得します。正常に終了した場合は式の値、エラーが発生した場合は null を戻します。

構文

```
xmlnode *xpgetnsetelemnode( xpnsetele *nsetele);
```

引数	IN/OUT	説明
nsetele	(IN)	ノード・セット要素。

15

XML Schema Processor for C

XML Schema は、XML DTD のかわりに使用できます。XML Schema は、DTD と同様にドキュメント構成を定義しますが、要素の内容のデータの型指定を行い、追加機能も提供します。XML Schema は、スカラー、実数、日時、URI、エンコードされたバイナリ・データなどの多くの組込みデータ型を提供します。拡張または制限によって、これらの基本データ型からユーザー定義のデータ型を構築できます。

現在、XML Schema はベータのままです。このバージョンは、2001 年 5 月の W3C の勧告を実装します。現在、このベータ実装は不完全で、一部のスキーマ機能のみが実装されています。

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』

XML Schema Processor for C

XML Schema Processor for C のコール順序

XML Schema プロセッサは、schemaInitialize()、schemaValidate()、...schemaValidate()、schemaTerminate() の順でコールされます。セッションの開始時に initialize() がコールされ、このファンクションによってセッション中に使用されるスキーマ・コンテキストが戻されます。

検証プロセスは、有効か無効かを判別します。ドキュメントは、スキーマに対して有効か無効かのいずれかです。ドキュメントが有効である場合は、エラー・コード 0（ゼロ）が戻されます。ドキュメントが無効である場合は、0（ゼロ）以外のエラー・コードが戻され、問題が示されます。警告とエラーの区別はありません。すべての問題はエラーであり、致命的とみなされ、検証はすぐに停止します。

検出されたスキーマはロードされ、スキーマ・コンテキストに保持されます。セッション中、スキーマは 1 回のみロードされます。xmlclean() と同様のクリーンアップ・コールはありません。そのため、新しいドキュメントを検証する前にすべてのメモリーを解放し、状態をリセットする必要がある場合は、コンテキストを終了し、最初からやりなおす必要があります。

検証するインスタンス・ドキュメントが、まず XML パーサーによって解析されます。インスタンスに対する XML コンテキストが、オプションのスキーマの URL とともにスキーマ検証関数に渡されます。最適のスキーマを指定すると、ロードされてデフォルトのスキーマになります。同じスキーマ・コンテキストを使用して、複数のドキュメントを処理できます。セッションが終了すると、schemaTerminate() ファンクションがコールされます。これによって、ロードされたスキーマによって割り当てられたすべてのメモリーが解放されます。

XML Schema Processor for C のデータ型

表 15-1 XML Schema Processor for C のデータ型

データ型	定義	説明
xsd	typedef struct xsd xsd;	スキーマ構造。
xsdctx	typedef struct xsdctx xsdctx;	XML Schema プロセッサのコンテキスト。

注意：xsdctx および xsd のコンテンツは内部的（不透明）に使用されるため、ユーザーからはアクセスできません。

XML Schema Processor for C のファンクションおよびメソッド

表 15-2 XML Schema Processor for C のファンクションおよびメソッドの概要

ファンクションまたはメソッド	説明
schemaInitialize()（15-3 ページ）	XML Schema プロセッサを初期化します。
schemaLoad()（15-4 ページ）	XML Schema ファイルをオフラインでロードします。
schemaTarget()（15-5 ページ）	スキーマに対するターゲットの名前空間の URI を戻します。
schemaTerminate()（15-5 ページ）	XML Schema プロセッサを終了します。
schemaValidate()（15-5 ページ）	スキーマに対してインスタンス・ドキュメントを検証します。

schemaInitialize()

説明

XML Schema プロセッサを初期化します。このファンクションは、スキーマの検証を実行する前にコールする必要があります。同じコンテキストを繰り返し使用して、複数のドキュメントを検証できます。割り当てられたコンテキストに対するポインタを戻します。それ以外の場合は、`null` を戻します。

構文

```
xsdctx *schemaInitialize( xmlctx *ctx,
                          uword *err);
```

引数

引数	IN/OUT	説明
ctx	(IN)	スキーマ・コンテキストのメモリーの割当てに使用される XML パーサー・コンテキスト。
err	(OUT)	エラー・コード。

schemaLoad()

説明

スキーマをロードし、エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

「nsp」が指定されていないことを示すには、null を指定します。「nsp」がいずれのターゲットの名前空間にも存在しないことを示すには、空の文字列を指定します。それ以外の場合は、「http://www.example.com/Report」などの名前空間の URI を指定します。「nsp」を指定する場合は、次の条件を満たす必要があります。

- 名前空間が存在する場合、その実際の値が、インポートされたスキーマの targetNamespace（属性）の実際の値と同じである。
- 名前空間が存在しない場合、インポートされたスキーマに targetNamespace（属性）が含まれない。

構文

```
uword schemaLoad( xsdctx *scctx,
                  oratext *uri,
                  oratext *nsp,
                  xsd **schema);
```

引数	IN/OUT	説明
scctx	(IN)	スキーマ・コンテキストにメモリーを割り当てるために使用される XML コンテキスト。
uri	(IN)	コンパイラのキャラクタ・セットで表されたスキーマの URL。
nsp	(IN)	コンパイラのキャラクタ・セットで表されたスキーマの名前空間（オプション）。
schema	(OUT)	戻されたスキーマ構造。

schemaTarget()

説明

スキーマに対する対象名前空間の URI を戻します。

構文

```
oratext *schemaTarget( xsd *schema);
```

引数	IN/OUT	説明
schema	(IN)	スキーマ構造。

schemaTerminate()

説明

XML Schema プロセッサを終了し、割り当てられたすべてのメモリーを解放します。コンテキストは再利用できない場合があります。

構文

```
void schemaTerminate( xsdctx *scctx);
```

引数	IN/OUT	説明
scctx	(IN)	終了するスキーマ・コンテキスト。

schemaValidate()

説明

インスタンス・ドキュメントをスキーマに対して検証します。デフォルトのスキーマを指定すると、指定されたスキーマがロードされ、デフォルトになります。正常に検証された場合（ドキュメントが構造的に有効で、すべてのデータ型が確認されている場合）は 0（ゼロ）、正常に検証されなかった場合はエラー・コードを戻します。

構文

```
uword schemaValidate( xsdctx *scctx,
                      xmlnode *root,
                      oratext *schema);
```

引数	IN/OUT	説明
ctx	(IN)	スキーマ・コンテキスト。
root	(IN)	検証するインスタンス・ドキュメントのルート要素。
schema	(IN)	コンパイラのキャラクタ・セットで表されたデフォルト・スキーマの URI。

第III部

XDK for C++ パッケージ

第 III 部に含まれる章は、次のとおりです。

16

XML Parser for C++

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』

Attr クラス

Attr の説明

このクラスには、単一のドキュメント・ノード属性の名前および値にアクセスするためのメソッドが含まれます。

Attr のメソッド

表 16-1 Attr のメソッドの概要

メソッド	説明
getName()（16-2 ページ）	属性の名前を戻します。
getValue()（16-2 ページ）	属性の値（定義）を戻します。
getSpecified()（16-3 ページ）	属性の specified フラグの値を戻します。
setValue()（16-3 ページ）	属性の値を設定します。

getName()

説明

属性の名前を戻します。

構文

```
DOMString getName();
```

getValue()

説明

属性の値（定義）を戻します。

構文

```
DOMString getValue();
```

getSpecified()

説明

DOM から、属性の specified フラグの値を戻します。元のドキュメントでこの属性に値が明示的に指定されている場合は `true`、それ以外の場合は `false` を戻します。ユーザーではなく、実装がこの属性を操作することに注意してください。ユーザーが属性の値を変更した場合（結果的に、デフォルト値と同じ値になった場合でも）、specified フラグが自動的に `true` になります。属性を DTD のデフォルト値として再指定するには、その属性を削除する必要があります。削除後、実装によって、`false` に指定された specified およびデフォルト値（存在する場合）で新しい属性が使用可能になります。

構文

```
boolean getSpecified();
```

setValue()

説明

属性の値を設定します。

構文

```
void setValue(DOMString value);
```

パラメータ	説明
value	属性の新しい値。

CDATASection クラス

CDATASection の説明

このクラスは、Text のサブクラスである CData ノードのタイプを実装します。すべてのメソッドを Text クラスから継承します。

Comment クラス

Comment の説明

このクラスは、CharacterData のサブクラスである COMMENT ノードのタイプを実装します。すべてのメソッドを CharacterData クラスから継承します。

CharacterData クラス

CharacterData の説明

このクラスには、Text ノードに対応付けられたデータにアクセスし、変更するためのメソッドが含まれます。

CharacterData のメソッド

表 16-2 CharacterData のメソッドの概要

メソッド	説明
appendData()（16-6 ページ）	ノードのデータに文字列を追加します。
deleteData()（16-7 ページ）	Text ノードのデータの部分文字列を削除します。
getData()（16-7 ページ）	Text ノードの値を取得します。
getLength()（16-7 ページ）	Text ノードのデータの長さを戻します。
insertData()（16-8 ページ）	ノードのデータに文字列を挿入します。
replaceData()（16-8 ページ）	ノードのデータの部分文字列を置換します。
setData()（16-9 ページ）	Text ノードの値を設定します。
substringData()（16-9 ページ）	ノードのデータの部分文字列をフェッチします。

appendData()

説明

ノードのデータに文字列を追加します。

構文

```
void appendData(DOMString arg);
```

パラメータ	説明
arg	追加する文字列。

deleteData()

説明

Text ノードのデータの部分文字列を削除します。

構文

```
void deleteData(  unsigned long offset,
                  unsigned long count);
```

パラメータ	説明
count	削除する文字数。
offset	削除する部分文字列の開始オフセット（0（ゼロ）から開始）。

getData()

説明

Text ノードのデータ（値）を DOMString として取得します。

構文

```
DOMString getData();
```

getLength()

説明

Text ノードのデータの長さを戻します。

構文

```
size_t getLength();
```

insertData()

説明

ノードのデータに文字列を挿入します。

構文

```
void insertData(  unsigned long offset,
                  DOMString arg);
```

パラメータ	説明
arg	挿入する文字列。
offset	挿入点（0（ゼロ）から開始）。

replaceData()

説明

ノードのデータの部分文字列を置換します。

構文

```
void replaceData(  unsigned long offset,
                   unsigned long count,
                     DOMString arg);
```

パラメータ	説明
arg	挿入する文字列。
count	置換する文字数。
offset	挿入点（0（ゼロ）から開始）。

setData()

説明

Text ノードのデータ（値）を設定します。

構文

```
void setData(  DOMString data);
```

パラメータ	説明
data	ノードのデータ。

substringData()

説明

ノードのデータの部分文字列をフェッチします。

構文

```
DOMString substringData(  unsigned long offset,
                          unsigned long count);
```

パラメータ	説明
count	部分文字列の長さ。
offset	部分文字列の開始オフセット（0（ゼロ）から開始）。

Document クラス

Document の説明

このクラスには、ノードを作成および取り出すためのメソッドが含まれます。

Document のメソッド

表 16-3 Document のメソッドの概要

メソッド	説明
createAttribute()（16-11 ページ）	属性ノードを作成します。
createAttributeNS()（16-11 ページ）	名前空間の情報を含む属性ノードを作成します。
createCDATASection()（16-12 ページ）	CDATA ノードを作成します。
createComment()（16-12 ページ）	コメント・ノードを作成します。
createDocumentFragment()（16-12 ページ）	ドキュメント・フラグメント・ノードを作成します。
createElement()（16-13 ページ）	要素ノードを作成します。
createElementNS()（16-13 ページ）	名前空間の情報を含む要素ノードを作成します。
createEntityReference()（16-13 ページ）	実体参照ノードを作成します。
createProcessingInstruction()（16-14 ページ）	処理命令ノードを作成します。
createTextNode()（16-14 ページ）	Text ノードを作成します。
getElementByID()（16-14 ページ）	指定された ID を持つ要素ノードを戻します。
getElementsByTagName()（16-15 ページ）	タグ名に基づいてノードを選択します。
getImplementation()（16-16 ページ）	ドキュメントの DTD を戻します。
importNode()（16-16 ページ）	ドキュメントに、別のドキュメントからのノードをコピーします。
saveString()（16-17 ページ）	メモリーを割当て、指定された文字列を保存します。

createAttribute()

説明

新しい属性ノードを作成し、そのノードに対するポインタを戻します。setValue() を使用してこの値を設定します。

構文

```
Attr* createAttribute(  DOMString name,
                        DOMString value);
```

パラメータ	説明
name	属性名。
value	属性値。

createAttributeNS()

説明

名前空間の情報を含む新しい属性ノードを作成し、そのノードに対するポインタを戻します。setValue() を使用してこの値を設定します。

構文

```
Attr* createAttribute( DOMString nspuri,
                       DOMString qname,
                       DOMSring value);
```

パラメータ	説明
nspuri	要素の名前空間 URI。
qname	属性の修飾名。
value	属性値。

createCDATASection()

説明

指定されたコンテンツを含む新しい CDATA ノードを作成し、そのノードに対するポインタを戻します。

構文

```
CDATASection* createCDATASection(  DOMString data);
```

パラメータ	説明
data	ノードのコンテンツ。

createComment()

説明

指定されたコンテンツを含む新しいコメント・ノードを作成し、そのノードに対するポインタを戻します。

構文

```
Comment* createComment(  DOMString data);
```

パラメータ	説明
data	ノードのコンテンツ。

createDocumentFragment()

説明

新しいドキュメント・フラグメント・ノードを作成し、そのノードに対するポインタを戻します。

構文

```
DocumentFragment* createDocumentFragment(void);
```

createElement()

説明

指定されたタグ名を持つ新しい要素ノードを作成し、そのノードに対するポインタを戻します。

構文

```
Element* createElement(  DOMString tagName);
```

パラメータ	説明
tagName	要素のタグ名。

createElementNS()

説明

指定された（タグの）名前および名前空間の情報を持つ新しい要素ノードを作成し、そのノードに対するポインタを戻します。

構文

```
Element* createElementNS( DOMString nspuri,
                          DOMSring qname);
```

パラメータ	説明
nspuri	要素の名前空間 URI。
qname	要素の修飾名。

createEntityReference()

説明

新しい実体参照ノードを作成し、そのノードに対するポインタを戻します。

構文

```
EntityReference* createEntityReference(  DOMString name);
```

パラメータ	説明
name	参照するエンディティの名前。

createProcessingInstruction()

説明

新しい処理命令ノードを作成し、そのノードに対するポインタを戻します。

構文

```
ProcessingInstruction* createProcessingInstruction( DOMString target,
                                                     DOMString data);
```

パラメータ	説明
data	ノードのデータ。
target	処理命令のターゲット部分。

createTextNode()

説明

新しい Text ノードを作成し、そのノードに対するポインタを戻します。

構文

```
Text* createTextNode(  DOMString data);
```

パラメータ	説明
data	ノードのデータ。

getElementByID()

説明

指定された ID を持つ要素ノードを戻します。ID が定義されていない（または他の問題が発生した）場合は、null を戻します。

構文

```
Element* getElementByID(  DOMString name);
```

パラメータ	説明
name	要素 ID。

getElementsByTagName()

説明

指定されたタグ名を持つすべての要素の NodeList を、ドキュメント・ツリーの先行順走査で検出した順に戻します。elem が `null` の場合は、ドキュメント全体を検索します。特殊値「`*`」を指定すると、すべてのタグに一致します。一致するタグが検出されない場合は、`null` を戻します。次の表に、オプションを示します。

構文	説明
NodeList* getElementsByTagName(Element *elem, DOMString tagname);	タグ名で要素を一致させます。
NodeList* getElementsByTagNameNS(Element *elem, DOMString nspuri, DOMString local);	タグ名および名前空間情報で要素を一致させます。

パラメータ	説明
elem	ルート・ノード。
tagname	選択するタグ名。
nspuri	名前空間 URI。
local	名前空間のローカル名。

getImplementation()

説明

DOM インプリメンテーション構造を戻します。このファンクションは現在使用されておらず、将来の DOM インプリメンテーションのために予約されています。

構文

```
DOMImplementation* getImplementation();
```

importNode()

説明

ドキュメントに、別のドキュメントからノードをインポートし、そのノードに対するポインタを戻します。戻されたノードが親ノードを持たない場合、値は null になります。ソース・ノードが変更されていない場合、または元のドキュメントから削除されていない場合は、ソース・ノードの新しいコピーを作成します。deep を true に指定した場合は、ノードの下位サブツリーを再帰的にインポートします。false に指定した場合は、ノードのみをインポートします。

追加情報が nodeType に応じて適切にコピーされます。このとき、XML ソースのフラグメントが 1 つのドキュメントから別のドキュメントにコピーされた場合に行われる動作のミラー化が試行され、2 つのドキュメントが異なる DTD を持つ場合があることが認識されます。ノードのタイプごとに実行される固有のアクションについては、DOM 2.0 仕様を参照してください。

構文

```
Node *importNode( Node *importedNode,
                  boolean deep);
```

パラメータ	説明
deep	サブツリーを再帰的にインポートするかどうかを判別します（true または false）。
importedNode	インポートするノード。

saveString()

説明

メモリーを割り当て、指定された文字列を保存して、その文字列に対するポインタを戻します。このファンクションは、ローカルに生成された文字列の格納に使用されます。

構文

```
DOMString saveString(  DOMString str);
```

パラメータ	説明
str	保存する文字列。

DocumentType クラス

DocumentType の説明

このクラスには、ドキュメントの Document Type Definition（DTD）に関する情報にアクセスするためのメソッドが含まれます。

DocumentType のメソッド

表 16-4 DocumentType のメソッドの概要

メソッド	説明
getName()（16-18 ページ）	DTD の名前を戻します。
getEntities()（16-18 ページ）	DTD の（汎用）エンティティの名前付きノード・マップを戻します。
getNotations()（16-19 ページ）	DTD の表記法の名前付きノード・マップを戻します。

getName()

説明

DTD の名前を戻します。

構文

```
DOMString getName();
```

getEntities()

説明

DTD の（汎用）エンティティのマップを戻します。

構文

```
NamedNodeMap* getEntities();
```

getNotations()

説明

DTD の表記法のマップを戻します。

構文

```
NamedNodeMap* getNotations();
```

DOMImplementation クラス

DOMImplementation の説明

このクラスには、XML パーサーによってサポートされる特定の DOM インプリメンテーションに関連するメソッドが含まれます。

DOMImplementation のメソッド

表 16-5 DOMImplementation のメソッドの概要

メソッド	説明
createDocument()（16-20 ページ）	Document ノードを作成して戻します。
createDocumentType()（16-21 ページ）	Document_Type（DTD）ノードを作成して戻します。
hasFeature()（16-21 ページ）	DOM インプリメンテーションが特定の機能を実装するかどうかをテストします。

createDocument()

説明

Document ノードを作成し、そのノードに対するポインタを戻します。DTD が null 以外の場合は、その Node.ownerDocument 属性が作成中のドキュメントに設定されます。

構文

```
Document *createDocument( DOMString uri,
                          DOMString qname,
                          DocumentType *dtd);
```

パラメータ	説明
dtd	ドキュメント・タイプ（DTD）。
qname	新しいドキュメント要素の修飾名。
uri	新しいドキュメント要素の名前空間 URI。

createDocumentType()

説明

Document_Type（DTD）・ノードを作成し、そのノードに対するポインタを戻します。

構文

```
DocumentType *createDocumentType( DOMString qname,
                                  DOMString pubid,
                                  DOMString sysid);
```

パラメータ	説明
pubid	外部サブセットの公開識別子。
qname	新しいドキュメント要素の修飾名。
sysid	外部サブセットのシステム識別子。

hasFeature()

説明

DOM インプリメンテーションが特定の機能を実装するかどうかをテストします。その機能がサポートされている場合は true、サポートされていない場合は false を戻します。

構文

```
boolean hasFeature( DOMString feature,
                    DOMString version);
```

パラメータ	説明
feature	テストする機能のパッケージ名。レベル 1 では、正当な値は「HTML」および「XML」です（大 / 小文字は区別されない）。
version	テストするパッケージ名のバージョン番号。レベル 1 では、これは文字列「1.0」です。バージョンが指定されていない場合は、この機能のすべてのバージョンがサポートされるため、true を戻します。

Element クラス

Element の説明

このクラスには、要素ノードに関連するメソッドが含まれます。

Element のメソッド

表 16-6 Element のメソッドの概要

メソッド	説明
getAttribute()（16-22 ページ）	指定された名前の属性を選択します。
getAttributeNode()（16-23 ページ）	指定された名前の属性を削除します。
getElementsByTagName()（16-23 ページ）	指定されたタグ名の要素ノードのリストを戻します。
getElementsByTagNameNS()（16-24 ページ）	指定されたタグ名の（名前空間を認識する）要素ノードのリストを戻します。
getTagName()（16-24 ページ）	ノードのタグ名を戻します。
initialize()（16-24 ページ）	新しい要素ノードを初期化します。
normalize()（16-25 ページ）	要素を正規化します（隣接した Text ノードをマージします）。
removeAttribute()（16-25 ページ）	指定された名前の属性を削除します。
removeAttributeNode()（16-25 ページ）	属性ノードを削除します。
setAttribute()（16-26 ページ）	指定された名前および値の新しい属性を作成します。
setAttributeNode()（16-26 ページ）	新しい属性ノードを追加します。

getAttribute()

説明

指定された属性の値（定義）を戻します。

構文

```
DOMString getAttribute(  DOMString name);
```

パラメータ	説明
name	属性名。

getAttributeNode()

説明

指定された属性に対するポインタを戻します。ポインタが検出されない場合は NONE を戻します。

構文

```
Attr* getAttributeNode(  DOMString name);
```

パラメータ	説明
name	属性名。

getElementsByTagName()

説明

一致する要素のリストを作成して戻します。

構文

```
NodeList* getElementsByTagName( DOMString name);
```

パラメータ	説明
name	一致させるタグ名。すべてのタグ名を一致させるには「*」を指定します。

getElementsByTagNameNS()

説明

一致する（名前空間を認識する）要素のリストを作成して戻します。

構文

```
NodeList* getElementsByTagNameNS( DOMString nspuri,
                                  DOMString local);
```

パラメータ	説明
local	一致させるローカル名。すべてのローカル名を一致させるには「*」を指定します。
nspuri	名前空間 URI。

getTagName()

説明

要素のタグ名を戻します。W3C のワーキング・グループでは、DOM 仕様について、「Node インタフェースには汎用の nodeName 属性があるにもかかわらず、Element インタフェース上には tagName 属性がある。これらの 2 つの属性には同じ値が含まれている必要があるが、ワーキング・グループでは、DOM API が様々なユーザー層のニーズを満たす必要があることを考慮し、両方をサポートする価値がある」と考えています。

構文

```
DOMString getTagName();
```

initialize()

説明

新しく割り当てられた要素ノードを初期化します。正常に初期化された場合は 0（ゼロ）、正常に初期化されなかった場合はエラー・コードを戻します。

構文

```
uword initialize( Document *doc,
                  DOMString nspuri,
                  DOMString qname);
```

パラメータ	説明
doc	XML 文書。
nspuri	名前空間 URI。
qname	修飾名。

normalize()

説明
隣接するすべての Text ノードをマージして、要素を正規化します。

構文
```
void normalize(void);
```

removeAttribute()

説明
指定された属性を削除します。

構文
```
void removeAttribute(  DOMString name);
```

パラメータ	説明
name	削除する属性の名前。

removeAttributeNode()

説明
指定された属性ノードを削除して戻します。

構文
```
Attr* removeAttributeNode(  Attr* oldAttr);
```

パラメータ	説明
name	削除する属性。

setAttribute()

説明

新しい属性を作成します。新しく作成された属性に対するポインタを戻します。

構文

```
Attr* setAttribute( DOMString name,
                    DOMString value);
```

パラメータ	説明
name	新しい属性の名前。
value	新しい属性の値。

setAttributeNode()

説明

新しい属性ノードを設定（追加）します。

構文

```
boolean setAttributeNode( Attr* newAttr,
                          Attr** oldAttr);
```

パラメータ	説明
newAttr	新しい属性に対するポインタ。
oldAttr	置換された属性に対する戻されたポインタ。

Entity クラス

Entity の説明

このクラスは、Node のサブクラスである ENTITY ノードのタイプを実装します。

Entity のメソッド

表 16-7 Entity のメソッドの概要

メソッド	説明
getNotationName()（16-27 ページ）	エンティティの NDATA（表記法名）を戻します。
getPublicId()（16-27 ページ）	エンティティの公開識別子を戻します。
getSystemId()（16-28 ページ）	エンティティのシステム識別子を戻します。

getNotationName()

説明

エンティティ・ノードの表記法名（NDATA）を戻します。

構文

```
DOMString getNotationName();
```

getPublicId()

説明

エンティティ・ノードの公開識別子を戻します。

構文

```
DOMString getPublicId();
```

getSystemId()

説明

エンティティ・ノードのシステム識別子を戻します。

構文

```
DOMString getSystemId();
```

EntityReference クラス

EntityReference の説明

このクラスは、Node のサブクラスである ENTITY_REFERENCE ノードのタイプを実装します。

NamedNodeMap クラス

NamedNodeMap の説明

このクラスには、ノード・マップ内のノード番号にアクセスし、個々のノードをフェッチするためのメソッドが含まれます。

NamedNodeMap のメソッド

表 16-8 NamedNodeMap のメソッドの概要

メソッド	説明
free()（16-30 ページ）	名前付きノード・マップを解放します。
getLength()（16-30 ページ）	マップ内のノード数を戻します。
getNamedItem()（16-31 ページ）	名前ごとにノードを選択します。
item()（16-31 ページ）	マップ内の *n* 番目のノードを戻します。
removeNamedItem()（16-32 ページ）	マップから指定のノードを削除します。
setNamedItem()（16-32 ページ）	ノードをマップに設定します。

free()

説明

名前付きノード・マップを解放します。

構文

```
void free();
```

getLength()

説明

マップ内のノード数を戻します。

構文

```
size_t getLength();
```

getNamedItem()

説明

マップから、指定された名前のノードを選択します。検出されない場合は null を戻します。

構文

```
Node* getNamedItem(  String name);
```

パラメータ	説明
name	選択するノードの名前。

item()

説明

ノード・マップ内の *n* 番目に対するポインタを戻します。

構文

```
Node* item(  size_t index);
```

パラメータ	説明
index	インデックス（0（ゼロ）から始まるノード番号）。

removeNamedItem()

説明

ノード・マップから、指定された名前のノードを削除します。削除されたノードに対するポインタを戻します。ポインタが検出されない場合は null を戻します。

構文

```
Node* removeNamedItem(  DOMString name);
```

パラメータ	説明
name	削除するノードの名前。

setNamedItem()

説明

ノードをマップに追加し、同じ名前で存在しているノードを置換します。

構文

```
boolean setNamedItem( Node *node,
                      Node **old);
```

パラメータ	説明
node	追加するノードの名前。
old	置換されたノードに対するポインタ。ノードが新しい場合は null を指定します。

Node クラス

Node の説明

このクラスには、ドキュメント・ノードの詳細なメソッドが含まれます。

Node のメソッド

表 16-9 Node のメソッドの概要

メソッド	説明
appendChild()（16-34 ページ）	現行ノードの子ノード・リストの最後に、新しい子を追加します。
cloneNode()（16-35 ページ）	既存のノード、およびそのすべての子ノード（オプション）を複製します。
getAttributes()（16-35 ページ）	すべての定義済ノード属性を含む構造を戻します。
getChildNode()（16-35 ページ）	指定されたノードのインデックス付けされた特定の子を戻します。
getChildNodes()（16-36 ページ）	指定されたノードのすべての子ノードを含む構造を戻します。
getContext()（16-36 ページ）	ノードのコンテキストを取得します。
getFirstChild()（16-36 ページ）	指定されたノードの最初の子ノードを戻します。
getLastChild()（16-37 ページ）	指定されたノードの最後の子ノードを戻します。
getLocal()（16-37 ページ）	ノードのローカル名を戻します。
getName()（16-37 ページ）	ノード名を戻します。
getNamespace()（16-37 ページ）	ノードの名前空間を戻します。
getNextSibling()（16-38 ページ）	ノードの直後の兄弟関係を戻します。
getOwnerDocument()（16-38 ページ）	ノードを含むドキュメント・ノードを戻します。
getParentNode()（16-38 ページ）	指定されたノードの親ノードを戻します。
getPrefix()（16-38 ページ）	ノードの名前空間の接頭辞を戻します。
getPreviousSibling()（16-39 ページ）	現行ノードの直前の兄弟関係を戻します。
getQualifiedName()（16-39 ページ）	指定されたノードの名前空間で修飾されたノードを戻します。

表 16-9 Node のメソッドの概要（続き）

メソッド	説明
getType()（16-39 ページ）	ノードの数値型コードを戻します。
getValue()（16-40 ページ）	ノードの値（データ）を戻します。
hasAttributes()（16-40 ページ）	ノードに定義済属性があるかどうかを判別します。
hasChildNodes()（16-40 ページ）	ノードが子ノードを持っているかどうかを判別します。
insertBefore()（16-40 ページ）	ノードの子のリストに新しい子ノードを挿入します。
numChildNodes()（16-41 ページ）	指定されたノードの子ノード数のカウントを戻します。
print()（16-41 ページ）	XML 文書をストリームまたはバッファ用にフォーマットします。
printSize()（16-42 ページ）	フォーマット対象の XML 文書のサイズを判別します（出力はしません）。
removeChild()（16-42 ページ）	現行ノードの子ノード・リストからノードを削除します。
replaceChild()（16-43 ページ）	子ノードを別のノードと置換します。
setValue()（16-43 ページ）	ノードの値（データ）を設定します。

appendChild()

説明

現行ノードの子ノード・リストに新しい子を追加し、そのノードに対するポインタを戻します。

構文

```
Node* appendChild(  Node *newChild);
```

パラメータ	説明
newChild	新しい子ノード。

cloneNode()

説明

ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。新しい複製に対するポインタを戻します。複製ノードは親を持ちません（parentNode() は null を戻します）。

要素を複製すると、デフォルトの属性を表すために XML プロセッサによって生成された属性を含む、すべての属性およびその値がコピーされます。ただし、このメソッドでは、ノードに含まれるテキストは、子の Text ノードに含まれているため、ディープ・クローンでないかぎりコピーされません。他のタイプのノードを複製すると、単純にそのノードのコピーが戻されます。

構文

```
Node* cloneNode(  boolean deep);
```

パラメータ	説明
deep	再帰フラグ。

getAttributes()

説明

ノードのすべての属性構造を取得します。ノードのすべての属性を説明する構造に対するポインタを戻します。属性が定義されていない場合は null を戻します。

構文

```
NamedNodeMap* getAttributes();
```

getChildNode()

説明

ノードの子ノードの 1 つを、インデックスを介して検出されたノードの子ノードに対するポインタとして戻します。

構文

```
Node* getChildNode(  uword index);
```

パラメータ	説明
index	子ノードの番号（0（ゼロ）から開始）。

getChildNodes()

説明

ノードの子ノードを、ノードのすべての子ノードを説明する構造に対するポインタとして戻します。

構文

```
NodeList* getChildNodes();
```

getContext()

説明

ノードのコンテキストを戻します。

構文

```
xmlctx* getContext();
```

getFirstChild()

説明

ノードの最初の子ノードに対するポインタを戻します。

構文

```
Node* getFirstChild();
```

getLastChild()

説明
ノードの最後の子ノードに対するポインタを戻します。

構文
```
Node* getLastChild();
```

getLocal()

説明
ノードのローカル名を戻します。

構文
```
DOMString getLocal();
```

getName()

説明
ノードの名前を戻します。ノードが名前を持たない場合は null を戻します。

構文
```
DOMString getName();
```

getNamespace()

説明
ノードの名前空間を戻します。null になる場合があります。

構文
```
DOMString getNamespace();
```

getNextSibling()

説明

ノードの直後の兄弟関係（親ノードの次の子ノード）を戻します。現在の子ノードが最後である場合は null を戻します。

構文

```
Node* getNextSibling();
```

getOwnerDocument()

説明

現行ノードを含むドキュメント・ノードを、そのドキュメント・ノードに対するポインタとして戻します。

構文

```
Document* getOwnerDocument();
```

getParentNode()

説明

ノードの親ノードを戻します。

構文

```
Node* getParentNode();
```

getPrefix()

説明

ノードの名前空間接頭辞を戻します。null になる場合があります。

構文

```
DOMString getPrefix();
```

getPreviousSibling()

説明

ノードの直前の兄弟関係（親ノードの前の子ノード）を戻します。現在の子ノードが最初である場合は null を戻します。

構文

```
Node* getPreviousSibling();
```

getQualifiedName()

説明

ノードの完全修飾（名前空間）名を戻します。

構文

```
DOMString getQualifiedName();
```

getType()

説明

ノードの数値型コードを戻します。戻される可能性があるコードは、次のとおりです。

DOCUMENT_FRAGMENT_NODE	ENTITY_NODE
ELEMENT_NODE	PROCESSING_INSTRUCTION_NODE
NOTATION_NODE	COMMENT_NODE
ATTRIBUTE_NODE	DOCUMENT_NODE
TEXT_NODE	DOCUMENT_TYPE_NODE
ENTITY_REFERENCE_NODE	CDATA_SECTION_NODE

構文

```
short getType();
```

getValue()

説明

ノードの値（データ）を戻します。ノードが値を持たない場合は null を戻します。

構文

```
DOMString getValue();
```

hasAttributes()

説明

ノードに定義済属性があるかどうかを判別します。ノードに属性がある場合は true、ない場合は false を戻します。

構文

```
boolean hasAttributes();
```

hasChildNodes()

説明

ノードに子ノードがあるかどうかを判別します。ノードに子ノードがある場合は true、ない場合は false を戻します。

構文

```
boolean hasChildNodes();
```

insertBefore()

説明

参照ノードの前に、新しい子ノードを親ノードの子ノード・リストに挿入します。新しい子ノードに対するポインタを戻します。参照ノードが null の場合は、新しいノードを最後に追加します。

構文

```
Node* insertBefore( Node *newChild,
                    Node *refChild);
```

パラメータ	説明
newChild	挿入する新しいノード。
refChild	参照ノード。新しいノードが前に挿入されます。

numChildNodes()

説明

ノードの子ノードのカウントを戻します。0（ゼロ）になる場合があります。

構文

```
uword numChildNodes();
```

print()

説明

XML 文書をストリームまたはバッファ用にフォーマットします。次の表に、オプションを示します。

構文	説明
void print(File *out = stdout, uword level = 0, uwoard step = 4);	新しいパーサー・オブジェクトを作成します。
void print(DOMString buffer, size_t bufsize, uword level = 0, uword step =4);	指定された ID を持つ新しいパーサー・オブジェクトを作成します。

パラメータ	説明
out	標準入出力ストリーム。
level	インデントの開始レベル。
step	各インデント・レベルの空白。

パラメータ	説明
buffer	宛先バッファ。
bufsize	宛先バッファのサイズ。

printSize()

説明

フォーマット済 XML 文書を実際に出力せずに、そのサイズ（バイト単位）を戻します。バッファを割り当てる目的で、文書の最終的なサイズを判別する場合に有効です。このファンクションのコールは、文書を実際に出力する場合と同等の高コストな操作であることに注意してください。

構文

```
size_t printSize( uword level = 0,
                  uwoard step = 4);
```

パラメータ	説明
level	インデントの開始レベル。
step	各インデント・レベルの空白。

removeChild()

説明

現行ノードの子ノード・リストから子ノードを削除します。

構文

```
Node* removeChild();
```

replaceChild()

説明

1 つのノードを他のノードで置換します。oldChild の親ノードの子ノード・リストに存在する oldChild を newChild で置換します。oldChild を戻します。

構文

```
Node* replaceChild( Node *newChild,
                    Node *oldChild);
```

パラメータ	説明
newChild	新しく置換するノード。
oldChild	置換される古いノード。

setValue()

説明

ノードの値（データ）を設定します。

構文

```
void setValue(DOMString data);
```

パラメータ	説明
data	ノードの新しいデータ。

NodeList クラス

NodeList の説明

このクラスには、NodeList からノードを取得するためのメソッドが含まれます。

NodeList のメソッド

表 16-10 NodeList のメソッドの概要

メソッド	説明
free()（16-44 ページ）	NodeList を解放します。
getLength()（16-44 ページ）	リスト内のノード数を戻します。
item()（16-45 ページ）	リスト内の *n* 番目のノードを戻します。

free()

説明

NodeList を解放します。

構文

```
void free();
```

getLength()

説明

リスト内のノード数を戻します。

構文

```
size_t getLength();
```

item()

説明

NodeList 内の *n* 番目のノードを戻します。

構文

```
Node* item( size_t index);
```

パラメータ	説明
index	ノード番号（0（ゼロ）から開始）。

Notation クラス

Notation の説明

このクラスは、Node のサブクラスである NOTATION ノードのタイプを実装します。

Notation のメソッド

表 16-11 Notation のメソッドの概要

メソッド	説明
getData()（16-46 ページ）	表記法のデータを戻します。
getTarget()（16-46 ページ）	表記法のターゲットを戻します。
setData()（16-47 ページ）	表記法のデータを設定します。

getData()

説明

表記法のデータを戻します。

構文

```
DOMString getData();
```

getTarget()

説明

表記法のターゲットを戻します。

構文

```
DOMString getTarget();
```

setData()

説明

表記法のデータを設定します。

構文

```
void setData(  DOMString data);
```

パラメータ	説明
data	新しいデータ。

ProcessingInstruction クラス

ProcessingInstruction の説明

このクラスは、Node のサブクラスである PROCESSING_INSTRUCTION ノードのタイプを実装します。

ProcessingInstruction のメソッド

表 16-12 ProcessingInstruction のメソッドの概要

メソッド	説明
getData()（16-48 ページ）	処理命令のデータを戻します。
getTarget()（16-48 ページ）	処理命令のターゲットを戻します。
setData()（16-49 ページ）	処理命令のデータを設定します。

getData()

説明

処理命令のデータを戻します。

構文

```
DOMString getData();
```

getTarget()

説明

処理命令のターゲット値を戻します。

構文

```
DOMString getTarget();
```

setData()

説明

処理命令にデータを設定します。

構文

```
void setData(DOMString data);
```

パラメータ	説明
data	処理命令の新しいデータ。

Text クラス

Text の説明

このクラスには、Text ノードに対応付けられたデータにアクセスし、変更するためのメソッドが含まれます（サブクラス CharacterData）。

Text のメソッド

splitText()

説明

Text ノードのデータ（値）を取得します。Text ノードを 2 つに分割します。元のノードは分割点までのデータを保持し、残りのデータは新しい Text ノードに含まれます。これが後で次のノードになります。新しい Text ノードに対するポインタを戻します。

構文

```
Text* splitText( unsigned long offset);
```

パラメータ	説明
offset	分割点。

XMLParser クラス

XMLParser の説明

このクラスには、パーサーを起動し、ドキュメントに関する高レベルの情報を戻すためのトップレベル・メソッドが含まれます。

XMLParser のメソッド

表 16-13 XMLParser のメソッドの概要

メソッド	説明
context()（16-52 ページ）	コンテキストを取得します。
createDocument()（16-52 ページ）	ドキュメント・ノードを作成し、そのノードに対するポインタを戻します。
getContent()（16-53 ページ）	ノードの内容モデルを戻します。
getDocType()（16-53 ページ）	DTD を説明する「DOCTYPE」構造に対するポインタを戻します。
getDocument()（16-53 ページ）	ドキュメントが正常に解析された後で、ドキュメントのルート・ノードに対するポインタを戻します。
getDocumentElement()（16-54 ページ）	ドキュメントが正常に解析された後で、ドキュメントのルート要素（ノード）に対するポインタを戻します。
getEncoding()（16-54 ページ）	現行ドキュメントの文字コード体系の名前（「ASCII」、「UTF-8」など）を戻します。
isSingleChar()（16-54 ページ）	現行ドキュメントを、「ASCII」などのシングルバイト文字としてエンコーディングするか、マルチバイト文字としてエンコーディングするかを指定するフラグを戻します。
isStandalone()（16-54 ページ）	ドキュメントが、<?xml?> 行上に単独で指定されている場合は、`true` を戻します。それ以外の場合は、`false` を戻します。
setAccess()（16-55 ページ）	指定されたアクセス・メソッドに対して I/O コールバック・ファンクションを設定します。
validate()（16-57 ページ）	ドキュメントを検証します。
xmlclean()（16-57 ページ）	前回の解析中に使用されたすべてのメモリーを解放します。
xmlinit()（16-58 ページ）	XML パーサーを初期化します。
xmlinitenc()（16-58 ページ）	エンコーディングされた XML パーサーを初期化します。

表 16-13 XMLParser のメソッドの概要（続き）

メソッド	説明
xmlparse()（16-59 ページ）	ドキュメントを解析します。
xmlparseBuffer()（16-61 ページ）	バッファを解析します。
xmlparseDTD()（16-62 ページ）	DTD を解析します。
xmlparseFile()（16-64 ページ）	ファイルからドキュメントを解析します。
xmlparseStream()（16-65 ページ）	エラー位置の情報を戻します。
xmlwhere()（16-66 ページ）	コンテキストを取得します。
xmlterm()（16-67 ページ）	ドキュメント・ノードを作成し、そのノードに対するポインタを戻します。

context()

説明

コンテキストを取得します。

構文

```
xmlctx* context();
```

createDocument()

説明

ドキュメント・ノードを作成し、そのノードに対するポインタを戻します。DTD が null 以外の場合は、その Node.ownerDocument 属性が作成中のドキュメントに設定されます。

構文

```
Document* createDocument( DOMString uri,
                          DOMString qname,
                          DocumentType *dtd);
```

パラメータ	説明
uri	新しいドキュメント要素の名前空間 URI。
qname	新しいドキュメント要素の修飾名。
dtd	ドキュメント・タイプ（DTD）。

getContent()

説明

ノードの内容モデルを戻します。内容モデルのノードは Node で、解析対象ドキュメントと同じファンクションを使用して検索および検査できます。

構文

```
Node* getContent(Node *node);
```

パラメータ	説明
node	戻す内容モデルを含むノード。

getDocType()

説明

DTD を記述する「DOCTYPE」構造に対するポインタを戻します。

構文

```
DocumentType* getDocType();
```

getDocument()

説明

ドキュメントが正常に解析された後で、ドキュメントのルート・ノードに対するポインタを戻します。ルートの要素ノードを戻す `getDocumentElement` と比較してください。

構文

```
Document* getDocument();
```

getDocumentElement()

説明

ドキュメントが正常に解析された後で、ドキュメントのルート要素（ノード）に対するポインタを戻します。

構文

```
Element* getDocumentElement();
```

getEncoding()

説明

現行ドキュメントの文字コード体系の名前（「ASCII」、「UTF-8」など）を戻します。エンコーディングがシングルバイトかマルチバイトかのみを戻す isSingleChar フラグと比較してください。

構文

```
DOMString getEncoding();
```

isSingleChar()

説明

現行ドキュメントを「ASCII」などのシングルバイト文字としてエンコーディングするか、「UTF-8」などのマルチバイト文字としてエンコーディングするかを指定するフラグを戻します。ドキュメントの実際のエンコーディング名を戻す getEncoding と比較してください。

構文

```
boolean isSingleChar();
```

isStandalone()

説明

ドキュメントが、<?xml?> 行上に単独で指定されている場合、true を戻します。それ以外の場合は、false を戻します。

構文

```
boolean isStandalone();
```

setAccess()

説明

指定されたアクセス・メソッドに対して I/O コールバック・ファンクションを設定します。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

ほとんどのメソッドには組込みコールバック・ファンクションが含まれているため、ユーザーが指定する必要はありません。ただし、XMLACCESS_STREAM の場合は、ユーザー自身がストリーム・コールバック・ファンクションを設定する必要があります。

次の 3 つのコールバック・ファンクションを起動すると、入力ソースのオープン、クローズおよび読込みが行われます。これらのファンクションは、ファンクション・プロトタイプ・マクロ（XML_OPENF、XML_CLOSEF および XML_READF）を使用して宣言しておく必要があります。

XML_OPENF は、オープン用ファンクションで、入力ソースをオープンするために 1 回コールされます。xmlihdl 共用体で永続ハンドルを設定する必要があります。永続ハンドルには、汎用ポインタ（void *）または整数（UNIX ファイルまたはソケット・ハンドル）のいずれかを使用できます。このファンクションは、正常に終実行されると XMLERR_OK を戻します。

パラメータ	IN / OUT	説明
ctx	(IN)	XML コンテキスト。
ih	(OUT)	オープンされたハンドルが置かれる場所。
length	(OUT)	既知の場合、入力ソースの合計の長さ（不明の場合は 0（ゼロ））。
parts	(IN)	構成要素に分割されたパス（不透明ポインタ）。
path	(IN)	オープンするソースへのフルパス。

XML-CLOSEF は、クローズ・ファンクションで、オープンされたソースをクローズし、リソースを解放します。

パラメータ	IN / OUT	説明
ctx	(IN)	XML コンテキスト。
ih	(IN)	入力ハンドル共用体。

XML-READF は、リーダー・ファンクションで、オープンされたソースからバッファにデータを読み込み、読み込んだバイト数を戻します。

EOI が戻された場合は、対応するクローズ・ファンクションが自動的にコールされます。

パラメータ	IN / OUT	説明
ctx	(IN)	XML コンテキスト。
ih	(IN)	入力ハンドル共用体。
dest	(OUT)	データの読込み先の宛先バッファ。
destsize	(IN)	宛先のサイズ。
path	(IN)	オープンするソースへのフルパス。エラー・メッセージでの使用にのみ提供されます。
nraw	(OUT)	読み込まれるバイト数。
eoi	(OUT)	情報の終わりが検出されたかどうかを判別します。各読込みの終了後に `true` または `false` に設定する必要があります。

構文

```
uword setAccess( xmlctx *ctx,
                 xmlacctype access,
                 XML_OPENF((*openf)),
                 XML_CLOSEF( (*closef)),
                 XML_READF((*readf)));
```

パラメータ	説明
ctx	XML コンテキスト。
access	アクセス・メソッドの列挙（`XMLACCESS_xxx`）。
openf	入力ソースをオープンするコールバック・ファンクション。
closef	入力ソースをクローズするコールバック・ファンクション。
readf	入力ソースを読み込むコールバック・ファンクション。

validate()

説明

ドキュメントを検証します。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

構文

```
uword validate( Node* root);
```

パラメータ	説明
root	検証するドキュメント・ノード。

xmlclean()

説明

前回の解析中に使用されたすべてのメモリーを解放します。XML パーサー内でメモリーを再利用します。ただし、システムには解放しません。xmlterm() によって、最後にすべてのメモリーがシステムに解放されます。解析と解析の間で xmlclean() がコールされない場合は、前回のドキュメントによって使用されたデータが割り当てられ、このデータに対するポインタが有効状態のままになります。そのため、複数のドキュメントのデータに同時にアクセスできます。ただし、DOM で操作できるのは現行のドキュメントのみです。1 つのコンテキスト内で、一度に 1 つのドキュメントのデータのみにアクセスするには、新しく解析を開始する前に xmlclean() をコールする必要があります。

構文

```
void xmlclean();
```

xmlinit()

説明

XML パーサーを初期化します。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

構文

```
uword xmlinit( DOMString encoding = 0,
               XML_MSGHDLRF((*msghdlr)) = 0,
               void *msgctx = 0, xmlsaxcb *saxcb = 0,
               void *saxcbctx = 0,
               DOMString lang = 0 );
```

パラメータ	説明
encoding	デフォルトの入力ファイルのエンコーディング。指定されていない場合は UTF-8 になります。
msghdlr	エラー・メッセージ・コールバック。
msgctx	msghdr に渡されるユーザー定義のコンテキスト・ポインタ。
saxcb	SAX コールバック構造（SAX を使用する場合のみ）。
saxcbctx	SAX コールバック・ファンクションに渡されるユーザー定義の SAX コンテンツ構造。
lang	エラー・メッセージの言語。現在は使用されていません。

xmlinitenc()

説明

エンコーディングされた XML パーサーを初期化します。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

構文

```
uword xmlinitenc( DOMString incoding = 0,
                  DOMString outcoding = 0,
                  XML_MSGHDLRF((*msghdlr)) = 0,
                  void *msgctx = 0, xmlsaxcb *saxcb = 0,
                  void *saxcbctx, DOMString lang = 0 );
```

パラメータ	説明
incoding	デフォルトの入力ファイルのエンコーディング。指定されていない場合は UTF-8 になります。
outcoding	ドキュメント・データの出力（DOM）エンコーディング。指定しない場合は最初の入力と同じになります。
msghdlr	エラー・メッセージ・コールバック。
msgctx	msghdr に渡されるユーザー定義のポインタ。
saxcb	SAX コールバック構造（SAX を使用する場合のみ）。
saxcbctx	SAX コールバック・ファンクションに渡されるユーザー定義の SAX コンテンツ構造。
lang	エラー・メッセージの言語。現在は使用されていません。

xmlparse()

説明

ドキュメントを解析します。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

構文

```
uword xmlparse( DOMString doc,
                DOMString encoding,
                ub4 flags);
```

パラメータ	説明
doc	ドキュメント・パス。
encoding	ドキュメントのエンコーディング。
flags	フラグ・ビットのマスク。

パーサー・フラグ・オプション	説明
XML_DTD_ONLY	外部サブセットを解析します。これは、xmlparsedtd() をコールする場合と同様に、完全な XML 文書ではなく外部サブセット（DTD）を解析します。主に、Class Generator によって使用され、完全な文書を必要とすることなく DTD からクラスを生成できます。
XML_FLAG_DISCARD_WHITESPACE	不要な空白（行の終わりなど）を削除します。デフォルトでは、空白がレポートされ、無視可能な空白が示されます。このオプションによって、要素の終了タグと次の要素の開始タグの間のすべての空白が削除されます。
XML_FLAG_STOP_ON_WARNING	警告が発生すると検証を停止します。このフラグが設定されていないかぎり、検証で発生した問題は警告（致命的ではない）とみなされます。このフラグを設定すると、最初の警告後に検証が停止します。
XML_FLAG_VALIDATE	検証を開始します。デフォルトでは、整形式であるかどうかのみが確認されます。
XML_FORCE_INCODING	入力ドキュメントを強制的にエンコーディング「incoding」で解析します。デフォルトの入力エンコーディングを incoding に指定できます。これによって、xmlinit に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。XML_FLAG_FORCE_INCODING を設定すると、ドキュメントは「incoding」として解析されます。
XML_WARN_DUPLICATE_ENTITY	複製エンティティ宣言が検出された場合、警告が発生します。通常、複製エンティティ宣言は、警告を発生させることなく無視されます。このフラグを設定すると、警告が発生します。

xmlparseBuffer()

説明

バッファを解析します。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

構文

```
uword xmlparseBuffer( DOMString buffer,
                      size_t len,
                      DOMString encoding,
                      ub4 flags);
```

パラメータ	説明
buffer	解析するドキュメントを含むバッファ。
len	ドキュメントの長さ。
encoding	ドキュメントのエンコーディング。
flags	フラグ・ビットのマスク。

パーサー・フラグ・オプション	説明
XML_DTD_ONLY	外部サブセットを解析します。これは、xmlparsedtd() をコールする場合と同様に、完全な XML 文書ではなく外部サブセット（DTD）を解析します。主に、Class Generator によって使用され、完全な文書を必要とすることなく DTD からクラスを生成できます。
XML_FLAG_DISCARD_WHITESPACE	不要な空白（行の終わりなど）を削除します。デフォルトでは、空白がレポートされ、無視可能な空白が示されます。このオプションによって、要素の終了タグと次の要素の開始タグの間のすべての空白が削除されます。
XML_FLAG_STOP_ON_WARNING	警告が発生すると検証を停止します。このフラグが設定されていないかぎり、検証で発生した問題は警告（致命的ではない）とみなされます。このフラグを設定すると、最初の警告後に検証が停止します。
XML_FLAG_VALIDATE	検証を開始します。デフォルトでは、整形式であるかどうかのみが確認されます。

パーサー・フラグ・オプション	説明
XML_FORCE_INCODING	入力ドキュメントを強制的にエンコーディング「incoding」で解析します。デフォルトの入力エンコーディングを incoding に指定できます。これによって、xmlinit に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。XML_FLAG_FORCE_INCODING を設定すると、ドキュメントは「incoding」として解析されます。
XML_WARN_DUPLICATE_ENTITY	複製エンティティ宣言が検出された場合、警告が発生します。通常、複製エンティティ宣言は、警告を発生させることなく無視されます。このフラグを設定すると、警告が発生します。

xmlparseDTD()

説明

DTD を解析します。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

構文

```
uword xmlparseDTD( DOMString uri,
                   DOMString name,
                   DOMString encoding,
                   ub4 flags);
```

パラメータ	説明
uri	DTD を指す URI。
name	DTD 名。
encoding	DTD のエンコーディング。
flags	フラグ・ビットのマスク。

パーサー・フラグ・オプション	説明
XML_DTD_ONLY	外部サブセットを解析します。これは、`xmlparsedtd()` をコールする場合と同様に、完全な XML 文書ではなく外部サブセット（DTD）を解析します。主に、Class Generator によって使用され、完全な文書を必要とすることなく DTD からクラスを生成できます。
XML_FLAG_DISCARD_WHITESPACE	不要な空白（行の終わりなど）を削除します。デフォルトでは、空白がレポートされ、無視可能な空白が示されます。このオプションによって、要素の終了タグと次の要素の開始タグの間のすべての空白が削除されます。
XML_FLAG_STOP_ON_WARNING	警告が発生すると検証を停止します。このフラグが設定されていないかぎり、検証で発生した問題は警告（致命的ではない）とみなされます。このフラグを設定すると、最初の警告後に検証が停止します。
XML_FLAG_VALIDATE	検証を開始します。デフォルトでは、整形式であるかどうかのみが確認されます。
XML_FORCE_INCODING	入力ドキュメントを強制的にエンコーディング「incoding」で解析します。デフォルトの入力エンコーディングを incoding に指定できます。これによって、xmlinit に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。`XML_FLAG_FORCE_INCODING` を設定すると、ドキュメントは「incoding」として解析されます。
XML_WARN_DUPLICATE_ENTITY	複製エンティティ宣言が検出された場合、警告が発生します。通常、複製エンティティ宣言は、警告を発生させることなく無視されます。このフラグを設定すると、警告が発生します。

xmlparseFile()

説明

ファイルからドキュメントを解析します。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

構文

```
uword xmlparseFile( DOMString path,
                    size_t len,
                    DOMString encoding,
                    ub4 flags);
```

パラメータ	説明
path	ドキュメント・パス。
len	未使用のパラメータ。
encoding	ドキュメントのエンコーディング。
flags	フラグ・ビットのマスク。

パーサー・フラグ・オプション	説明
XML_DTD_ONLY	外部サブセットを解析します。これは、xmlparsedtd() をコールする場合と同様に、完全な XML 文書ではなく外部サブセット（DTD）を解析します。主に、Class Generator によって使用され、完全な文書を必要とすることなく DTD からクラスを生成できます。
XML_FLAG_DISCARD_WHITESPACE	不要な空白（行の終わりなど）を削除します。デフォルトでは、空白がレポートされ、無視可能な空白が示されます。このオプションによって、要素の終了タグと次の要素の開始タグの間のすべての空白が削除されます。
XML_FLAG_STOP_ON_WARNING	警告が発生すると検証を停止します。このフラグが設定されていないかぎり、検証で発生した問題は警告（致命的ではない）とみなされます。このフラグを設定すると、最初の警告後に検証が停止します。
XML_FLAG_VALIDATE	検証を開始します。デフォルトでは、整形式であるかどうかのみが確認されます。

パーサー・フラグ・オプション	説明
XML_FORCE_INCODING	入力ドキュメントを強制的にエンコーディング「incoding」で解析します。デフォルトの入力エンコーディングを incoding に指定できます。これによって、xmlinit に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。XML_FLAG_FORCE_INCODING を設定すると、ドキュメントは「incoding」として解析されます。
XML_WARN_DUPLICATE_ENTITY	複製エンティティ宣言が検出された場合、警告が発生します。通常、複製エンティティ宣言は、警告を発生させることなく無視されます。このフラグを設定すると、警告が発生します。

xmlparseStream()

説明

ファイルからドキュメントを解析します。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

構文

```
uword xmlparseStream( DOMString path,
                      void *stream,
                      DOMString encoding,
                      ub4 flags);
```

パラメータ	説明
path	未使用のパラメータ。
stream	入力ストリーム。
encoding	ドキュメントのエンコーディング。
flags	フラグ・ビットのマスク。

パーサー・フラグ・オプション	説明
XML_DTD_ONLY	外部サブセットを解析します。これは、xmlparsedtd() をコールする場合と同様に、完全な XML 文書ではなく外部サブセット（DTD）を解析します。主に、Class Generator によって使用され、完全な文書を必要とすることなく DTD からクラスを生成できます。
XML_FLAG_DISCARD_WHITESPACE	不要な空白（行の終わりなど）を削除します。デフォルトでは、空白がレポートされ、無視可能な空白が示されます。このオプションによって、要素の終了タグと次の要素の開始タグの間のすべての空白が削除されます。
XML_FLAG_STOP_ON_WARNING	警告が発生すると検証を停止します。このフラグが設定されていないかぎり、検証で発生した問題は警告（致命的ではない）とみなされます。このフラグを設定すると、最初の警告後に検証が停止します。
XML_FLAG_VALIDATE	検証を開始します。デフォルトでは、整形式であるかどうかのみが確認されます。
XML_FORCE_INCODING	入力ドキュメントを強制的にエンコーディング「incoding」で解析します。デフォルトの入力エンコーディングを incoding に指定できます。これによって、xmlinit に指定された incoding がオーバーライドされます。入力エンコーディングは、BOM、XMLDecl などに基づいて自動的に判別されない場合、incoding であると想定されます。IANA/MIME エンコーディング名には「UTF-8」、「ASCII」などを使用する必要があります。XML_FLAG_FORCE_INCODING を設定すると、ドキュメントは「incoding」として解析されます。
XML_WARN_DUPLICATE_ENTITY	複製エンティティ宣言が検出された場合、警告が発生します。通常、複製エンティティ宣言は、警告を発生させることなく無視されます。このフラグを設定すると、警告が発生します。

xmlwhere()

説明

エラー位置の情報を戻します。エラーの発生中に、ユーザー・エラー・コールバック・ファンクション内からのみコールする必要があります。

構文

```
boolean xmlwhere( ub4 *line,
                  DOMString *path,
                  uword idx);
```

パラメータ	説明
line	エラーが発生した行番号。
path	エラーが発生したパスまたは URL。
idx	エラー・スタック内の位置（0（ゼロ）から開始）。

xmlterm()

説明

XML パーサーを終了し、メモリーを分解して解放します。

構文

```
void xmlterm();
```

C++ SAX API

C++ SAX API の説明

SAX API は、コールバックに基づいています。ドキュメント全体を解析し、(DOM インタフェースによって）参照されるデータ構造に変換するかわりに、SAX インタフェースはシリアルに動作します。ドキュメントが処理されると、適切な SAX のユーザー・コールバック・ファンクションが起動されます。各コールバック・ファンクションは、エラー・コードを戻します。0（ゼロ）は成功、0（ゼロ）以外の値は失敗を表します。0（ゼロ）以外のコードが戻された場合、ドキュメント処理は停止します。

SAX を使用するには、ファンクション・ポインタを使用して xmlsaxcb 構造を初期化し、xmlinit() コールに渡します。ユーザー定義のコンテキスト構造体に対するポインタを含めることもできます。そのコンテキスト・ポインタは、各 SAX ファンクションに渡されます。

この SAX 機能は、XML Parser for C バージョンの機能と同じです。

SAX コールバック構造

```
typedef struct  {
sword (*startDocument) (void *ctx);
sword (*endoocument) (void *ctx);
sword (*startElement) (void *ctx, const oratext *name,
        const struct xmlnodes *attrs);
sword (*endElement) (void *ctx, const oratext *name);
sword (*characters) (void *ctx, const oratext *ch, size_t len);
sword (*ignorableWhitespace) (void *ctx, const oratext *ch, size_t len);
sword (*processingInstruction) (void *ctx, const oratext *target,
        const oratext *data);
sword (*notationDecl) (void *ctx, const oratext *name, const oratext *publicId,
        const oratext *systemId);
sword (*unparsedEntityDecl) (void *ctx, const oratext *name,
        const oratext *public const oratext *systemId,
        const oratext *notationName);
sword (*nsStartElement) (void *ctx, const oratext *qname,
        const oratext *local, const oratext *nsp,
        const struct xmlnodes *attrs);
sword (*comment) (void *ctx, const oratext *data);
sword (*elementDecl) (void *ctx, const oratext *name,
        const oratext *content);
sword (*attributeDecl) (void *ctx, const oratext *elem,
        const oratext *attr, const oratext *body);
} xmlsaxcb;
```

C++ SAX API のメソッド

表 16-14 C++ SAX API のメソッドの概要

メソッド	説明
startDocument()（16-69 ページ）	ドキュメントの処理を開始します。1 回のみコールされます。
endDocument()（16-70 ページ）	ドキュメントの処理を終了します。1 回のみコールされます。
startElement()（16-70 ページ）	新しいドキュメント要素の処理を開始します。要素ごとに 1 回のみコールされます。
endElement()（16-71 ページ）	ドキュメント要素の処理を終了します。要素ごとに 1 回のみコールされます。
characters()（71 ページ）	リテラル・テキストを処理します。リテラル・テキストのピースごとに 1 回コールされます。
ignorableWhitespace()（16-72 ページ）	無視可能な（重要でない）空白を処理します。無視可能な空白のピースごとに 1 回コールされます。
processingInstruction()（16-72 ページ）	処理命令を処理します。処理命令ごとに 1 回コールされます。
notationDecl（16-73 ページ）	表記法を処理します。表記法ごとに 1 回コールされます。
unparsedEntityDecl()（16-73 ページ）	未解析のエンティティ宣言を処理します。未解析のエンティティ宣言ごとに 1 回コールされます。
nsStartElement()（16-74 ページ）	要素が明示的な名前空間を使用する場合に、新しいドキュメント要素の処理を開始します。要素ごとに 1 回のみコールされます。
comment()（16-75 ページ）	XML ソース・コメントに関する通知を受け取ります。
elementDecl()（16-75 ページ）	要素宣言に関する通知を受け取ります。
attributeDecl()（16-76 ページ）	要素の属性宣言に関する通知を受け取ります。

startDocument()

説明

ドキュメントの処理を開始します。1 回のみコールされます。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword startDocument( void *ctx);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。

endDocument()

説明

ドキュメントの処理を終了します。1 回のみコールされます。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword endDocument( void *ctx);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。

startElement()

説明

新しいドキュメント要素の処理を開始します。要素ごとに 1 回のみコールされます。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword startElement(  void *ctx,
                     const oratext *name,
                     const struct xmlnodes *attrs);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。
name	ノード名。
attrs	ノードの属性の配列。

endElement()

説明

ドキュメント要素の処理を終了します。要素ごとに 1 回のみコールされます。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword endElement( void *ctx,
                  const oratext *name);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。
name	ノード名。

characters()

説明

リテラル・テキストを処理します。リテラル・テキストのピースごとに 1 回コールされます。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword characters( void *ctx,
                  const oratext *ch,
                  size_t len);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。
ch	テキストに対するポインタ。
len	テキスト内の文字数。

ignorableWhitespace()

説明

無視可能な（重要でない）空白を処理します。無視可能な空白のピースごとに 1 回コールされます。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword ignorableWhitespace( void *ctx,
                           const oratext *ch,
                           size_t len);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。
ch	空白テキストに対するポインタ。
len	空白テキスト内の文字数。

processingInstruction()

説明

処理命令を処理します。処理命令ごとに 1 回コールされます。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword processingInstruction( void *ctx,
                             const oratext *target,
                             const oratext *data);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。
data	処理命令のデータ。
target	処理命令のターゲット。

notationDecl

説明

表記法を処理します。表記法ごとに 1 回コールされます。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword notationDecl( void *ctx,
                    const oratext *name,
                    const oratext *publicId,
                    const oratext *systemId);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。
name	表記法名。
publicId	公開識別子。
systemId	システム識別子。

unparsedEntityDecl()

説明

未解析のエンティティ宣言を処理します。未解析のエンティティ宣言ごとに 1 回コールされます。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword unparsedEntityDecl( void *ctx,
                          const oratext *name,
                          const oratext *publicId,
                          const oratext *systemId,
                          const oratext *notationName);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。
name	エンティティ名。

パラメータ	説明
publicId	公開識別子。
systemId	システム識別子。
notationName	表記法名。

nsStartElement()

説明

要素が明示的な名前空間を使用する場合に、新しいドキュメント要素の処理を開始します。要素ごとに 1 回のみコールされます。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword startElement( void *ctx,
                    const oratext *qname,
                    const oratext *local,
                    const oratext *namespace,
                    const struct xmlnodes *attrs);
```

パラメータ	説明
ctx	`initialize()` に渡されるユーザー定義のコンテキスト。
qname	修飾された名前空間。
local	要素のローカル名。
namespace	要素の名前空間。
attrs	指定された属性またはデフォルトの属性。

comment()

説明

XML ソース・コメントに関する通知を受け取ります。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword comment( void *ctx,
               const oratext *data);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。
data	コメントの本体。

elementDecl()

説明

要素宣言に関する通知を受け取ります。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword elementDecl( void *ctx,
                   const oratext *name,
                   const oratext *content);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。
name	宣言されている要素の名前。
content	要素のコンテンツ・モデル。

attributeDecl()

説明

要素の属性宣言に関する通知を受け取ります。正常に終了した場合は 0（ゼロ）、エラーの場合は数値のエラー・コードを戻します。

構文

```
sword attributeDecl( void *ctx,
                     const oratext *elem,
                     const oratext *attr,
                     const oratext *body);
```

パラメータ	説明
ctx	initialize() に渡されるユーザー定義のコンテキスト。
elem	属性が宣言される要素の名前。
attr	宣言される属性の名前。
body	宣言される属性の本体。

C++ DOM API

表 16-15 C++ DOM API クラス

クラス	スーパークラス
Attribute	Node
CDATASection	Text
CharacterData	Node
Comment	CharacterData
Document	Node
DocumentFragment	Node
DocumentType	Node
Element	Node
Entity	Node
EntityReference	Node
Notation	Node
ProcessingInstruction	Node
Text	CharacterData

17

XSLT Processor for C++

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』

XSLProcessor クラス

XSLProcessor の説明

このクラスには、XSLT プロセッサを起動するためのトップレベル・メソッドが含まれます。

XSLProcessor のメソッド

xslprocess()

説明

XML 文書ソースで XSL スタイルシートを処理します。数値のエラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

構文

```
uword xslprocess( XMLParser *docctx,
                  XMLParser *xslctx,
                  XMLParser *resctx,
                  Node **result);
```

引数	IN / OUT	説明
docctx	(IN/OUT)	XML 文書コンテキスト。
xslctx	(IN)	XSL スタイルシート・コンテキスト。
resctx	(IN)	結果のドキュメント・フラグメント・コンテキスト。
result	(IN/OUT)	結果のドキュメント・フラグメント・ノード。

XPObject クラス

XPObject の説明

このクラスには、XPObject クラスのトップレベル・メソッドが含まれます。このクラスのオブジェクトは、XPath 式の評価結果として作成されます。

XPObject のメソッド

表 17-1 XPObject のメソッドの概要

メソッド	説明
XPObject()（17-3 ページ）	XPObject のコンストラクタです。
getbooleanval()（17-3 ページ）	XPObject のブール値を戻します。
getnumval()（17-4 ページ）	XPObject のノード設定値を戻します。
getstrval()（17-4 ページ）	XPObject の数値を戻します。
getnsetval()（17-4 ページ）	XPObject の文字列値を戻します。
getxpobjtyp()（17-5 ページ）	XPObject の型を戻します。

XPObject()

説明

XPObject のコンストラクタです。

構文

```
XPObject(xpobj *xo);
```

getbooleanval()

説明

オブジェクトのブール値を戻します。

構文

```
boolean getbooleanva();
```

getnumval()

説明

オブジェクトの数値を戻します。

構文

```
double getnumval();
```

getstrval()

説明

オブジェクトの文字列値を戻します。

構文

```
oratext* getstrval();
```

getnsetval()

説明

オブジェクトのノード設定値を戻します。

構文

```
xpnset* getnsetval();
```

getxpobjtyp()

説明

オブジェクト型を戻します。戻される型は、次のいずれかです。

- XPOBJTYP_BOOL
- XPOBJTYP_NUM
- XPOBJTYP_STR
- XPOBJTYP_NSET
- XPOBJTYP_RTFRAG

構文

```
xpobjtyp getxpobjtyp();
```

XPath クラス

XPath の説明

このクラスには、XPath プロセッサを起動するためのトップレベル・メソッドが含まれます。

XPath のメソッド

表 17-2 XPath のメソッドの概要

メソッド	説明
XPath()（17-6 ページ）	XPath のコンストラクタです。
~XPath（17-7 ページ）	XPath のデストラクタです。
parsexpathexpr()（17-7 ページ）	XPath 式を解析します。
evalxpathexpr()（17-8 ページ）	事前に解析された XPath 式を評価します。

XPath()

説明

XPath オブジェクトを作成して戻します。このコールは常に正常に実行されます。

構文

```
XPath( xmlctx *ctx,
       xmlnode *xslnode,
       xmlnode* xml_node,
       oratext* baseURI,
       size_t nctxels,
       xmlnode **ctxnodes);
```

引数	IN / OUT	説明
ctx	(IN)	XML コンテキスト。null に指定できます。
xslnode	(IN)	名前空間の拡張に使用される XSL ノード。null に設定できます。
xml_node	(IN)	コンテキスト・ノード。解析のために null に設定します。

引数	IN / OUT	説明
baseURI	(IN)	解析のためのベース URI。
nctxels	(IN)	現行のノード・セット内のノードの数。
ctxnodes	(IN)	現行のノード・セット。

~XPath

説明

XPath オブジェクトを削除または破棄します。

構文

```
~XPath();
```

parsexpathexpr()

説明

XPath 式を解析します。正常に終了した場合は式ツリー、エラーが発生した場合は null を戻します。

構文

```
xpexpr* parsexpathexpr( oratext *expr,
                        sword *err);
```

引数	IN / OUT	説明
expr	(IN)	文字列形式の式。
err	(OUT)	エラー・コード。

evalxpathexpr()

説明

XPath 式を評価します。正常に終了した場合は結果オブジェクト、エラーが発生した場合は null を戻します。getxpobjtyp() をコールして、次のいずれかの型の結果オブジェクトが取得されます。

- XPOBJTYP_BOOL
- XPOBJTYP_NUM
- XPOBJTYP_STR
- XPOBJTYP_NSET
- XPOBJTYP_RTFRAG

構文

```
XPObject* evalxpathexpr( xpexpr *exprtree,
                         sword *err);
```

引数	IN / OUT	説明
exprtree	(IN)	ツリー形式の式。
err	(OUT)	エラー・コード。

18

XML Schema Processor for C++

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』

XMLSchema クラス

XMLSchema の説明

スキーマ API は非常に単純で、初期化、検証、... 検証、終了の順に処理されます。

検証プロセスでは、有効か無効化を判別します。ドキュメントは、スキーマに対して有効か無効かのいずれかです。ドキュメントが有効である場合は、エラー・コード 0（ゼロ）が戻されます。ドキュメントが無効である場合は、0（ゼロ）以外のエラー・コードが戻され、問題が示されます。警告とエラーの区別はありません。すべての問題はエラーであり、致命的であるとみなされ、検証はすぐに停止します。

検出されたスキーマはロードされ、スキーマ・コンテキストに保持されます。セッション中、スキーマは 1 回のみロードされます。XML パーサー・ファンクション `xmlclean()` と同様のクリーンアップ・コールはありません。そのため、新しいドキュメントを検証する前にすべてのメモリーを解放し、状態をリセットする必要がある場合は、コンテキストを終了し、最初からやりなおす必要があります。

XMLSchema のファンクション

表 18-1 XMLSchema のファンクションの概要

ファンクション	説明
initialize()（18-2 ページ）	XML Schema プロセッサを初期化します。
load()（18-3 ページ）	追加のスキーマ・ファイルをロードします。
target()（18-3 ページ）	ターゲットの名前空間の URI を戻します。
terminate()（18-4 ページ）	XML Schema プロセッサを終了します。
validate()（18-4 ページ）	スキーマに対してインスタンス・ドキュメントを検証します。

initialize()

説明

XML Schema プロセッサを初期化します。エラー・コードを戻します。正常に終了した場合は、`0`（ゼロ）を戻します。

構文

```
uword initialize( XMLParser *parser);
```

引数	説明
parser	メモリーの割当てに使用される XMLParser オブジェクト。

load()

説明

スキーマを明示的にロードします（ロードされていない場合）。新しいスキーマまたは古いスキーマに対するポインタを戻します。デフォルト・スキーマの URL は検証メソッドに直接渡せるため、通常は、このメソッドを使用する必要はありません。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

構文

```
uword load( oratext *uri,
            oratext *nsp,
            Schema **schema);
```

引数	説明
uri	コンパイラのキャラクタ・セットで表されたスキーマの URL。
nsp	コンパイラのキャラクタ・セットで表されたスキーマの名前空間（null になる場合があります）。
schema	スキーマに対する、戻されたポインタ。

target()

説明

load() によって戻されるスキーマ・ハンドルを指定して、ターゲットの名前空間 URI を戻します。

構文

```
oratext *target( Schema *schema);
```

引数	説明
schema	load() メソッドによって戻されるスキーマ・ハンドル。

terminate()

説明

XML Schema プロセッサを終了し、メモリーの分解、解放などを行います。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。

構文

```
void terminate(void);
```

validate()

説明

スキーマに対してインスタンス・ドキュメントを検証します。

構文

```
uword validate( Element *root,
                oratext *url);
```

引数	説明
root	parser.getDocumentElement() によって戻されるドキュメントのルート要素。
url	デフォルト・スキーマの URL（オプション）。

19

XML Class Generator for C++

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』

XML Class Generator for C++ の概要

XML Class Generator for C++ は Document Type Definition（DTD）または XML Schema を使用して、各定義済要素のクラスを生成します。その後、C++ プログラムでこれらのクラスを使用して、DTD に準拠する XML 文書を作成します。

入力

入力は、DTD、外部 DTD または XML Schema を含む XML 文書です。完全な XML 文書を入力すると、文書本体自体は無視されます。DTD のみが関連しますが、ダミー・ドキュメントは DTD に準拠する必要があります。

出力

完全な XML 文書を入力した場合の出力は、DTD の後に指定された C++ ソース・ファイル .cpp と .h の組合せです。外部 DTD またはスキーマの場合は、生成されたファイルの名前を指定する必要があります。各クラス（要素）用に提供されているコンストラクタを使用すると、オブジェクトを作成できます。オブジェクトを作成するには、最初は空で作成し、その後に子またはデータを追加するか、または最初から子や初期データの完全なセットで作成する 2 つの方法があります。#PCDATA（および Mixed）要素用に提供されているメソッドを使用すると、データを設定したり、適切な場合は、要素の属性を設定することができます。

関連する XML 標準

- W3C の eXtensible Markup Language（XML）1.0 勧告
- W3C のドキュメント・オブジェクト・モデル（DOM）レベル 1 1.0 勧告
- W3C の XML Namespace 勧告
- Simple API for XML（SAX）1.0

使用例

単独パーサーは、次のようにコールすることによって、実行可能ファイルとしてコールできます。

```
bin/xmlcg like
xmlcg [flags] <XML document>
```

表 19-1 オプション・フラグ

フラグ	コマンド	説明
-d	directory	出力ディレクトリを指定します。デフォルトは現行のディレクトリです。
-e	encoding	デフォルトの入力ファイルのエンコーディングを指定します。
-h	help	使用方法のヘルプを表示します。

XMLClassGenerator クラス

XMLClassGenerator の説明

このクラスには、DTD またはスキーマ（XML Schema）に基づいてクラスを作成するためのメソッドが含まれます。

XMLClassGenerator のメソッド

generate()

説明

指定された DTD またはスキーマのクラスを生成します。エラー・コードを戻します。正常に終了した場合は、0（ゼロ）を戻します。次の表に、オプションを示します。

構文	説明
uword XMLClassGenerator::generate(DocumentType *dtd, DOMString outdir);	指定された DTD のクラスを生成します。出力ディレクトリ outdir（outdir が null の場合は、現行のディレクトリ）に、DTD に従って名前付けされた DTDname.h および DTDname.cpp の 2 つのファイルが作成されます。DTD 内の定義済要素ごとに 1 つのクラスが生成されます。
uword XMLClassGenerator::generate(Schema *schema, DOMString name, DOMString outdir);	指定されたスキーマのクラスを生成します。出力ディレクトリ outdir（outdir が null の場合は、適切なディレクトリ）に、name.h および name.cpp の 2 つのファイルが作成されます。スキーマ内の要素ごとに 1 つのクラスが生成されます。

パラメータ	説明
dtd	クラスの生成対象となる要素を含む DTD。
outdir	出力ファイルを置くディレクトリ。
schema	クラスの生成対象となる要素を含むスキーマ。
name	生成されたファイルおよび囲み C++ 名前空間の名前。

generated クラス（DTD 用）

generated の説明

generated クラスは、DTD 内の定義済要素ごとに作成されます。名前は要素と同じです。

要素（または属性）の名前は、C++ 識別子として直接使用できない場合、その名前をコンパイラのキャラクタ・セット（ASCII または EBCDIC）に変換後、マップできない文字をそれらのコード・ポイントの 2 文字の 16 進数に置き換えることによって、有効な識別子にマップされます。たとえば、要素名「Curçao」は、「Curacao」にマップされます。再マップされた名前がすでに使用されている場合は、名前が一意になるように、その末尾に数字が追加されます（たとえば「Curacao0」）。generated クラスによって作成された要素および属性は元の名前を持つことに注意してください。再マッピングは、C++ での構文上正しくなるように、生成されたコードにのみ適用されます。再マッピングは、これらの要素によって作成された XML 要素およびデータには適用されません。

要素を作成するには、空の要素の作成後に子を 1 つずつ追加する方法、および初期データまたは子を使用して要素を作成する方法の 2 つの方法があります。たとえば、次のような要素宣言について考えてみます。

```
<!ELEMENT B (#PCDATA | F)*>
```

次のコンストラクタが提供されます。

コンストラクタ	説明
B(Document *doc);	子を含まない空の要素を作成します。
B(Document *doc, String data);	PCDATA を使用して要素を初期化します。
(Document *doc, F *theF);	要素 F の単一の子ノードを使用して要素を初期化します。

PCDATA を含む B のような要素にも、次のメソッドを使用して作成後にデータを追加できます。

```
void addData( Document *doc, String data);
```

次の使用方法は同じ結果を戻します。

```
b = new B("data");
```

および

```
b = new B();
b->addData("data");
```

同様に、次の使用方法も同じ結果を戻します。

```
f=newF(...);
b=new B(f);
```

および

```
f=newF(...);
b=newE();
b->addNode(F);
```

コンストラクタを作成する場合、修飾子「?」（オプション）、「*」（0（ゼロ）以上）および「+」（1 以上）は無視されます。たとえば、次の要素について考えてみます。

```
<!ELEMENT Sample (A* | (B, (C? | (D, E)*)) | F)+>
```

修飾子が存在しない場合と同様に、次のコンストラクタが作成されます。

```
Sample(Document *doc);
Sample(Document *doc, A *theA);
Sample(Document *doc, B *theB, C *theC);
Sample(Document *doc, B *theB, D *theD, E *theE);
Sample(Document *doc, F *theF);
```

初期の子を受け入れるフォームの 1 つを使用して、必要な要素を作成できない場合、最初に空の要素を宣言し、必要に応じて addNode() を使用してノードを追加する必要があります。

要素の各属性には、属性の値を設定するためのメソッド setattrname() が提供されます。たとえば、次の要素宣言について考えてみます。

```
<!ELEMENT D (#PCDATA)>
<!ATTLIST D foo CDATA #REQUIRED>
```

クラス D には次のメソッドが提供されます。

```
Attr* setfoo(String value);
```

注意：作成された要素は、作成中に妥当性がテストされません。ユーザーは、最終要素に対して、XML パーサーの validate() メソッドを明示的にコールする必要があります。

generated のメソッド

表 19-2 generated（DTD 用）のメソッドの概要

メソッド	説明
class()（19-7 ページ）	コンストラクタです。
addData()（19-8 ページ）	要素に PCDATA を追加します。
addNode()（19-8 ページ）	要素にノードを追加します。
setattribute()（19-9 ページ）	要素の属性の 1 つを設定します。

class()

説明

クラス・コンストラクタです。指定されたドキュメントに属する要素を作成します。この項の冒頭に示した例を参照してください。次の表に、オプションを示します。

構文	説明
class(Document *doc);	子を含まない要素を作成します（必要に応じて `addData()` および `addNode()` を使用して、要素にデータおよびノードを追加する必要があります）。
class(Document *doc, ...);	要素の定義に応じて、初期のデータまたは子を提供するために使用されます。

パラメータ	説明
doc	要素が属するドキュメント。
...	要素の定義によって変わる引数。

addData()

説明

要素に、指定された値を含む PCDATA サブノードを追加することによって、データを追加します。addData を複数回コールした場合、ノードには複数の PCDATA サブノードが追加されます。構成終了時に、XMLParser::normalize() を使用してこれを正規化する必要があります。

構文

```
void addData( Document *doc,
              String data);
```

パラメータ	説明
doc	要素が属するドキュメント。
data	追加するデータ。

addNode()

説明

要素に子ノードを追加します。この時点では、結果の要素の構造を検証する必要はありません。要素を適切に構成するのはユーザーが行ってください。適切に構成されたかどうかは、XMLParser::validate() で検証できます。

プロトタイプ

```
void addNode( node thenode);
```

パラメータ	説明
thenode	追加するノード。

setattribute()

説明

指定された値を使用して、要素の属性を設定します。各属性には、`setattribute()` という名前の 1 つのメソッドが提供されます。作成された属性を戻します。

プロトタイプ

```
Attr* setattribute( String value);
```

パラメータ	説明
value	属性値。

generated クラス（スキーマ用）

generated の説明

すべての generated クラスは、生成されたファイルと同じ名前を持つ C++ 名前空間で囲まれます。たとえば、Foo.cpp には、名前空間 Foo が含まれます。

generated クラスは、スキーマ内の要素ごとに作成されます。親要素の名前が接頭辞となるローカル要素を除き、クラスは要素と同じ名前を持ちます。

要素（または属性）の名前は、C++ 識別子として直接使用できない場合、その名前をコンパイラのキャラクタ・セット（ASCII または EBCDIC）に変換してから、マップできない文字をそれらのコード・ポイントの 2 文字の 16 進数に置き換えることによって、有効な識別子にマップされます。たとえば、要素名「Curçao」は、「Curacao」にマップします。再マップされた名前がすでに使用されている場合は、名前が一意になるように、その末尾に数字が追加されます（たとえば「Curacao0」）。generated クラスによって作成された要素および属性は元の名前を持つことに注意してください。再マッピングは、C++ での構文上正しくなるように、生成されたコードにのみ適用されます。再マッピングは、これらの要素によって作成された XML 要素およびデータには適用されません。

要素を作成するには、空の要素の作成後に子を 1 つずつ追加する方法、および初期データまたは子を使用して要素を作成する方法の 2 つの方法があります。たとえば、次の要素宣言があるとします。

```
<element name="foo">
   <complexType content="mixed">
      <element ref="thing" minOccurs="0"/>
      <attribute name="bar" use="required" type="int"/>
   </complexType>
</element>
```

次のコンストラクタが提供されます。

```
foo(Document *doc);
     // Makes an empty element with no children

foo(Document *doc, DOMString s);
     // Initializes it with PCDATA

foo(Document *doc, Que::thing *the_thing);
     // Initialized with a single child node of element thing
```

PCDATA を含む foo のような要素にも、作成後にデータを追加するためのメソッドが提供されます。

```
void foo::addData(Document *doc, DQMString s);
```

foo の各子要素にも、「アセンブラ」が提供されます。

```
void foo::addNode(Queue::thing *the_thing);
```

次の使用方法は同じ結果を戻します。

```
f = new Queue::foo("data");
```

および

```
f = new Queue::foo();
f->addData("data");
```

同様に、次の使用方法も同じ結果を戻します。

```
f = new Queue::foo(...);
t = new Queue::thing(...);
```

および

```
f = new Queue::foo(...);
t = new Queue::thing(...);
f->addNode(t);
```

初期要素で可能なすべての組合せにコンストラクタが提供されているわけではありません（特に、出現回数が変化する場合）。適切なコンストラクタが存在しない場合は、要素を作成する必要があります。たとえば、次の要素定義について考えてみます。

```
<element name="map-data">
   <complexType content="mixed">
      <element ref="aq:item" minOccurs="0" maxOccurs="*"/>
   </complexType>
</element>
```

map-data 要素には、任意の数の aq:item の子が含まれます。この場合、出現を 1 回のみ許可するコンストラクタが提供されます。複数回出現する場合は、次の例（4 回出現）のように作成する必要があります。

```
md = new Queue::map~data(doc, i1);
md->addNode(i2);
md->addNode(i3);
md->addNode(i4);
```

要素の各属性には、属性の値を設定するためのメソッド set_attrname() が提供されます。たとえば、次の要素宣言について考えてみます。

```
<element name="client-operation">
   <complexType content="mixed">
      <element name="txid" type="string" minOccurs="0"/>
      <attribute name="opcode" use="required" type="aq:opcode_type"/>
   </complexType>
```

```
</element>
```

属性を設定するメソッドが提供されます。

```
Attr* client_operation::set_opcode(DOMString s);
```

注意：作成された要素は、作成中に妥当性がテストされません。ユーザーは、最終要素に対して、XML パーサーの validate メソッドを明示的にコールする必要があります。

generated のメソッド

表 19-3 generated（スキーマ用）のメソッドの概要

メソッド	説明
class()（19-12 ページ）	クラス・コンストラクタです。
addData()（19-13 ページ）	要素に PCDATA を追加します。
addNode()（19-13 ページ）	要素にノードを追加します。
set_attribute()（19-14 ページ）	要素の属性の 1 つを設定します。

class()

説明

指定されたドキュメントに属する要素を作成します。この項の冒頭に示した例を参照してください。次の表に、オプションを示します。

構文	説明
class(Document *doc);	子を含まない要素を作成します（必要に応じて `addData()` および `addNode()` を使用して、要素にデータおよびノードを追加する必要があります）。
class(Document *doc, ...);	要素の定義に応じて、初期のデータまたは子を提供するために使用されます。

パラメータ	説明
doc	要素が属するドキュメント。
...	要素の定義によって変わる引数。

addData()

説明

要素に、指定された値を含む PCDATA サブノードを追加することによって、データを追加します。addData を複数回コールした場合、ノードには複数の PCDATA サブノードが追加されます。構成終了時に、`XMLParser::normalize()` を使用してこれを正規化する必要があります。

構文

```
void addData( Document *doc,
                  String data);
```

パラメータ	説明
doc	要素が属するドキュメント。
data	追加するデータ。

addNode()

説明

要素に子ノードを追加します。この時点では、結果の要素の構造を検証する必要はありません。要素を適切に構成するのはユーザーが行ってください。適切に構成されたかどうかは、`XMLParser::validate()` で検証できます。

構文

```
void addNode( node the_node);
```

パラメータ	説明
the_node	追加するノード。

set_attribute()

説明

指定された値を使用して、要素の属性を設定します。作成された属性を戻します。各属性には、set_attribute() という名前の 1 つのメソッドが提供されます。

構文

```
Attr* set_attribute( String value);
```

第 IV 部

XDK for PL/SQL

第 IV 部に含まれる章は、次のとおりです。

20

XML SQL Utility（XSU）for PL/SQL

XML SQL Utility（XSU）for PL/SQL は、次の 2 つのパッケージで構成されます。

- DBMS_XMLQuery パッケージ
- DBMS_XMLSave パッケージ

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

DBMS_XMLQuery パッケージ

DBMS_XMLQuery の説明

この API は、DB_to_XML 型の機能を提供します。

DBMS_XMLQuery の型

表 20-1 DBMS_XMLQuery の型

型	説明
ctxType	問合せのコンテキスト・ハンドルの型です。newContext() の戻り型です。

DBMS_XMLQuery の定数

表 20-2 DBMS_XMLQuery の定数

定数	説明
DB_ENCODING	DB キャラクタ・エンコーディングが使用されるシグナルに使用されます。
DEFAULT_ROWSETTAG	結果セットから生成される XML を囲む要素のタグ名（ほとんどの場合、ルート・ノードのタグ名）で、ROWSET と表されます。
DEFAULT_ERRORTAG	発生したエラーを囲むデフォルトのタグで、ERROR と表されます。
DEFAULT_ROWIDATTR	データベース・レコードに対応する XML 要素のカーディナリティ属性のデフォルト名で、NUM と表されます。
DEFAULT_ROWTAG	データベース・レコードに対応する要素のデフォルトのタグ名で、ROW と表されます。
DEFAULT_DATE_FORMAT	デフォルトの日付マスクで、MM/dd/yyyy HH:mm:ss と表されます。
ALL_ROWS	すべての行を出力する必要があることを示します。
NONE	出力に XML メタデータ（DTD またはスキーマ）を含める必要がないことを指定します。
DTD	DTD の生成を指定します。
SCHEMA	XML Schema を生成するかどうかを指定します。
LOWER_CASE	小文字のタグ名を使用します。
UPPER_CASE	大文字のタグ名を使用します。

DBMS_XMLQuery のファンクションおよびプロシージャ

表 20-3 DBMS_XMLQuery のファンクションおよびプロシージャの概要

ファンクションまたはプロシージャ	説明
newContext()（20-5 ページ）	問合せコンテキストを作成し、コンテキスト・ハンドルを戻します。
closeContext()（20-5 ページ）	特定の問合せコンテキストのクローズまたは割当て解除を行います。
setRowsetTag()（20-6 ページ）	XML データセットを囲むタグを設定します。
setRowTag()（20-6 ページ）	データベースに対応する XML 要素を囲むタグを設定します。
setErrorTag()（20-7 ページ）	XML のエラー・ドキュメントを囲むタグを設定します。
setRowIdAttrName()（20-7 ページ）	行の囲みタグの ID 属性名を設定します。
setRowIdAttrValue()（20-8 ページ）	列値が行の囲みタグの ID 属性に割り当てられるスカラー列を指定します。
setCollIdAttrName()（20-8 ページ）	コレクション要素の区切りタグの ID 属性名を設定します。
useNullAttributeIndicator()（20-9 ページ）	`null` であることを示すために、XML 属性を使用するかどうかを指定します。
useTypeForCollElemTag()（20-9 ページ）	コレクション要素の型名をコレクション要素のタグ名として使用するように XSU に指示します。
setTagCase()（20-10 ページ）	生成された XML タグの大 / 小文字を設定します。
setDateFormat()（20-10 ページ）	XML 文書に生成日付の書式を設定します。
setMaxRows()（20-11 ページ）	XML に変換する行の最大数を設定します。
setSkipRows()（20-11 ページ）	スキップする行数を設定します。
setStylesheetHeader()（20-12 ページ）	スタイルシート・ヘッダーを設定します。
setXSLT()（20-12 ページ）	生成された XML に適用するスタイルシートを登録します。
setXSLTParam()（20-13 ページ）	最上位のスタイルシートのパラメータ値を設定します。
removeXSLTParam()（20-14 ページ）	特定した最上位のスタイルシートのパラメータを削除します。
setBindValue()（20-14 ページ）	特定したバインド名の値を設定します。
setMetaHeader()（20-15 ページ）	XML メタ・ヘッダーを設定します。

表 20-3 DBMS_XMLQuery のファンクションおよびプロシージャの概要（続き）

ファンクションまたはプロシージャ	説明
setDataHeader()（20-15 ページ）	XML データ・ヘッダーを設定します。
setEncodingTag()（20-16 ページ）	XML 文書にエンコーディングする処理命令を設定します。
setRaiseException()（20-16 ページ）	呼び出された例外を発生させるように XSU に通知します。
setRaiseNoRowsException()（20-17 ページ）	生成された XML 文書が空の場合に、理由にかかわらず `OracleXMLNoRowsException` が発生するかどうかを XSU に通知します。
setSQLToXMLNameEscaping()（20-17 ページ）	XML 識別子にマップされる SQL オブジェクト名が有効な XML 識別子でない場合、XML タグのエスケープを行うかどうかを指定します。
propagateOriginalException()（20-18 ページ）	例外が発生した場合、例外を OracleXMLSQLException でラップするかわりに、呼び出された例外を発生させる必要があることを XSU に通知します。
getExceptionContent()（20-18 ページ）	発生した例外のエラー・コードおよびエラー・メッセージを戻します。
getDTD()（20-19 ページ）	DTD を生成します。
getNumRowsProcessed()（20-19 ページ）	問合せで処理された行の数を戻します。
getVersion()（20-20 ページ）	使用中の XSU のバージョンを出力します。
getXML()（20-20 ページ）	XML 文書を生成します。

newContext()

説明

問合せコンテキストを作成し、コンテキスト・ハンドルを戻します。次の表に、オプションを示します。

構文	説明
FUNCTION newContext(sqlQuery IN VARCHAR2) RETURN ctxType;	文字列から問合せコンテキストを作成します。
FUNCTION newContext(sqlQuery IN CLOB) RETURN ctxType;	CLOB から問合せコンテキストを作成します。

パラメータ	IN / OUT	説明
sqlQuery	(IN)	SQL 問合せ（XML への変換結果）。

closeContext()

説明

特定の問合せコンテキストのクローズまたは割当て解除を行います。

構文

```
PROCEDURE closeContext( ctxHdl IN ctxType);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。

setRowsetTag()

説明

XML データセットを囲むタグを設定します。

構文

```
PROCEDURE setRowsetTag(ctxHdl IN ctxType, tag IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
tag	(IN)	タグ名。

setRowTag()

説明

データベース・レコードに対応する XML 要素を囲むタグを設定します。

構文

```
PROCEDURE setRowTag(ctxHdl IN ctxType, tag IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
tag	(IN)	タグ名。

setErrorTag()

説明

XML のエラー・ドキュメントを囲むタグを設定します。

構文

```
PROCEDURE setErrorTag( ctxHdl IN ctxType,
                       tag IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
tag	(IN)	タグ名。

setRowIdAttrName()

説明

行の囲みタグの ID 属性名を設定します。このタグに、null または空の文字列を設定すると、ROWID 属性が省略されます。

構文

```
PROCEDURE setRowIdAttrName( ctxHdl IN ctxType,
                            attrName IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
attrName	(IN)	属性名。

setRowIdAttrValue()

説明

列値が行の囲みタグの ID 属性に割り当てられるスカラー列を指定します。colName に null または空の文字列を指定すると、ROWID 属性に行カウントの値（0、1、2 など）が割り当てられます。

構文

```
PROCEDURE setRowIdAttrValue( ctxHdl IN ctxType,
                             colName IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
colName	(IN)	ROWID 属性に割り当てられる値を持つ列。

setCollIdAttrName()

説明

コレクション要素の区切りタグの ID 属性名を設定します。

構文

```
PROCEDURE setCollIdAttrName( ctxHdl IN ctxType,
                             attrName IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
attrName	(IN)	属性名。

useNullAttributeIndicator()

説明

null であることを示すために、XML 属性を使用するか、XML 文書内の特定のエンティティを省略するかを指定します。

構文

```
PROCEDURE useNullAttributeIndicator( ctxHdl IN ctxType,
                                     flag IN BOOLEAN := true);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	null を示すために属性を使用するかどうかを指定します。

useTypeForCollElemTag()

説明

デフォルトでは、コレクション要素のタグ名は、コレクションのタグ名の後に「_item」が続く名前になります。引数に true を指定してこのメソッドをコールすると、XSU はコレクション要素の型名をコレクション要素のタグ名として使用します。

構文

```
PROCEDURE useTypeForCollElemTag( ctxHdl IN ctxType,
                                 flag IN BOOLEAN := true);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	型名を使用するかどうかを指定します。

setTagCase()

説明

生成された XML タグの大 / 小文字を設定します。

構文

```
PROCEDURE setTagCase( ctxHdl IN ctxType,
                      tCase IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
tCase	(IN)	タグの大 / 小文字（0= 混在、1= 小文字、2= 大文字)。

setDateFormat()

説明

XML 文書に生成日付の書式を設定します。日付書式パターン（日付マスク）の構文は、java.text.SimpleDateFormat クラスの要件を満たす必要があります。マスクを null または空の文字列に設定すると、デフォルトのマスク DEFAULT_DATE_FORMAT が使用されます。

構文

```
PROCEDURE setDateFormat( ctxHdl IN ctxType,
                         mask IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
mask	(IN)	日付マスク。

setMaxRows()

説明

XML に変換する行の最大数を設定します。デフォルトでは、最大数は設定されていません。

構文

```
PROCEDURE setMaxRows ( ctxHdl IN ctxType,
                       rows IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
rows	(IN)	生成する行の最大数。

setSkipRows()

説明

スキップする行数を設定します。デフォルトでは、0（ゼロ）行がスキップされます。

構文

```
PROCEDURE setSkipRows( ctxHdl IN ctxType,
                       rows IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
rows	(IN)	スキップする行の最大数。

setStylesheetHeader()

説明

生成された XML 文書にスタイルシート・ヘッダー（スタイルシート処理命令）を設定します。uri 引数に null を指定すると、スタイルシート・ヘッダーおよびスタイルシートの型の設定が解除されます。

構文

```
PROCEDURE setStylesheetHeader( ctxHdl IN ctxType,
                               uri IN VARCHAR2,
                               type IN VARCHAR2 := 'text/xsl');
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
uri	(IN)	スタイルシートの URI。
type	(IN)	スタイルシートの型（デフォルトは「text/xsl」）。

setXSLT()

説明

生成された XML に適用するスタイルシートを登録します。スタイルシートがすでに登録されている場合は、新しいスタイルシートに置き換えられます。次の表に、オプションを示します。

構文	説明
PROCEDURE setXSLT(ctxHdl IN ctxType, uri IN VARCHAR2, ref IN VARCHAR2 := null);	スタイルシートの登録を解除するには、uri に null を指定します。
PROCEDURE setXSLT(ctxHdl IN ctxType, stylesheet CLOB, ref IN VARCHAR2 := null);	スタイルシートの登録を解除するには、stylesheet に null または空の文字列を指定します。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
uri	(IN)	スタイルシート URI。
stylesheet	(IN)	スタイルシート。
ref	(IN)	インクルード、インポートおよび外部エンティティの URL。

setXSLTParam()

説明

最上位のスタイルシートのパラメータ値を設定します。パラメータ値は、有効な XPath 式が想定されます（したがって、文字列リテラル値を明示的に囲む必要があります)。スタイルシートが登録されてない場合、このメソッドは動作しません。

構文

```
PROCEDURE setXSLTParam( ctxHdl IN ctxType,
                        name IN VARCHAR2,
                        value IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
name	(IN)	最上位のスタイルシート・パラメータ名。
value	(IN)	スタイルシート・パラメータに割り当てられる値。

removeXSLTParam()

説明

最上位のスタイルシートのパラメータ値を削除します。スタイルシートが登録されてない場合、このメソッドは動作しません。

構文

```
PROCEDURE removeXSLTParam( ctxHdl IN ctxType,
                           name IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
name	(IN)	最上位のスタイルシート・パラメータ名。

setBindValue()

説明

特定したバインド名の値を設定します。

構文

```
PROCEDURE setBindValue( ctxHdl IN ctxType,
                        bindName IN VARCHAR2,
                        bindValue IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
bindName	(IN)	バインド名。
bindValue	(IN)	バインド値。

setMetaHeader()

説明

XML メタ・ヘッダーを設定します。ヘッダーが設定されると、このオブジェクトが生成した各 XML 文書のメタデータ部（DTD または XML Schema）の先頭に挿入されます。最後に指定したメタ・ヘッダーが使用されることに注意してください。また、ヘッダー・パラメータに null を指定すると、メタ・ヘッダーの設定が解除されます。

構文

```
PROCEDURE setMetaHeader( ctxHdl IN ctxType,
                         header IN CLOB := null);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
header	(IN)	ヘッダー。

setDataHeader()

説明

XML データ・ヘッダーを設定します。このデータ・ヘッダーは、XML のエンティティであり、問合せによって生成された XML エンティティ（行セット）の先頭に追加されます。2 つのエンティティは、docTag 引数で指定されたタグで囲まれます。最後に指定したデータ・ヘッダーが使用されることに注意してください。また、ヘッダーに null を指定すると、パラメータによってデータ・ヘッダーの設定が解除されます。

構文

```
PROCEDURE setDataHeader( ctxHdl IN ctxType,
                    header IN CLOB := null,
                    tag IN VARCHAR2 := null);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
header	(IN)	ヘッダー。
tag	(IN)	データ・ヘッダーおよび行セットを囲むタグ。

setEncodingTag()

説明

XML 文書にエンコーディングする処理命令を設定します。

構文

```
PROCEDURE setEncodingTag( ctxHdl IN ctxType,
                          enc IN VARCHAR2 := DB_ENCODING);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
enc	(IN)	使用するエンコーディング。

setRaiseException()

説明

呼び出された例外を発生させるように XSU に通知します。このコールが行われなかった場合、または flag 引数を `false` に指定した場合、XSU は SQL 例外をキャッチし、その例外メッセージから XML 文書を生成します。

構文

```
PROCEDURE setRaiseException( ctxHdl IN ctxType,
                             flag IN BOOLEAN := true);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	呼び出された例外を発生させるかどうかを指定します（発生させる場合は `true`、発生させない場合は `false`）。

setRaiseNoRowsException()

説明

生成された XML 文書が空の場合に、理由にかかわらず OracleXMLNoRowsException を発生させるかどうかを XSU に通知します。デフォルトでは、例外は発生しません。

構文

```
PROCEDURE setRaiseNoRowsException( ctxHdl IN ctxType,
                                   flag IN BOOLEAN := true);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	データが見つからなかった場合に、OracleXMLNoRowsException を発生させるかどうかを指定します（発生させる場合は `true`、発生させない場合は `false`）。

setSQLToXMLNameEscaping()

説明

XML 識別子にマップされる SQL オブジェクト名が有効な XML 識別子でない場合、XML タグのエスケープを行うかどうかを指定します。

構文

```
PROCEDURE setSQLToXMLNameEscaping( ctxHdl IN ctxType,
                                   flag IN BOOLEAN := true);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	エスケープを行うかどうかを指定します（行う場合は `true`、行わない場合は `false`）。

propagateOriginalException()

説明

例外が発生した場合、例外を OracleXMLSQLException でラップするかわりに、呼び出された例外を発生させる必要があることを XSU に通知します。

構文

```
PROCEDURE propagateOriginalException( ctxHdl IN ctxType,
                                      flag IN BOOLEAN);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	元の例外を伝播するかどうかを指定します（伝播する場合は `true`、伝播しない場合は `false`）。

getExceptionContent()

説明

発生した例外のエラー・コードおよびエラー・メッセージ（SQL エラー・コード）を引数を介して戻します。これによって、例外が呼び出されるたびに Oracle JVM が例外を発生させることを防止できます。そのため、レンダリングしている PL/SQL が元の例外にアクセスできません。

構文

```
PROCEDURE getExceptionContent( ctxHdl IN ctxType,
                               errNo OUT NUMBER,
                               errMsg OUT VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
errNo	(IN)	エラー番号。
errMsg	(IN)	エラー・メッセージ。

getDTD()

説明

コンテキストの初期化に使用される SQL 問合せに基づいて、DTD を生成して戻します。次の表に、オプションを示します。

構文	説明
FUNCTION getDTD(ctxHdl IN ctxType, withVer IN BOOLEAN := false) RETURN CLOB;	コンテキストの初期化に使用される SQL 問合せに基づいて、DTD を生成して戻すファンクション。
PROCEDURE getDTD(ctxHdl IN ctxType, xDoc IN CLOB, withVer IN BOOLEAN := false);	コンテキストおよび CLOB の xDOC の初期化に使用される SQL 問合せに基づいて、DTD を生成して戻すプロシージャ。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
withVer	(IN)	バージョン情報を生成するかどうかを指定します（生成する場合は `true`、生成しない場合は `false`）。
xDoc	(IN)	生成された XML 文書を書き込む CLOB。

getNumRowsProcessed()

説明

問合せで処理された行の数を戻します。

構文

```
FUNCTION getNumRowsProcessed( ctxHdl IN ctxType) RETURN NUMBER;
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。

getVersion()

説明

使用中の XSU のバージョンを出力します。

構文

```
PROCEDURE getVersion;
```

getXML()

説明

新しいコンテキストを作成し、問合せを実行し、XML を戻してコンテキストをクローズします。これは有効なファンクションです。コンテキストを明示的にオープンまたはクローズする必要がありません。次の表に、オプションを示します。

構文	説明
FUNCTION getXML(sqlQuery IN VARCHAR2, metaType IN NUMBER := NONE) RETURN CLOB;	SQL 問合せを文字列形式で使用するファンクション。
FUNCTION getXML(sqlQuery IN CLOB, metaType IN NUMBER := NONE) RETURN CLOB;	SQL 問合せを CLOB 形式で使用するファンクション。
FUNCTION getXML(ctxHdl IN ctxType, metaType IN NUMBER := NONE); RETURN CLOB	コンテキストの初期化に使用される SQL 問合せに基づいて、XML 文書を生成するファンクション。
PROCEDURE getXML(ctxHdl IN ctxType, xDoc IN CLOB, metaType IN NUMBER := NONE);	コンテキストの初期化に使用される SQL 問合せに基づいて、XML 文書を生成するプロシージャ。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
sqlQuery	(IN)	SQL 問合せ。
metaType	(IN)	XML メタデータの型（NONE、DTD または SCHEMA）。
xDoc	(IN)	生成された XML 文書を書き込む CLOB。

DBMS_XMLSave パッケージ

DBMS_XMLSave の説明

この API は、XML_to_DB 型の機能を提供します。

DBMS_XMLSave の型

表 20-4 DBMS_XMLSave の型

型	説明
ctxType	問合せのコンテキスト・ハンドルの型です。newContext() の戻り型です。

DBMS_XMLSave の定数

表 20-5 DBMS_XMLSave の定数

定数	説明
DEFAULT_ROWTAG	データベース・レコードに対応する要素のデフォルトのタグ名で、ROW と表されます。
DEFAULT_DATE_FORMAT	デフォルトの日付マスクで、MM/dd/yyyy HH:mm:ss と表されます。
MATCH_CASE	XML 要素をデータベースのエンティティへマップする場合、XSU が大 / 小文字を区別するように指定します。
IGNORE_CASE	XML 要素をデータベースのエンティティへマップする場合、XSU が大 / 小文字を区別しないように指定します。

DBMS_XMLSave のファンクションおよびプロシージャ

表 20-6 DBMS_XMLSave のファンクションおよびプロシージャの概要

ファンクションまたはプロシージャ	説明
newContext()（20-24 ページ）	保存コンテキストを作成し、コンテキスト・ハンドルを戻します。
closeContext()（20-24 ページ）	特定の保存コンテキストのクローズまたは割当て解除を行います。
setRowTag()（20-25 ページ）	XML 文書で使用するタグを指定し、データベースに対応する XML 要素を囲みます。

表 20-6 DBMS_XMLSave のファンクションおよびプロシージャの概要

ファンクションまたはプロシージャ	説明
setIgnoreCase()（20-25 ページ）	XSU は、XML 要素をデータベースへマップします。
setDateFormat()（20-26 ページ）	XML 文書の日付書式を、XSU に通知します。
setBatchSize()（20-26 ページ）	DML 操作中に使用されるバッチ・サイズを変更します。
setCommitBatch()（20-27 ページ）	コミットのバッチ・サイズを設定します。
setSQLToXMLNameEscaping()（20-27 ページ）	XML 識別子にマップされる SQL オブジェクト名が有効な XML 識別子でない場合、XML タグのエスケープを行うかどうかを指定します。
setUpdateColumn()（20-28 ページ）	更新列リストに列を追加します。
clearUpdateColumnList()（20-28 ページ）	更新列のリストを消去します。
setPreserveWhitespace()（20-29 ページ）	空白を保持するかどうかを XSU に通知します。
setKeyColumn()（20-29 ページ）	キー列リストに列を追加します。
clearKeyColumnList()（20-30 ページ）	キー列リストを消去します。
setXSLT()（20-30 ページ）	保存する XML に適用する XSL 変換を登録します。
setXSLTParam()（20-31 ページ）	最上位のスタイルシートのパラメータ値を設定します。
removeXSLTParam()（20-31 ページ）	最上位のスタイルシートのパラメータ値を削除します。
insertXML()（20-32 ページ）	コンテキストの作成時に指定された表に、XML 文書を挿入します。
updateXML()（20-33 ページ）	XML 文書を指定して、表を更新します。
deleteXML()（20-34 ページ）	コンテキストの作成時に指定した表から、XML 文書のデータで指定したレコードを削除します。
propagateOriginalException()（20-34 ページ）	例外が発生した場合、例外を OracleXMLSQLException でラップするかわりに、呼び出された例外を発生させる必要があることを XSU に通知します。
getExceptionContent()（20-35 ページ）	発生した例外のエラー・コードおよびエラー・メッセージを、引数を介して戻します。

newContext()

説明

保存コンテキストを作成し、コンテキスト・ハンドルを戻します。

構文

```
FUNCTION newContext(targetTable IN VARCHAR2) RETURN ctxType;
```

パラメータ	IN / OUT	説明
targetTable	(IN)	XML 文書をロードするターゲット表。

closeContext()

説明

特定の保存コンテキストのクローズまたは割当て解除を行います。

構文

```
PROCEDURE closeContext(ctxHdl IN ctxType);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。

setRowTag()

説明

XML 文書で使用するタグを指定し、データベース・レコードに対応する XML 要素を囲みます。

構文

```
PROCEDURE setRowTag( ctxHdl IN ctxType,
                     tag IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
tag	(IN)	タグ名。

setIgnoreCase()

説明

XSU は、要素名（XML タグ）に基づいて XML 要素をデータベースの列または属性にマップします。このファンクションは、大 / 小文字を区別しないように XSU に通知します。

構文

```
PROCEDURE setIgnoreCase( ctxHdl IN ctxType,
                         flag IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	XML 文書内のタグで大 / 小文字を無視するかどうかを指定します（0=false、1=true）。

setDateFormat()

説明

XML 文書の日付書式を、XSU に通知します。日付書式パターン（日付マスク）の構文は、java.text.SimpleDateFormat クラスの要件を満たす必要があります。マスクを null または空の文字列に設定すると、デフォルトのマスク OracleXMLCore.DATE_FORMAT が使用されます。

構文

```
PROCEDURE setDateFormat( ctxHdl IN ctxType,
                         mask IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
mask	(IN)	日付マスク。

setBatchSize()

説明

DML 操作中に使用されるバッチ・サイズを変更します。挿入、更新および削除を実行する場合、別々の文ではなく、1 回の処理でそれらの操作が実行されるように、バッチ処理を行う方が適しています。欠点は、その操作を実行する前に、すべてのバインド値を保持するためのメモリーがさらに多く必要になることです。バッチ処理を行う場合、バッチ処理の実行後にのみコミットが発生することに注意してください。そのため、バッチ内の 1 文が正常に実行されなかった場合、バッチ処理全体がロールバックされます。バッチ処理によるパフォーマンスの向上を考えると、大きな問題ではありませんが、これを防止するには、バッチ・サイズを 1 に設定します。

構文

```
PROCEDURE setBatchSize( ctxHdl IN ctxType,
                        batchSize IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
batchSize	(IN)	バッチ・サイズ。

setCommitBatch()

説明

コミットのバッチ・サイズを設定します。コミットのバッチ・サイズとは、実行後にコミットする必要がある挿入の回数または記録を意味します。commitBatch が 1 未満またはセッションが自動コミット・モードの場合、XSU は明示的なコミットを行わないことに注意してください。コミットのデフォルトのバッチ・サイズは 0（ゼロ）です。

構文

```
PROCEDURE setCommitBatch( ctxHdl IN ctxType,
                          batchSize IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
batchSize	(IN)	コミットのバッチ・サイズ。

setSQLToXMLNameEscaping()

説明

XML 識別子にマップされる SQL オブジェクト名が有効な XML 識別子でない場合、XML タグのエスケープを行うかどうかを指定します。

構文

```
PROCEDURE setSQLToXMLNameEscaping( ctxHdl IN ctxType,
                                   flag IN BOOLEAN := true);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	エスケープを行うかどうかを指定します。

setUpdateColumn()

説明

更新列リストに列を追加します。挿入の場合、デフォルトでは、表のすべての列に値が挿入されます。更新の場合、デフォルトでは、XML 文書の ROW 要素にあるタグに対応する列のみが更新されます。更新列リストを指定すると、このリストを構成する列のみが更新または挿入されます。

構文

```
PROCEDURE setUpdateColumn( ctxHdl IN ctxType,
                           colName IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
colName	(IN)	更新列リストに追加される列。

clearUpdateColumnList()

説明

更新列のリストを消去します。

構文

```
PROCEDURE clearUpdateColumnList( ctxHdl IN ctxType);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。

setPreserveWhitespace()

説明

空白を保持するかどうかを XSU に通知します。

構文

```
PROCEDURE setPreserveWhitespace( ctxHdl IN ctxType,
                                 flag IN BOOLEAN := true);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	XSU に空白を保持させるかどうかを指定します。

setKeyColumn()

説明

キー列リストに列を追加します。更新または削除の場合、UPDATE 文または DELETE 文の WHERE 句を構成するキー列リストの列です。キー列リストは更新される前に指定する必要がありますが、削除操作ではオプションです。

構文

```
PROCEDURE setKeyColumn( ctxHdl IN ctxType,
                        colName IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
colName	(IN)	キー列リストに追加される列。

clearKeyColumnList()

説明

キー列リストを消去します。

構文

```
PROCEDURE clearKeyColumnList( ctxHdl IN ctxType);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。

setXSLT()

説明

保存する XML に適用する XSL 変換を登録します。スタイルシートがすでに登録されている場合は、新しいスタイルシートに置き換えられます。スタイルシートの登録を解除するには、URI に `null` を指定します。次の表に、オプションを示します。

構文	説明
PROCEDURE setXSLT(ctxHdl IN ctxType, uri IN VARCHAR2, ref IN VARCHAR2 := null);	URI を介してスタイルシートを渡します。
PROCEDURE setXSLT(ctxHdl IN ctxType, stylesheet IN CLOB, ref IN VARCHAR2 := null);	CLOB を介してスタイルシートを渡します。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
uri	(IN)	登録するスタイルシートの URI。
ref	(IN)	挿入、インポートおよび外部エンティティの URL。
stylesheet	(IN)	登録するスタイルシートを含む CLOB。

setXSLTParam()

説明

最上位のスタイルシートのパラメータ値を設定します。パラメータは、有効な XPath 式である必要があります（このため、文字列リテラル値を明示的に囲む必要があります）。

構文

```
PROCEDURE setXSLTParam( ctxHdl IN ctxType,
                        name IN VARCHAR2,
                        value IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
name	(IN)	パラメータ名。
value	(IN)	XPath 式としてのパラメータ値。

removeXSLTParam()

説明

最上位のスタイルシートのパラメータ値を削除します。

構文

```
PROCEDURE removeXSLTParam( ctxHdl IN ctxType,
                           name IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
name	(IN)	パラメータ名。

insertXML()

説明

コンテキストの作成時に指定した表に XML 文書を挿入し、挿入した行の数を戻します。次の表に、オプションを示します。

構文	説明
FUNCTION insertXML(ctxHdl IN ctxType, xDoc IN VARCHAR2) RETURN NUMBER;	xDoc パラメータを、VARCHAR2 として渡します。
FUNCTION insertXML(ctxHdl IN ctxType, xDoc IN CLOB) RETURN NUMBER;	xDoc パラメータを、CLOB として渡します。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
xDoc	(IN)	XML 文書を含む文字列。

updateXML()

説明

XML 文書のデータを使用して、コンテキストの作成時に指定した表を更新し、更新した行の数を戻します。次の表に、オプションを示します。

構文	説明
FUNCTION updateXML(ctxHdl IN ctxType, xDoc IN VARCHAR2) RETURN NUMBER;	xDoc パラメータを、VARCHAR2 として渡します。
FUNCTION updateXML(ctxHdl IN ctxType, xDoc IN CLOB) RETURN NUMBER;	xDoc パラメータを、CLOB として渡します。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
xDoc	(IN)	XML 文書を含む文字列。

deleteXML()

説明

コンテキストの作成時に指定された表から、XML 文書のデータで指定したレコードを削除して、削除した行の数を戻します。次の表に、オプションを示します。

構文	説明
FUNCTION deleteXML(ctxHdl IN ctxPType, xDoc IN VARCHAR2) RETURN NUMBER;	xDoc パラメータに VARCHAR2 型を使用します。
FUNCTION deleteXML(ctxHdl IN ctxType, xDoc IN CLOB) RETURN NUMBER;	xDoc パラメータに CLOB 型を使用します。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
xDoc	(IN)	XML 文書を含む文字列。

propagateOriginalException()

説明

例外が発生した場合、例外を OracleXMLSQLException でラップするかわりに、呼び出された例外を発生させる必要があることを XSU に通知します。

構文

```
PROCEDURE propagateOriginalException( ctxHdl IN ctxType,
                                      flag IN BOOLEAN);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	元の例外を伝播するかどうかを指定します（0=false、1=true）。

getExceptionContent()

説明

発生した例外のエラー・コードおよびエラー・メッセージ（SQL エラー・コード）を引数を介して戻します。これによって、例外が呼び出されるたびに Oracle JVM が例外を発生させることを防止できます。そのため、PL/SQL をレンダリングすると、元の例外にアクセスできません。

構文

```
PROCEDURE getExceptionContent( ctxHdl IN ctxType,
                               errNo OUT NUMBER,
                               errMsg OUT VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
errNo	(IN)	エラー番号。
errMsg	(IN)	エラー・メッセージ。

第 V 部

XML データベース・サポート： Oracle XML DB for Java

第 V 部に含まれる章は、次のとおりです。

21

Java API for XMLType

この章では、oracle.xdb.dom パッケージに含まれ、Java DOM API for XMLType を実装するクラスについて説明します。Oracle XML DB の DOM API は、W3C の DOM 勧告の実装に加えて、Oracle 固有の拡張機能を提供します。この章の内容は次のとおりです。

表 21-1 に、各クラスおよびそのクラスが実装する W3C の DOM インタフェースを示します。

表 21-1 Java DOM API for XMLType のクラスおよび W3C のインタフェース

Java DOM API for XMLType のクラス	W3C の DOM インタフェース勧告
XDBAttribute クラス	org.w3c.dom.Attribute
XDBCData クラス	org.w3c.dom.CData
XDBCharData クラス	org.w3c.dom.CharData
XDBComment クラス	org.w3c.dom.Comment
XDBDocument クラス	org.w3c.dom.Document
XDBDomImplementation クラス	org.w3c.dom.DOMImplementation
XDBElement クラス	org.w3c.dom.Element
XDBEntity クラス	org.w3c.dom.Entity
XDBNamedNodeMap クラス	org.w3c.dom.NamedNodeMap
XDBNode クラス	org.w3c.dom.Node
XDBNodeList クラス	org.w3c.dom.NodeList
XDBNotation クラス	org.w3c.dom.Notation
XDBProcInst クラス	org.w3c.dom.ProcessingInstruction
XDBText クラス	org.w3c.dom.Text

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

XDBAttribute クラス

XDBAttribute の説明

このクラスは、W3C の DOM の Node インタフェース（XOB との対話用）である org.w3c.dom.Attribute を実装します。

XDBAttribute の構文

```
public class XDBAttribute

oracle.xdb.dom.XDBAttribute
```

XDBCData クラス

XDBCData の説明

このクラスは、W3C の Text インタフェースである org.w3c.dom.CData を実装します。

XDBCData の構文

```
public class XDBCData
```

oracle.xdb.dom.XDBCData

XDBCharData クラス

XDBCharData の説明

このクラスは、W3C の CharacterData インタフェースである org.w3c.dom.CharData を実装します。

XDBCharData の構文

```
public class XDBCharData
```

oracle.xdb.dom.XDBCharData

XDBComment クラス

XDBComment の説明

このクラスは、org.w3c.dom.Comment インタフェースを実装します。

XDBComment の構文

```
public class XDBComment
```

oracle.xdb.dom.XDBComment

XDBDocument クラス

XDBDocument の説明

このクラスは、org.w3c.dom.Document インタフェースを実装します。

XDBDocument の構文

```
public class XDBDocument

oracle.xdb.dom.XDBDocument
```

XDBDocument のメソッド

XDBDocument()

説明

クラス・コンストラクタです。次の表に、オプションを示します。

構文	説明
public XDBDocument();	新しいドキュメントを作成します。サーバーでのみ使用できます。
public XDBDocument(byte[] source);	ソースからドキュメントを移入します。サーバーでのみ使用できます。
public XDBDocument(Connection conn);	ドキュメント・ソースをキャッシュするために接続をオープンします。
public XDBDocument(Connection conn, byte[] source);	ドキュメント・ソースのバイトをキャッシュするために接続をオープンします。
public XDBDocument(Connection conn, String source);	XML テキストを含む文字列をキャッシュするために接続をオープンします。
public XDBDocument(String source);	XML テキストを含む文字列です。サーバーでのみ使用できます。

パラメータ	説明
source	XML テキストを含むソース。
conn	使用する接続。

XDBDomImplementation クラス

XDBDomImplementation の説明

このクラスは、org.w3c.dom.DomImplementation を実装します。

XDBDomImplementation の構文

```
public class XDBDomImplementation
```

oracle.xdb.dom.XDBDomImplementation

XDBDomImplementation のメソッド

XDBDomImplementation()

説明

サーバーへの JDBC 接続をオープンします。

構文

```
public  XDBDomImplementation();
```

XDBElement クラス

XDBElement の説明

このクラスは、org.w3c.dom.Element を実装します。

XDBElement の構文

```
public class XDBElement
```

oracle.xdb.dom.XDBElement

XDBEntity クラス

XDBEntity の説明

このクラスは、org.w3c.dom.Entity を実装します。

XDBEntity の構文

```
public class XDBEntity
```

oracle.xdb.dom.XDBEntity

XDBNamedNodeMap クラス

XDBNamedNodeMap の説明

このクラスは、org.w3c.dom.NamedNodeMap を実装します。

XDBNamedNodeMap の構文

```
public class XDBNamedNodeMap
```

oracle.xdb.dom.XDBNamedNodeMap

XDBNode クラス

XDBNode の構文

```
public abstract class XDBNode

oracle.xdb.dom.XDBNode
```

XDBNode の説明

このクラスは、W3C の DOM の Node インタフェース（XOB との対話用）である org.w3c.dom.Node を実装します。

XDBNode のメソッド

write()

説明

このノード（およびすべてのサブノード）の XML を出力ストリームに書き込みます。OutputStream が ServletOutputStream の場合、サーブレットの出力がコミットされ、データは、システム固有のストリーム・メカニズムによって書き込まれます。

構文

```
public void write( OutputStream s,
                   String charEncoding,
                   short indent);
```

パラメータ	説明
s	出力の書込み先のストリーム。XML テキストを含みます。
charEncoding	IANA 文字コード（「ISO 8859」など）。
indent	ネストした要素をインデントする文字数。

XDBNodeList クラス

XDBNodeList の説明

このクラスは、org.w3c.dom.NodeList を実装します。

XDBNodeList の構文

```
public class XDBNodeList
```

oracle.xdb.dom.XDBNodeList

XDBNotation クラス

XDBNotation の説明

このクラスは、org.w3c.dom.Notation を実装します。

XDBNotation の構文

```
public class XDBNotation
```

oracle.xdb.dom.XDBNotation

XDBProcInst クラス

XDBProcInst の説明

このクラスは、W3C の DOM の ProcessingInstruction インタフェースである `org.w3c.dom.ProcInst` を実装します。

XDBProcInst の構文

```
public class XDBProcInst
```

`oracle.xdb.dom.XDBProcInst`

XDBText クラス

XDBText の説明

このクラスは、org.w3c.dom.Text を実装します。

XDBText の構文

```
public class XDBText

oracle.xdb.dom.XDBText
```

XMLType クラス

XMLType の説明

このクラスは、SQL 型 XMLType 用の Java メソッドを実装します。

XMLType の構文

```
public class XMLType
```

oracle.xdb.XMLType

XMLType のメソッド

表 21-2 XMLType のメソッドの概要

メソッド	説明
createXML()（21-19 ページ）	XMLType を作成します。
getInputStream()（21-20 ページ）	XMLType データに対応する入力ストリームを戻します。
getStringVal()（21-20 ページ）	XMLType から XML データを含む文字列値を取得します。
getClobVal()（21-20 ページ）	XMLType から XML データを含む CLOB 値を取得します。
extract()（21-20 ページ）	XMLType から指定されたノード・セットを抽出します。
existsNode()（21-21 ページ）	XMLType 内に指定されたノード・セットが存在するかどうかを確認します。
transform()（21-21 ページ）	指定された XSL 文書を使用して、XMLType を変換します。
isFragment()（21-22 ページ）	XMLType が通常のドキュメントであるか、またはドキュメント・フラグメントであるかを確認します。
getDOM()（21-22 ページ）	XMLType に対応する DOM 文書を取得します。
writeToOutputStream()（21-22 ページ）	XMLType データを指定された出力ストリームに書き込みます。

createXML()

説明

XMLType を作成します。作成できない場合は、`java.sql.SQLException` が発生します。次の表に、オプションを示します。

構文	説明
public static XMLType createXML(OPAQUE opq);	XMLType バイトを含む不透明型を指定して、XMLType を作成して戻します。
public static XMLType createXML(Connection conn, String xmlval);	XML データを含む文字列を指定して、XMLType を作成して戻します。
public static XMLType createXML(Connection conn, CLOB xmlval);	XML データを含む CLOB を指定して、XMLType を作成して戻します。
public static XMLType createXML(Connection conn, Document domdoc);	DOM 文書のインスタンスを指定して、XMLType を作成して戻します。
public static XMLType createXML(Connection conn, InputStream is);	XML データに InputStream を指定して、XMLType を作成して戻します。

パラメータ	説明
opq	XMLType の作成元の不透明オブジェクト。
conn	使用する接続オブジェクト。
xmlval	XML データを含む値。
domdoc	DOM ツリーを表す DOM 文書。

getInputStream()

説明

XMLType ドキュメントに対応する入力ストリームを戻します。InputStream メソッドを使用して XML データを読み込むことができるようになります。

構文

```
public InputStream getInputStream();
```

getStringVal()

説明

XMLType から XML データを含む文字列値を取得します。java.sql.SQLException が発生します。

構文

```
public String getStringVal();
```

getClobVal()

説明

XMLType から XML データを含む CLOB 値を取得します。java.sql.SQLException が発生します。

構文

```
public CLOB getClobVal();
```

extract()

説明

XMLType から指定されたノード・セットを抽出して戻します。このノード・セットは、XPath 式によって指定されます。元の XMLType は変更されません。Thick の場合にのみ機能します。指定された式に一致するノードが存在しない場合は、null を戻します。java.sql.SQLException が発生します。

構文

```
public XMLType extract( String xpath,
                        String nsmap);
```

パラメータ	説明
xpath	検索するノードを指定する XPath 式。
nsmap	XPath 式の接頭辞を解決する名前空間のマップ（フォーマットは「xmlns=a.com xmlns:b=b.com」）。

existsNode()

説明

XMLType 内に指定されたノード・セットが存在するかどうかを確認します。このノード・セットは、XPath 式によって指定されます。指定されたノードが XMLType に存在する場合は `true`、存在しない場合は `false` を戻します。`java.sql.SQLException` が発生します。

構文

```
public boolean existsNode( String xpath,
                           String nsmap);
```

パラメータ	説明
xpath	検索するノードを指定する XPath 式。
nsmap	XPath 式の接頭辞を解決する名前空間のマップ（フォーマットは「xmlns=a.com xmlns:b=b.com」）。

transform()

説明

指定された XSL 文書を使用して、XMLType を変換して戻します。新しい（変換された）XML 文書を戻します。`java.sql.SQLException` が発生します。

構文

```
public XMLType transform( XMLType xsldoc,
                          String parammap);
```

パラメータ	説明
xsldoc	XMLType に適用する XSL 文書。

パラメータ	説明
parammap	XSL 変換に対して指定される最上位のパラメータ。「a=b c=d e=f」というフォーマットにする必要があります。null を指定できます。

isFragment()

説明

XMLType が通常のドキュメントであるか、またはドキュメント・フラグメントであるかを確認します。ドキュメントがフラグメントの場合は true、フラグメントでない場合は false を戻します。java.sql.SQLException が発生します。

構文

```
public boolean isFragment();
```

getDOM()

説明

XMLType に対応する DOM 文書を取得します。この文書は org.w3c.dom.Document です。コール元は、文書上ですべての DOM 操作を実行できます。この文書がバイナリ文書の場合、getDOM ファンクションは null を戻します。java.sql.SQLException が発生します。

構文

```
public org.w3c.dom.Document getDOM();
```

writeToOutputStream()

説明

XML データを指定された出力ストリームに書き込みます。

構文

```
public void writeToOutputStream( OutputStream os);
```

パラメータ	説明
os	データが書き込まれる出力ストリーム。

22

Oracle XML DB の JavaBeans API

Oracle XML DB の JavaBeans API は、Java、XML Schema および SQL のエンティティに対する JavaBeans クラスとデータ型のマッピング関係を実装します。

この章の内容は次のとおりです。

参照： 次のマニュアルまたは URL を参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』
- http://www.w3.org/XML/Schema

JavaBeans クラスおよびメソッドへの XML Schema のマッピング

表 22-1 に、JavaBeans クラスおよびそのメソッドへの XML Schema エンティティのマッピングを示します。

表 22-1 JavaBeans クラスおよびメソッドへの XML Schema エンティティのマッピング

XML Schema エンティティ	JavaBeans クラスまたはメソッド	説明
Attribute	Get/setAttribute{attrname}	ドキュメントの子要素の属性および指定された属性を取得または設定します。
Complextype	Get/set{complextype classname}.	複合型で定義された子要素の場合は、個別の Bean クラスを生成します。追加で処理命令またはコメントが生成します。処理命令およびコメントの DOM クラスを戻します。
Scalar data	Get/set{scalar type name}	maxOccurs > 1 の子を生成します。子の型のリストを取得または設定します。

XML Schema データ型、SQL データ型および JavaBeans データ型への JavaBeans API のマッピング

表 22-2 に、JavaBeans API によって使用される、XML Schema データ型、SQL データ型および Java データ型間のマッピングを示します。

表 22-2 XML Schema データ型、SQL データ型および Java データ型からのマッピング

XML Schema データ型	SQL データ型	JavaBeans データ型
Boolean	boolean	boolean
String	URI reference	ID
IDREF	ENTITY	NOTATION
Language	NCName	Name
java.lang.String	String	DECIMAL
INTEGER	LONG	SHORT
INT	POSITIVEINTEGER	NONPOSITIVEINTEGER
oracle.sql.Number	int	FLOAT
DOUBLE	oracle.sql.Number	float
TIMEDURATION	TIMEPERIOD	RECURRINGDURATION
DATE	TIME	MONTH,YEAR
RECURRINGDATE	java.sql.Timestamp	Time
REF	oracle.sql.Ref	Ref
BINARY	oracle.sql.RAW	Byte[]
QNAME	java.lang.String	String

23

Java/JNDI 用のリソース API

この章では、oracle.xdb.spi パッケージについて説明します。oracle.xdb.spi に含まれるクラスは、アプリケーションに Java Naming and Directory Interface（JNDI）への共通のアクセスを提供するサービス・プロバイダ・ドライバを実装します。

JNDI は、Sun 社のプログラミング・インタフェースです。このインタフェースは、Java プログラムを Domain Name System（DNS）、Lightweight Directory Access Protocol（LDAP）、Novell Directory Service（NDS）などのネーミング・サービスおよびディレクトリ・サービスに接続します。

アプリケーションは JNDI API に書き込まれます。ディレクトリ・ドライバは JNDI サービス・プロバイダ・インタフェース（SPI）に書き込まれます。oracle.xdb.spi のクラスは、主要な JNDI SPI インタフェースおよび Oracle XML DB の WebDAV サポートを実装します。

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

XDBContext クラス

XDBContext の説明

このクラスは、javax.naming.context を拡張する、Oracle XML DB 用の Java ネーミングおよびコンテキストのインタフェースを実装します。現在の実装はフェデレーションをサポートしないため、他の名前空間の存在が認識されません。

XDBContext の構文

```
public class XDBContext

oracle.xdb.spi.XDBContext
```

XDBContext のメソッド

XDBContext()

説明

XDBContext クラスのコンストラクタです。次の表に、オプションを示します。

構文	説明
public XDBContext(Hashtable env);	環境を指定して、XDBContext クラスのインスタンスを作成します。
public XDBContext(Hashtable env, String path);	環境およびパスを指定して、XDBContext クラスのインスタンスを作成します。

パラメータ	説明
env	コンテキストのプロパティを記述するための環境。
path	コンテキストの初期パス。

XDBContextFactory クラス

XDBContextFactory の説明

このクラスは、javax.naming.context を実装します。

XDBContextFactory の構文

```
public class XDBContextFactory
oracle.xdb.spi.XDBContextFactory
```

XDBContextFactory のメソッド

XDBContextFactory()

説明

XDBContextFactory クラスのコンストラクタです。

構文

```
public  XDBContextFactory();
```

XDBNameParser クラス

XDBNameParser の説明

このクラスは、javax.naming.NameParser を実装します。

XDBNameParser の構文

```
public class XDBNameParser
oracle.xdb.spi.XDBNameParser
```

XDBNamingEnumeration クラス

XDBNamingEnumeration の説明

このクラスは、javax.naming.NamingEnumeration を実装します。

XDBNamingEnumeration の構文

```
public class XDBNamingEnumeration
oracle.xdb.spi.XDBNamingEnumeration
```

XDBResource クラス

XDBResource の説明

このクラスは、Oracle XML DB の JNDI サービス・プロバイダ・インタフェース（SPI）の主要な機能を実装します。現在の実装はフェデレーションをサポートしないため、他の名前空間の存在が認識されません。

XDBResource の構文

```
public class XDBResource extends java.lang.Object

java.lang.Object
  |
  +--oracle.xdb.spi.XDBResource
```

XDBResource のメソッド

表 23-1 XDBResource のメソッドの概要

メソッド	説明
XDBResource()（23-7 ページ）	XDBResource の新しいインスタンスを作成します。
getAuthor()（23-7 ページ）	リソースの作成者を戻します。
getComment()（23-8 ページ）	リソースの DAV コメントを戻します。
getContent()（23-8 ページ）	リソースのコンテンツを戻します。
getContentType()（23-8 ページ）	リソースのコンテンツ・タイプを戻します。
getCreateDate()（23-8 ページ）	リソースの作成日を戻します。
getDisplayName()（23-9 ページ）	リソースの表示名を戻します。
getLanguage()（23-9 ページ）	リソースの言語を戻します。
getLastModDate()（23-9 ページ）	リソースの最終変更日を戻します。
getOwnerId()（23-9 ページ）	リソースの所有者 ID を戻します。
setACL()（23-10 ページ）	リソースに ACL を設定します。
setAuthor()（23-10 ページ）	リソースの作成者を設定します。
setComment()（23-10 ページ）	リソースの DAV コメントを設定します。
setContent()（23-11 ページ）	リソースのコンテンツを設定します。

表 23-1 XDBResource のメソッドの概要（続き）

メソッド	説明
setContentType()（23-11 ページ）	リソースのコンテンツ・タイプを設定します。
setCreateDate()（23-11 ページ）	リソースの作成日を設定します。
setDisplayName()（23-12 ページ）	リソースの表示名を設定します。
setLanguage()（23-12 ページ）	リソースの言語を設定します。
setLastModDate()（23-12 ページ）	リソースの最終変更日を設定します。
setOwnerId()（23-13 ページ）	リソースの所有者 ID を設定します。

XDBResource()

説明

XDBResource の新しいインスタンスを作成します。次の表に、オプションを示します。

構文	説明
public XDBResource(Hashtable env);	環境を指定して、XDBResource の新しいインスタンスを作成します。
public XDBResource(Hashtable env, String path);	環境およびパスを指定して、XDBResource の新しいインスタンスを作成します。

パラメータ	説明
env	指定された環境。
path	リソースへのパス。

getAuthor()

説明

リソースの作成者を取得します。

構文

```
public String getAuthor();
```

getComment()

説明

リソースの DAV（Web Distributed Authoring and Versioning）コメントを取得します。

構文

```
public String getComment();
```

getContent()

説明

リソースのコンテンツを戻します。

構文

```
public Object getContent();
```

getContentType()

説明

リソースのコンテンツ・タイプを戻します。

構文

```
public String getContentType();
```

getCreateDate()

説明

リソースの作成日を戻します。

構文

```
public Date getCreateDate();
```

getDisplayName()

説明

リソースの表示名を戻します。

構文

```
public String getDisplayName();
```

getLanguage()

説明

リソースの言語を戻します。

構文

```
public String getLanguage();
```

getLastModDate()

説明

リソースの最終変更日を戻します。

構文

```
public Date getLastModDate();
```

getOwnerId()

説明

リソースの所有者 ID を戻します。このメソッドによって戻される所有者 ID の値は、ALL_USERS などのカタログ・ビューによって提供されるデータベース・ユーザーのユーザー ID の値です。

構文

```
public long getOwnerId();
```

setACL()

説明
リソースに ACL を設定します。

構文
```
public void setACL( String aclpath);
```

パラメータ	説明
aclpath	ACL リソースへのパス。

setAuthor()

説明
リソースの作成者を設定します。

構文
```
public void setAuthor( String authname);
```

パラメータ	説明
authname	リソースの作成者。

setComment()

説明
リソースの DAV（Web Distributed Authoring and Versioning）コメントを設定します。

構文
```
public void setComment(String davcom);
```

パラメータ	説明
davcom	リソースの DAV コメント。

setContent()

説明
リソースのコンテンツを設定します。

構文
```
public void setContent( Object xmlobj);
```

パラメータ	説明
xmlobj	リソースのコンテンツ。

setContentType()

説明
リソースのコンテンツ・タイプを設定します。

構文
```
public void setContentType( String conttype);
```

パラメータ	説明
conttype	リソースのコンテンツ・タイプ。

setCreateDate()

説明
リソースの作成日を設定します。

構文
```
public void setCreateDate( Date credate);
```

パラメータ	説明
credate	リソースの作成日。

setDisplayName()

説明
リソースの表示名を設定します。

構文
```
public void setDisplayName( String dname);
```

パラメータ	説明
dname	リソースの表示名。

setLanguage()

説明
リソースの言語を設定します。

構文
```
public void setLanguage(String lang);
```

パラメータ	説明
lang	リソースの言語。

setLastModDate()

説明
リソースの最終変更日を設定します。

構文
```
public void setLastModDate( Date d);
```

パラメータ	説明
d	リソースの最終変更日。

setOwnerId()

説明

リソースの所有者 ID を設定します。このメソッドによって戻される所有者 ID の値は、ALL_USERS などのカタログ・ビューによって提供されるデータベース・ユーザーのユーザー ID の値です。

構文

```
public void setOwnerId( long ownerid);
```

パラメータ	説明
ownerid	リソースの所有者 ID。

第 VI 部

XML データベース・サポート： Oracle XML DB for PL/SQL

第 VI 部に含まれる章は、次のとおりです。

24

XMLType API for PL/SQL

この章では、サーバー固有の XML サポートに使用される型およびファンクションについて説明します。この章の内容は次のとおりです。

- XMLType の説明
- XMLType のファンクションおよびプロシージャ

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』

XMLType

XMLType の説明

XMLType は、XML データを処理するためのシステム定義の不透明型です。XMLType には、XML ノードや XML フラグメントを抽出するためのメンバー・ファンクションが事前定義されています。

XMLType の列を作成し、それに XML 文書を挿入できます。また、SQL 関数 SYS_XMLGEN および SYS_XMLAGG を使用して、XML 文書を XMLType インスタンスとして動的に生成することもできます。

次に例を示します。

```
CREATE TABLE Xml_tab ( xmlval xmltype);

INSERT INTO Xml_tab VALUES (
   xmltype('<?xml version="1.0"?>
               <EMP>
                  <EMPNO>221</EMPNO>
                  <ENAME>John</ENAME>
               </EMP>'));

INSERT INTO Xml_tab VALUES (
   xmltype('<?xml version="1.0"?>
               <PO>
                  <PONO>331</PONO>
                  <PONAME>PO_1</PONAME>
               </PO>'));
```

ここで、従業員番号の数値を抽出します。

```
SELECT e.xmlval.extract('//EMPNO/text()').getNumVal() as empno
   FROM Xml_tab
   WHERE e.xmlval.existsnode('/EMP/EMPNO')  = 1;
```

XMLType のファンクションおよびプロシージャ

表 24-1 XMLType のファンクションおよびプロシージャの概要

ファンクション	説明
XMLType()（24-4 ページ）	XMLType データ型のインスタンスを作成するコンストラクタです。このコンストラクタは、XML を CLOB または VARCHAR2 として取るか、またはオブジェクト型を取ることができます。
createXML()（24-6 ページ）	XMLType インスタンスを作成し、戻すための静的ファンクションです。
existsNode()（24-7 ページ）	XMLType インスタンスおよび XPath を取り、XPath の適用によって空でないノードの集合が戻される場合は 1、それ以外の場合は 0（ゼロ）を戻します。
extract()（24-8 ページ）	XMLType インスタンスおよび XPath を取り、XPath 式を適用し、結果を XMLType として戻します。
isFragment()（24-9 ページ）	入力 XMLType インスタンスがフラグメントかどうかを確認します。フラグメントは、複数のルート要素を持つ XML インスタンスです。
getClobVal()（24-9 ページ）	XMLType インスタンスの値を CLOB として戻します。
getNumberVal()（24-9 ページ）	XMLType インスタンスの値を NUMBER として戻します。これは、入力 XMLType インスタンスが単純な Text ノードを含み、数値に変換可能な場合にのみ有効です。
getStringVal()（24-9 ページ）	XMLType インスタンスの値を文字列として戻します。
transform()（24-10 ページ）	XMLType インスタンスおよび対応付けられたスタイルシート（それ自体が XMLType インスタンス）を取り、そのスタイルシートを適用し、結果を XML として戻します。
toObject()（24-10 ページ）	XMLType インスタンスをオブジェクト型に変換します。
isSchemaBased()（24-11 ページ）	入力 XMLType インスタンスが XML Schema に基づく場合は 1、XML Schema に基づかない場合は 0（ゼロ）を戻します。
getSchemaURL()（24-11 ページ）	入力が XML Schema に基づく場合は、XML Schema URL を戻します。
getRootElement()（24-11 ページ）	入力インスタンスのルート要素を戻します。インスタンスがフラグメントの場合は、`null` を戻します。
createSchemaBasedXML()（24-12 ページ）	入力スキーマの URL を使用して、XML Schema に基づかないインスタンスから XML Schema に基づく XMLType インスタンスを作成します。

表 24-1 XMLType のファンクションおよびプロシージャの概要（続き）

ファンクション	説明
createNonSchemaBasedXML()（24-12 ページ）	XML Schema に基づく入力インスタンスから XML Schema に基づかない XML を作成します。
getNamespace()（24-12 ページ）	XML Schema に基づく文書内にある最上位要素の名前空間を戻します。
schemaValidate()（24-13 ページ）	XML Schema に基づいて入力インスタンスを検証します。入力インスタンスが XML Schema に基づかない場合、エラーが発生します。
isSchemaValidated()（24-13 ページ）	インスタンスがスキーマに対して検証されたかどうかを確認します。
setSchemaValidated()（24-13 ページ）	高コストのスキーマ検証を回避するためのスキーマ検証済フラグを設定します。
isSchemaValid()（24-14 ページ）	指定されたスキーマの URL に基づいて、入力インスタンスがスキーマに対して妥当であるかどうかを確認します。

XMLType()

説明

XMLType のコンストラクタです。次の表に、オプションを示します。

構文	説明
constructor function XMLType(xmlData IN clob, schema IN varchar2 := NULL, validated IN number := 0, wellformed IN number := 0) return self as result deterministic	オプションで、指定されたスキーマ・パラメータおよび XML データ・パラメータを使用して、XML Schema に基づく XMLType インスタンスを作成します。
constructor function XMLType(xmlData IN varchar2, schema IN varchar2 := NULL, validated IN number := 0, wellformed IN number := 0) return self as result deterministic	オプションで、指定されたスキーマ・パラメータおよび XML データ・パラメータを使用して、XML Schema に基づく XMLType インスタンスを作成します。

構文	説明
constructor function XMLType (xmlData IN "<ADT_1>", schema IN varchar2 := NULL, element IN varchar2 := NULL, validated IN number := 0) return self as result deterministic	オプションで、指定されたオブジェクト型パラメータから XML Schema に基づく XMLType インスタンスを作成します。
onstructor function XMLType(xmlData IN SYS_REFCURSOR, schema in varchar2 := NULL, element in varchar2 := NULL, validated in number := 0) return self as result deterministic	オプションで、指定された REF カーソル・パラメータから XML Schema に基づく XMLType インスタンスを作成します。

パラメータ	IN / OUT	説明
xmlData	(IN)	CLOB、REF カーソル、VARCHAR2 またはオブジェクト型形式の実際のデータ。
schema	(IN)	指定されたスキーマに入力を準拠させるために使用するオプションのスキーマの URL。
validated	(IN)	指定された XML Schema に基づいてインスタンスが妥当であるかどうかを示すフラグ（デフォルトは 0（ゼロ））。
wellformed	(IN)	入力が整形式であるかどうかを示すフラグ。このフラグが設定されている場合、データベースは入力インスタンスが整形式かどうかを確認しません（デフォルトは 0（ゼロ））。
element	(IN)	ADT_1 または REF カーソル・コンストラクタの場合のオプションの要素名（デフォルトは `null`）。

createXML()

説明

XMLType インスタンスを作成し、戻すための静的ファンクションです。データを渡すために使用する文字列パラメータおよび CLOB パラメータには、整形式かつ妥当な XML 文書が含まれる必要があります。次の表に、オプションを示します。

構文	説明
STATIC FUNCTION createXML(xmlData IN varchar2) RETURN XMLType deterministic	文字列から XMLType インスタンスを作成します。
STATIC FUNCTION createXML(xmlData IN clob) RETURN XMLType	CLOB から XMLType インスタンスを作成します。
STATIC FUNCTION createXML (xmlData IN clob, schema IN varchar2, validated IN number := 0, wellformed IN number := 0) RETURN XMLType deterministic	指定されたスキーマ・パラメータおよび XML データ・パラメータを使用して、XML Schema に基づく XMLType インスタンスを作成します。
STATIC FUNCTION createXML (xmlData IN varchar2, schema IN varchar2, validated IN number := 0, wellformed IN number := 0) RETURN XMLType deterministic	指定されたスキーマ・パラメータおよび XML データ・パラメータを使用して、XML Schema に基づく XMLType インスタンスを作成します。
STATIC FUNCTION createXML (xmlData IN "<ADT_1>", schema IN varchar2 := NULL, element IN varchar2 := NULL, validated IN NUMBER := 0) RETURN XMLType deterministic	ユーザー定義型のインスタンスから XML インスタンスを作成します。

構文	説明
STATIC FUNCTION createXML (xmlData IN SYS_REFCURSOR, schema in varchar2 := NULL, element in varchar2 := NULL, validated in number := 0) RETURN XMLType deterministic	REF カーソルから XML インスタンスを作成します。任意の SQL 問合せをカーソルとして渡すことができます。

パラメータ	IN / OUT	説明
xmlData	(IN)	CLOB、REF カーソル、VARCHAR2 またはオブジェクト型形式の実際のデータ。
schema	(IN)	指定されたスキーマに入力を準拠させるために使用するオプションの XML Schema URL。
validated	(IN)	指定された XML Schema に基づいてインスタンスが妥当であるかどうかを示すフラグ（デフォルトは 0（ゼロ））。
wellformed	(IN)	入力が整形式であるかどうかを示すフラグ。このフラグが設定されている場合、データベースは入力インスタンスが整形式かどうかを確認しません（デフォルトは 0（ゼロ））。
element	(IN)	ADT_1 または REF カーソル・コンストラクタの場合のオプションの要素名（デフォルトは null）。

existsNode()

説明

メンバー・ファンクションです。ノードが存在するかどうかを確認します。XPath 文字列が null の場合、または文書が空の場合は、0（ゼロ）を戻します。それ以外の場合は、1 を戻します。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION existsNode(xpath IN varchar2) RETURN number deterministic	指定された XPath 式をドキュメントに適用すると、有効なノードが戻されるかどうかを確認します。

構文	説明
MEMBER FUNCTION existsNode(xpath in varchar2, nsmap in varchar2) RETURN number deterministic	XPath 式と名前空間情報を使用し、XPath の適用によってノードが戻されるかどうかを確認します。

パラメータ	IN / OUT	説明
xpath	(IN)	テストする XPath 式。
nsmap	(IN)	オプションの名前空間マッピング。

extract()

説明

メンバー・ファンクションです。XMLType フラグメントを抽出し、結果ノードを含む XMLType インスタンスを戻します。XPath がどのノードも戻さない場合、null を戻します。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION extract(xpath IN varchar2) RETURN XMLType deterministic	指定された XPath 式をドキュメントに適用し、フラグメントを XMLType として戻します。
MEMBER FUNCTION extract(xpath IN varchar2, nsmap IN varchar2) RETURN XMLType deterministic	XPath 式を名前空間マッピングとともに XML データに適用して、結果のフラグメントを含む XMLType インスタンスを戻します。

パラメータ	IN / OUT	説明
xpath	(IN)	適用する XPath 式。
nsmap	(IN)	名前空間のマッピング情報へのオプションの接頭辞。

isFragment()

説明

XMLType インスタンスが整形式のドキュメントかフラグメントのどちらに対応しているかを判別します。XMLType インスタンスがフラグメントを含んでいる場合は 1、整形式ドキュメントを含んでいる場合は 0（ゼロ）を戻します。

構文

```
MEMBER FUNCTION isFragment RETURN number deterministic
```

getClobVal()

メンバー・ファンクションです。シリアル化された XML 表現を含む CLOB を戻します。一時 CLOB が戻された場合、使用後にその一時 CLOB を解放する必要があります。

構文

```
MEMBER FUNCTION getClobVal RETURN clob deterministic
```

getNumberVal()

説明

メンバー・ファンクションです。XMLType インスタンスが指すテキスト値からフォーマットされる数値を戻します。XMLType は、数値を含む有効な Text ノードを指す必要があります。

構文

```
MEMBER FUNCTION getNumberVal RETURN number deterministic
```

getStringVal()

説明

メンバー・ファンクションです。ドキュメントを文字列として戻します。シリアル化された XML 表現を含む文字列、または Text ノードの場合はテキスト自体を戻します。XML 文書が VARCHAR2 の最大サイズである 4000 バイトより大きい場合は、実行時にエラーが発生します。

構文

```
MEMBER FUNCTION getStringVal RETURN varchar2 deterministic
```

transform()

説明

メンバー・ファンクションです。XSL スタイルシート引数、および名前と値の組の文字列として渡される最上位のパラメータを使用して、XML データを変換します。parammap 以外のいずれかの引数が null の場合、null を戻します。

構文

```
MEMBER FUNCTION transform(xsl IN XMLType, parammap in varchar2 := NULL) RETURN
XMLType deterministic
```

パラメータ	IN / OUT	説明
xsl	(IN)	変換を定義する XSL スタイルシート。
parammap	(IN)	XSL に対する最上位のパラメータ。名前と値の組の文字列です。

toObject()

説明

メンバー・プロシージャです。オプションのスキーマ引数および最上位要素引数を使用して、XML データをユーザー定義型のインスタンスに変換します。

構文

```
MEMBER PROCEDURE toObject(SELF in XMLType, object OUT "<ADT_1>", schema in varchar2
:= NULL, element in varchar2 := NULL)
```

パラメータ	IN / OUT	説明
SELF	(IN)	変換元のインスタンス。メンバー・プロシージャとして使用する場合は暗黙的です。
object	(IN)	変換後のオブジェクト。このファンクションには、必要な型のオブジェクト・インスタンスが渡されます。
schema	(IN)	スキーマの URL。XMLType インスタンスと変換後のオブジェクト・インスタンスのマッピングは、スキーマを使用して指定できます。

パラメータ	IN / OUT	説明
element	(IN)	最上位要素名。XML Schema 文書は、準拠する XML インスタンス・ドキュメントの最上位要素を指定しません。XMLType インスタンスをマップするための、XML Schema 文書内の最上位要素名は、このパラメータを使用して指定します。

isSchemaBased()

説明

メンバー・ファンクションです。XMLType インスタンスが XML Schema に基づくかどうかを判別します。XMLType インスタンスが XML Schema に基づく場合は 1、XML Schema に基づかない場合は 0（ゼロ）を戻します。

構文

```
MEMBER FUNCTION isSchemaBased return number deterministic
```

getSchemaURL()

説明

メンバー・ファンクションです。XMLType インスタンスが XML Schema に基づく文書である場合は、その XMLType インスタンスに対応する XML Schema URL を戻します。XML Schema に基づかない文書の場合は、`null` を戻します。

構文

```
MEMBER FUNCTION getSchemaURL return varchar2 deterministic
```

getRootElement()

説明

メンバー・ファンクションです。XMLType インスタンスのルート要素を取得します。インスタンスがフラグメントの場合は、`null` を戻します。

構文

```
MEMBER FUNCTION getRootElement return varchar2 deterministic
```

createSchemaBasedXML()

説明

メンバー・ファンクションです。XML Schema に基づかない XML およびスキーマの URL から XML Schema に基づく XMLType インスタンスを作成します。

構文

```
MEMBER FUNCTION createSchemaBasedXML(schema IN varchar2 := NULL) return XMLType
deterministic
```

パラメータ	IN / OUT	説明
schema	(IN)	スキーマの URL。このパラメータが null の場合は、XMLType インスタンスにスキーマの URL が含まれている必要があります。

createNonSchemaBasedXML()

説明

メンバー・ファンクションです。XML Schema に基づくインスタンスから XML Schema に基づかない XML 文書を作成します。

構文

```
MEMBER FUNCTION createNonSchemaBasedXML return XMLType deterministic
```

getNamespace()

説明

メンバー・ファンクションです。インスタンス内の最上位要素の名前空間を戻します。入力がフラグメントまたは XML Schema に基づかないインスタンスである場合は、null を戻します。

構文

```
MEMBER FUNCTION getNamespace return varchar2 deterministic
```

schemaValidate()

説明

メンバー・プロシージャです。XML インスタンスがそのスキーマに対して検証されていない場合、その検証を行います。XML Schema に基づかない文書の場合、エラーが発生します。検証が正常に実行されなかった場合は、エラーが発生します。正常に実行された場合は、ドキュメントの状態が検証済に変更されます。

構文

```
MEMBER PROCEDURE schemaValidate(self IF OUT NOCOPY XMLType)
```

isSchemaValidated()

説明

メンバー・ファンクションです。XMLType インスタンスの検証状態を戻し、XML Schema に基づくインスタンスが、実際にそのスキーマに対して検証されたかどうかを通知します。インスタンスがスキーマに対して検証済である場合は 1、検証済でない場合は 0（ゼロ）を戻します。

構文

```
MEMBER FUNCTION isSchemaValidated return NUMBER deterministic
```

setSchemaValidated()

説明

メンバー・ファンクションです。入力 XML インスタンスの検証状態を設定します。

構文

```
MEMBER PROCEDURE setSchemaValidated(self IN OUT NOCOPY XMLType,
                                    flag IN BINARY_INTEGER := 1)
```

パラメータ	IN / OUT	説明
flag	(IN)	未検証の場合は 0（ゼロ）、検証済の場合は 1。デフォルト値は 1 です。

isSchemaValid()

説明

メンバー・ファンクションです。入力インスタンスが指定されたスキーマに準拠しているかどうかを確認します。XML インスタンスの検証状態は変更しません。XML Schema URL が指定されておらず、XML 文書が XML Schema に基づく場合、XMLType インスタンス自身のスキーマに対して準拠しているかどうかが確認されます。

構文

```
member function isSchemaValid(schurl IN VARCHAR2 := NULL, elem IN VARCHAR2 := NULL)
return NUMBER deterministic
```

パラメータ	IN / OUT	説明
schurl	(IN)	準拠しているかどうかの確認基準となる XML Schema URL。
elem	(IN)	検証の基準となる指定されたスキーマの要素。これは、複数の最上位要素を定義する XML Schema があり、これらの要素のうち、特定の要素に対して準拠しているかどうかを確認する必要がある場合に有効です。

25

PL/SQL DOM API for XMLType

DBMS_XMLDOM パッケージ内にある XML Parser for PL/SQL は、XMLType オブジェクト、および XML Schema に基づく場合と XML Schema に基づかない場合の両方の文書にアクセスするために使用されます。

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

DBMS_XMLDOM パッケージ

DBMS_XMLDOM の説明

ドキュメント・オブジェクト・モデル（DOM）は、HTML ドキュメントおよび XML 文書用の Application Program Interface（API）です。DOM は、ドキュメントの論理構造、ドキュメントへのアクセス方法および操作方法を定義します。DOM 仕様では、「ドキュメント」という用語は、幅広い意味で使用されます。XML は、多様なシステムに格納できる様々な情報を表す手段としてますます需要が高まっていますが、情報の多くは、従来、ドキュメントとしてではなくデータとして認識されてきました。これに対し、XML はデータをドキュメントとして表し、DOM を使用して、このデータを管理します。

プログラマは、DOM を使用してドキュメントを構築し、その構造をナビゲートしたり、要素およびコンテンツを追加、変更および削除することができます。HTML ドキュメントまたは XML 文書のすべてのコンテンツは、DOM を使用してアクセス、変更、削除または追加できます。ただし、いくつかの例外もあります。特に、XML の内部および外部サブセットに対する DOM インタフェースは、まだ仕様が確立されていません。

W3C による DOM 仕様の重要な目的の 1 つは、様々な環境およびアプリケーションで使用できる標準プログラミング・インタフェースを提供することです。DOM は、すべてのプログラミング言語で使用できるように設計されています。DOM 標準はオブジェクト指向であるため、PL/SQL に適用するには、次の変更が必要となります。

- Node、Element などの様々な DOM インタフェースには、それぞれ同等の PL/SQL 型 DOMNode や DOMElement があります。
- WRONG_DOCUMENT_ERR、HIERARCHY_REQUEST_ERR などの様々な DOMException コードには、同様の名前が付いた PL/SQL 例外があります。
- ELEMENT_NODE、ATTRIBUTE_NODE などの様々な DOM ノードの型コードには、同様の名前が付いた PL/SQL 定数があります。
- DOM 型で定義済のメソッドは、DOM 型をパラメータとして取るファンクションまたはプロシージャになります。たとえば、appendChild を DOM ノード n で実行するには、25-28 ページの PL/SQL ファンクション appendChild() を次のとおり実行します。

```
FUNCTION appendChild( n DOMNode,
                      newChild IN DOMNode)
                      RETURN DOMNode;
```

また、setAttribute を DOM 要素 elem で実行するには、25-59 ページの PL/SQL プロシージャ setAttribute() を次のとおり実行します。

```
PROCEDURE setAttribute( elem DOMElement,
                        name IN VARCHAR2,
                        value IN VARCHAR2);
```

DOM は、継承階層を定義します。たとえば、ドキュメント、要素および属性は、ノードのサブタイプとして定義されます。このため、Node インタフェースで定義済のメソッドは、ドキュメント、要素および属性でも使用可能です。PL/SQL の場合、直接このような継承を行うことはできないため、異なる DOM 型で makeNode ファンクションをコールし、これらの DOM 型を DOMNode に変換する必要があります。この変換後に、DOMNode を指定する適切なファンクションまたはプロシージャをコールすると、これらの型を操作できるようになります。その後、型固有の機能が必要になった場合、make*() ファンクションを使用して、DOMNode を元の型に戻すこともできます。この場合、DOM* は必要な DOM 型を表します。

この PL/SQL DOM インタフェースの実装は、改訂 REC-DOM-Level-1-19981001 の DOM 標準に準拠します。このマニュアルで説明する型およびメソッドは、PL/SQL パッケージ DBMS_XMLDOM で使用可能です。

- データベースを起動する前に、initialization.ORA ファイル内で読込み元ディレクトリおよび書込み先ディレクトリを指定する必要があります。次に例を示します。

```
UTL_FILE_DIR=/mypath/insidemypath
```

- 読込み元ファイルおよび書込み先ファイルは、サーバー・ファイル・システム上に存在する必要があります。

DBMS_XMLDOM の型

次に、DBMS_XMLDOM.DOMTYPE に定義されている型を示します。

表 25-1 DBMS_XMLDOM の型

型	説明
DOMNode	DOM の Node インタフェースを実装します。
DOMNamedNodeMap	DOM の NamedNodeMap インタフェースを実装します。
DOMNodeList	DOM の NodeList インタフェースを実装します。
DOMAttr	DOM の Attribute インタフェースを実装します。
DOMCDataSection	DOM の CDataSection インタフェースを実装します。
DOMCharacterData	DOM の Character Data インタフェースを実装します。
DOMComment	DOM の Comment インタフェースを実装します。
DOMDocumentFragment	DOM の DocumentFragment インタフェースを実装します。
DOMElement	DOM の Element インタフェースを実装します。
DOMEntity	DOM の Entity インタフェースを実装します。

表 25-1 DBMS_XMLDOM の型（続き）

型	説明
DOMEntityReference	DOM の EntityReference インタフェースを実装します。
DOMNotation	DOM の Notation インタフェースを実装します。
DOMProcessingInstruction	DOM の Processing Instruction インタフェースを実装します。
DOMText	DOM の Text インタフェースを実装します。
DOMImplementation	DOM の DOMImplementation インタフェースを実装します。
DOMDocumentType	DOM の Document Type インタフェースを実装します。
DOMDocument	DOM の Document インタフェースを実装します。

DBMS_XMLDOM の事前定義定数

表 25-2 に、DBMS_XMLDOM に定義されている定数を示します。たとえば、getNodeType(myNode) などのリクエストが行われた場合、その戻り型は次のいずれかの定数になります。

表 25-2 DBMS_XMLDOM の事前定義定数

定数	説明
ELEMENT_NODE	ノードは要素です。
ATTRIBUTE_NODE	ノードは属性です。
TEXT_NODE	ノードは Text ノードです。
CDATA_SECTION_NODE	ノードは CDATA セクションです。
ENTITY_REFERENCE_NODE	ノードは実体参照です。
ENTITY_NODE	ノードはエンティティです。
PROCESSING_INSTRUCTION_NODE	ノードは処理命令です。
COMMENT_NODE	ノードはコメントです。
DOCUMENT_NODE	ノードはドキュメントです。
DOCUMENT_TYPE_NODE	ノードは Document Type Definition です。
DOCUMENT_FRAGMENT_NODE	ノードはドキュメント・フラグメントです。
NOTATION_NODE	ノードは表記法です。

DBMS_XMLDOM の例外

表 25-3 に、DBMS_XMLDOM に定義されている例外を示します。

表 25-3 DBMS_XMLDOM の例外

例外	説明
INDEX_SIZE_ERR	インデックスまたはサイズが負であるか、または許容値より大きい場合。
DOMSTRING_SIZE_ERR	指定された範囲のテキストが DOMString 文字列より多い場合。
HIERARCHY_REQUEST_ERR	任意のノードが属さない場所に挿入された場合。
WRONG_DOCUMENT_ERR	ノードが、そのノードを作成したドキュメントとは別の（そのノードをサポートしない）ドキュメントで使用された場合。
INVALID_CHARACTER_ERR	名前などに、無効または不正な文字が指定された場合。正当な文字の定義については、XML 仕様の production 2 を参照してください。また、正当な名前文字の定義については、production 5 を参照してください。
NO_DATA_ALLOWED_ERROR	データをサポートしないノードでデータが指定された場合。
NO_MODIFICATION_ALLOWED_ERR	変更が許可されていないオブジェクトを変更しようとした場合。
NO_FOUND_ERR	ノードが存在しないコンテキストでノードを参照しようとした場合。
NOT_SUPPORTED_ERR	リクエストされた種類のオブジェクトや操作を実装がサポートしない場合。
INUSE_ATTRIBUTE_ERR	すでに他の場所で使用されている属性を追加しようとした場合。

DBMS_XMLDOM のメソッド

DBMS_XMLDOM のサブプログラムは、W3C のインタフェースに基づいて、グループに分類されます。

表 25-4 DBMS_XMLDOM のメソッドの概要

グループまたはメソッド	説明
DOM ノード・メソッド	
isNull()（25-13 ページ）	ノードが null かどうかをテストします。
makeAttr()（25-13 ページ）	ノードを属性にキャストします。

表 25-4 DBMS_XMLDOM のメソッドの概要（続き）

グループまたはメソッド	説明
makeCDataSection()（25-14 ページ）	ノードを CDATA セクションにキャストします。
makeCharacterData()（25-14 ページ）	ノードを文字データにキャストします。
makeComment()（25-15 ページ）	ノードをコメントにキャストします。
makeDocumentFragment()（25-15 ページ）	ノードをドキュメント・フラグメントにキャストします。
makeDocumentType()（25-16 ページ）	ノードをドキュメント・タイプにキャストします。
makeElement()（25-16 ページ）	ノードを要素にキャストします。
makeEntity()（25-17 ページ）	ノードをエンティティにキャストします。
makeEntityReference()（25-17 ページ）	ノードを実体参照にキャストします。
makeNotation()（25-17 ページ）	ノードを表記法にキャストします。
makeProcessingInstruction()（25-18 ページ）	ノードを DOM 処理命令にキャストします。
makeText()（25-18 ページ）	ノードを DOM テキストにキャストします。
makeDocument()（25-18 ページ）	ノードを DOM 文書にキャストします。
writeToFile()（25-19 ページ）	ノードのコンテンツをファイルに書き込みます。
writeToBuffer()（25-19 ページ）	ノードのコンテンツをバッファに書き込みます。
writeToClob()（25-20 ページ）	ノードのコンテンツを CLOB に書き込みます。
getNodeName()（25-21 ページ）	ノードの名前を取得します。
getNodeValue()（25-21 ページ）	ノードの値を取得します。
setNodeValue()（25-22 ページ）	ノードの値を設定します。
getNodeType()（25-22 ページ）	ノードのタイプを取得します。
getParentNode()（25-23 ページ）	ノードの親ノードを取得します。
getChildNodes()（25-23 ページ）	ノードの子ノードを取得します。
getFirstChild()（25-24 ページ）	ノードの最初の子ノードを取得します。
getLastChild()（25-24 ページ）	ノードの最後の子ノードを取得します。
getPreviousSibling()（25-25 ページ）	ノードの直前の兄弟関係を取得します。
getNextSibling()（25-25 ページ）	ノードの直後の兄弟関係を取得します。

表 25-4 DBMS_XMLDOM のメソッドの概要（続き）

グループまたはメソッド	説明
getAttributes()（25-26 ページ）	ノードの属性を取得します。
getOwnerDocument()（25-26 ページ）	ノードの所有者ドキュメントを取得します。
insertBefore()（25-27 ページ）	参照先の子ノードの前に子ノードを挿入します。
replaceChild()（25-27 ページ）	古い子ノードを新しい子ノードで置換します。
removeChild()（25-28 ページ）	指定された子ノードをノードから削除します。
appendChild()（25-28 ページ）	ノードに新しい子ノードを追加します。
hasChildNodes()（25-29 ページ）	ノードに子ノードが存在するかどうかをテストします。
cloneNode()（25-29 ページ）	ノードを複製します。
DOM 名前付きノード・マップ・メソッド	
isNull()（25-30 ページ）	名前付きノード・マップが null かどうかをテストします。
getNamedItem()（25-30 ページ）	名前で指定された項目を取得します。
setNamedItem()（25-31 ページ）	名前で指定された項目をマッピング内に設定します。
removeNamedItem()（25-31 ページ）	名前で指定された項目を削除します。
item()（25-32 ページ）	マッピング内の指定されたインデックスの項目を取得します。
getLength()（25-32 ページ）	マッピング内の項目数を取得します。
DOM ノード・リスト・メソッド	
isNull()（25-33 ページ）	NodeList が null かどうかをテストします。
item()（25-33 ページ）	NodeList 内の指定されたインデックスの項目を取得します。
getLength()（25-34 ページ）	リスト内の項目数を取得します。
DOM 属性メソッド	
isNull()（25-34 ページ）	属性ノードが null かどうかをテストします。
makeNode()（25-35 ページ）	属性をノードにキャストします。
getQualifiedName()（25-35 ページ）	属性の修飾名を取得します。
getNamespace()（25-35 ページ）	属性の名前空間 URI を取得します。
getLocalName()（25-36 ページ）	属性のローカル名を取得します。

表 25-4 DBMS_XMLDOM のメソッドの概要（続き）

グループまたはメソッド	説明
getExpandedName()（25-36 ページ）	属性の拡張名を取得します。
getName()（25-36 ページ）	属性の名前を取得します。
getSpecified()（25-37 ページ）	属性が元の要素で指定されているかどうかをテストします。
getValue()（25-37 ページ）	属性値を取得します。
setValue()（25-38 ページ）	属性値を設定します。
DOM CDATA セクション・メソッド	
isNull()（25-38 ページ）	CDATA セクションが null かどうかをテストします。
makeNode()（25-39 ページ）	CDATA セクションをノードにキャストします。
DOM 文字データ・メソッド	
isNull()（25-39 ページ）	文字データが null かどうかをテストします。
makeNode()（25-40 ページ）	文字データをノードにキャストします。
getData()（25-40 ページ）	ノードのデータを取得します。
setData()（25-41 ページ）	データをノードに設定します。
getLength()（25-41 ページ）	データの長さを取得します。
substringData()（25-42 ページ）	データの部分文字列を取得します。
appendData()（25-42 ページ）	指定されたデータをノードのデータに追加します。
insertData()（25-43 ページ）	指定されたオフセットでノードにデータを挿入します。
deleteData()（25-43 ページ）	指定されたオフセットからデータを削除します。
replaceData()（25-44 ページ）	指定されたオフセットからデータを置換します。
DOM コメント・メソッド	
isNull()（25-44 ページ）	コメントが null かどうかをテストします。
makeNode()（25-45 ページ）	コメントをノードにキャストします。
DOM インプリメンテーション・メソッド	
isNull()（25-45 ページ）	DOM インプリメンテーション・ノードが null かどうかをテストします。
hasFeature()（25-46 ページ）	DOM インプリメンテーションが、指定された機能を実装するかどうかをテストします。

表 25-4 DBMS_XMLDOM のメソッドの概要（続き）

グループまたはメソッド	説明
DOM ドキュメント・フラグメント・メソッド	
isNull()（25-46 ページ）	ドキュメント・フラグメントが null かどうかをテストします。
makeNode()（25-47 ページ）	ドキュメント・フラグメントをノードにキャストします。
DOM ドキュメント・タイプ・メソッド	
isNull()（25-47 ページ）	ドキュメント・タイプが null かどうかをテストします。
makeNode()（25-48 ページ）	ドキュメント・タイプをノードにキャストします。
findEntity()（25-48 ページ）	ドキュメント・タイプ内の指定されたエンティティを検出します。
findNotation()（25-49 ページ）	ドキュメント・タイプ内の指定された表記法を検出します。
getPublicId()（25-49 ページ）	ドキュメント・タイプの公開識別子を取得します。
getSystemId()（25-50 ページ）	ドキュメント・タイプのシステム識別子を取得します。
writeExternalDTDToFile()（25-50 ページ）	Document Type Definition をファイルに書き込みます。
writeExternalDTDToBuffer()（25-51 ページ）	Document Type Definition をバッファに書き込みます。
writeExternalDTDToClob()（25-52 ページ）	Document Type Definition を CLOB に書き込みます。
getName()（25-52 ページ）	ドキュメント・タイプの名前を取得します。
getEntities()（25-53 ページ）	ドキュメント・タイプ内のエンティティのノード・マップを取得します。
getNotations()（25-53 ページ）	ドキュメント・タイプ内の表記法のノード・マップを取得します。
DOM 要素メソッド	
isNull()（25-54 ページ）	要素が null かどうかをテストします。
makeNode()（25-54 ページ）	要素をノードにキャストします。
getQualifiedName()（25-55 ページ）	要素の修飾名を取得します。
getNamespace()（25-55 ページ）	要素の名前空間 URI を取得します。

表 25-4 DBMS_XMLDOM のメソッドの概要（続き）

グループまたはメソッド	説明
getLocalName()（25-55 ページ）	要素のローカル名を取得します。
getExpandedName()（25-56 ページ）	要素の拡張名を取得します。
getChildrenByTagName()（25-56 ページ）	任意のタグ名の要素の子を取得します。
getElementsByTagName()（25-57 ページ）	サブツリー内の任意のタグ名の要素を取得します。
resolveNamespacePrefix()（25-58 ページ）	接頭辞を名前空間 URI に解決します。
getTagName()（25-58 ページ）	要素のタグ名を取得します。
getAttribute()（25-58 ページ）	名前で指定された属性ノードを取得します。
setAttribute()（25-59 ページ）	名前で指定された属性を設定します。
removeAttribute()（25-59 ページ）	名前で指定された属性を削除します。
getAttributeNode()（25-60 ページ）	名前で指定された属性ノードを取得します。
setAttributeNode()（25-60 ページ）	要素に属性ノードを設定します。
removeAttributeNode()（25-61 ページ）	要素内の属性ノードを削除します。
normalize()（25-61 ページ）	要素の子である Text ノードを正規化します。
DOM エンティティ・メソッド	
isNull()（25-62 ページ）	エンティティが null かどうかをテストします。
makeNode()（25-62 ページ）	エンティティをノードにキャストします。
getPublicId()（25-63 ページ）	エンティティの公開識別子を取得します。
getSystemId()（25-63 ページ）	エンティティのシステム識別子を取得します。
getNotationName()（25-63 ページ）	エンティティの表記法名を取得します。
DOM 実体参照メソッド	
isNull()（25-64 ページ）	実体参照が null かどうかをテストします。
makeNode()（25-64 ページ）	実体参照を null にキャストします。

表 25-4 DBMS_XMLDOM のメソッドの概要（続き）

グループまたはメソッド	説明
DOM 表記法メソッド	
isNull()（25-65 ページ）	表記法が null かどうかをテストします。
makeNode()（25-65 ページ）	表記法をノードにキャストします。
getPublicId()（25-66 ページ）	表記法の公開識別子を取得します。
getSystemId()（25-66 ページ）	表記法のシステム識別子を取得します。
DOM 処理命令メソッド	
isNull()（25-67 ページ）	処理命令が null かどうかをテストします。
makeNode()（25-67 ページ）	処理命令をノードにキャストします。
getData()（25-68 ページ）	処理命令のデータを取得します。
getTarget()（25-68 ページ）	処理命令のターゲットを取得します。
setData()（25-69 ページ）	処理命令のデータを設定します。
DOM テキスト・メソッド	
isNull()（25-69 ページ）	テキストが null かどうかをテストします。
makeNode()（25-70 ページ）	テキストをノードにキャストします。
splitText()（25-70 ページ）	Text ノードのコンテンツを 2 つの Text ノードに分割します。
DOM 文書メソッド	
isNull()（25-71 ページ）	文書が null かどうかをテストします。
makeNode()（25-71 ページ）	文書をノードにキャストします。
newDOMDocument()（25-71 ページ）	新しい文書を作成します。
freeDocument()（25-72 ページ）	文書を解放します。
getVersion()（25-72 ページ）	文書のバージョンを取得します。
setVersion()（25-73 ページ）	文書のバージョンを設定します。
getCharset()（25-73 ページ）	文書のキャラクタ・セットを取得します。
setCharset()（25-74 ページ）	文書のキャラクタ・セットを設定します。
getStandalone()（25-74 ページ）	文書が単独に指定されているかどうかを取得します。
setStandalone()（25-75 ページ）	文書を単独に設定します。

表 25-4 DBMS_XMLDOM のメソッドの概要（続き）

グループまたはメソッド	説明
writeToFile()（25-75 ページ）	文書をファイルに書き込みます。
writeToBuffer()（25-76 ページ）	文書をバッファに書き込みます。
writeToClob()（25-76 ページ）	文書を CLOB に書き込みます。
writeExternalDTDToFile()（25-77 ページ）	文書の DTD をファイルに書き込みます。
writeExternalDTDToBuffer()（25-78 ページ）	文書の DTD をバッファに書き込みます。
writeExternalDTDToClob()（25-78 ページ）	文書の DTD を CLOB に書き込みます。
getDoctype()（25-79 ページ）	文書の DTD を取得します。
getImplementation()（25-79 ページ）	DOM インプリメンテーションを取得します。
getDocumentElement()（25-80 ページ）	文書のルート要素を取得します。
createElement()（25-80 ページ）	新しい要素を作成します。
createDocumentFragment()（25-81 ページ）	新しい文書フラグメントを作成します。
createTextNode()（25-81 ページ）	Text ノードを作成します。
createComment()（25-82 ページ）	コメント・ノードを作成します。
createCDATASection()（25-82 ページ）	CDATA セクション・ノードを作成します。
createProcessingInstruction()（25-83 ページ）	処理命令を作成します。
createAttribute()（25-83 ページ）	属性を作成します。
createEntityReference()（25-84 ページ）	実体参照を作成します。
getElementsByTagName()（25-84 ページ）	サブツリー内の任意のタグ名の要素を取得します。

DOM ノード・メソッド

isNull()

説明

指定された DOM ノードが null かどうかを確認します。null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( n DOMNode) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
n	(IN)	確認する DOM ノード。

makeAttr()

説明

指定された DOM ノードを DOM 属性にキャストし、その DOM 属性を戻します。

構文

```
FUNCTION makeAttr( n DOMNode) RETURN DOMAttr;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeCDataSection()

説明

指定された DOM ノードを DOM CDATA セクションにキャストし、その DOM CDATA セクションを戻します。

構文

```
FUNCTION makeCDataSection( n DOMNode) RETURN DOMCDataSection;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeCharacterData()

説明

指定された DOM ノードを DOM 文字データにキャストし、その DOM 文字データを戻します。

構文

```
FUNCTION makeCharacterData( n DOMNode) RETURN DOMCharacterData;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeComment()

説明

指定された DOM ノードを DOM コメントにキャストし、その DOM コメントを戻します。

構文

```
FUNCTION makeComment(n DOMNode) RETURN DOMComment;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeDocumentFragment()

説明

指定された DOM ノードを DOM ドキュメント・フラグメントにキャストし、その DOM ドキュメント・フラグメントを戻します。

構文

```
FUNCTION makeDocumentFragment( n DOMNode) RETURN DOMDocumentFragment;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeDocumentType()

説明

指定された DOM ノードを DOM ドキュメント・タイプにキャストし、その DOM ドキュメント・タイプを戻します。

構文

```
FUNCTION makeDocumentType(n DOMNode) RETURN DOMDocumentType;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeElement()

説明

指定された DOM ノードを DOM 要素にキャストし、その DOM 要素を戻します。

構文

```
FUNCTION makeElement(n DOMNode) RETURN DOMElement;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeEntity()

説明

指定された DOM ノードを DOM エンティティにキャストし、その DOM エンティティを戻します。

構文

```
FUNCTION makeEntity(n DOMNode) RETURN DOMEntity;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeEntityReference()

説明

指定された DOM ノードを DOM 実体参照にキャストし、その DOM 実体参照を戻します。

構文

```
FUNCTION makeEntityReference(n DOMNode) RETURN DOMEntityReference;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeNotation()

説明

指定された DOM ノードを DOM 表記法にキャストし、その DOM 表記法を戻します。

構文

```
FUNCTION makeNotation(n DOMNode) RETURN DOMNotation;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeProcessingInstruction()

説明

指定された DOM ノードを DOM 処理命令にキャストし、その DOM 処理命令を戻します。

構文

```
FUNCTION makeProcessingInstruction( n DOMNode)
                                    RETURN DOMProcessingInstruction;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeText()

説明

指定された DOM ノードを DOM テキストにキャストし、その DOM テキストを戻します。

構文

```
FUNCTION makeText(n DOMNode) RETURN DOMText;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

makeDocument()

説明

指定された DOM ノードを DOM 文書にキャストし、その DOM 文書を戻します。

構文

```
FUNCTION makeDocument(n DOMNode) RETURN DOMDocument;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM ノード。

writeToFile()

説明

指定されたファイルに XML ノードを書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeToFile(n DOMNode, fileName VARCHAR2);	データベース・キャラクタ・セットを使用して、指定されたファイルに XML ノードを書き込みます。
PROCEDURE writeToFile(n DOMNode, fileName VARCHAR2, charset VARCHAR2);	個別のパラメータとして渡されるキャラクタ・セットを使用して、指定されたファイルに XML ノードを書き込みます。

パラメータ	IN / OUT	説明
n	(IN)	DOM ノード。
fileName	(IN)	書込み先のファイル。
charset	(IN)	指定されたキャラクタ・セット。

writeToBuffer()

説明

指定されたバッファに XML ノードを書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2);	データベース・キャラクタ・セットを使用して、指定されたバッファに XML ノードを書き込みます。
PROCEDURE writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2, charset VARCHAR2);	個別のパラメータとして渡されるキャラクタ・セットを使用して、指定されたバッファに XML ノードを書き込みます。

パラメータ	IN / OUT	説明
n	(IN)	DOM ノード。
buffer	(IN/OUT)	書込み先のバッファ。
charset	(IN)	指定されたキャラクタ・セット。

writeToClob()

説明

指定された CLOB に XML ノードを書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeToClob(n DOMNode, cl IN OUT CLOB);	データベース・キャラクタ・セットを使用して、指定された CLOB に XML ノードを書き込みます。
PROCEDURE writeToClob(n DOMNode, cl IN OUT CLOB, charset VARCHAR2);	個別のパラメータとして渡されるキャラクタ・セットを使用して、指定された CLOB に XML ノードを書き込みます。

パラメータ	IN / OUT	説明
n	(IN)	DOM ノード。
cl	(IN/OUT)	書込み先の CLOB。
charset	(IN)	指定されたキャラクタ・セット。

getNodeName()

説明

ノードの名前を、ノードのタイプに応じて取得します。

構文

```
FUNCTION getNodeName(n DOMNode) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

getNodeValue()

説明

このノードの値を、ノードのタイプに応じて取得します。

構文

```
FUNCTION getNodeValue(n DOMNode) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

setNodeValue()

説明

このノードの値を、ノードのタイプに応じて設定します。ノードの値が null になるように定義されている場合は、ノードの値を設定しても効果はありません。

構文

```
PROCEDURE setNodeValue( n DOMNode,
                        nodeValue IN VARCHAR2);
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。
nodeValue	IN	ノードに設定する値。

getNodeType()

説明

実際の実装オブジェクトの型を表すコードを戻します。

構文

```
FUNCTION getNodeType(n DOMNode) RETURN NUMBER;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

getParentNode()

説明

ノードの親ノードを取得します。属性、ドキュメント、ドキュメント・フラグメント、エンティティ、表記法以外のすべてのノードは、親ノードを持ちます。ただし、ノードが作成された直後でツリーに追加されていない場合、またはツリーから削除された場合は、null を戻します。

構文

```
FUNCTION getParentNode(n DOMNode) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

getChildNodes()

説明

このノードのすべての子ノードを含む NodeList を取得します。子ノードが存在しない場合は、ノードを含まない NodeList を戻します。

構文

```
FUNCTION getChildNodes(n DOMNode) RETURN DOMNodeList;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

getFirstChild()

説明

ノードの最初の子ノードを取得します。該当するノードが存在しない場合は、null を戻します。

構文

```
FUNCTION getFirstChild( n DOMNode) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

getLastChild()

説明

ノードの最後の子ノードを取得します。該当するノードが存在しない場合は、null を戻します。

構文

```
FUNCTION getLastChild( n DOMNode) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

getPreviousSibling()

説明

このノードの直前のノードを取得します。該当するノードが存在しない場合は、null を戻します。

構文

```
FUNCTION getPreviousSibling( n DOMNode) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

getNextSibling()

説明

このノードの直後のノードを取得します。該当するノードが存在しない場合は、null を戻します。

構文

```
FUNCTION getNextSibling( n DOMNode) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

getAttributes()

説明

ノードが要素の場合は、ノードの属性を含む名前付きノード・マップを取得します。ノードが要素でない場合は、null を取得します。

構文

```
FUNCTION getAttributes( n DOMNode) RETURN DOMNamedNodeMap;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

getOwnerDocument()

説明

ノードに対応するドキュメント・オブジェクトを取得します。これは、新しいノードを作成するために使用されるドキュメント・オブジェクトでもあります。このノードがドキュメント、またはどのドキュメントでも使用されていないドキュメント・タイプである場合は、null を戻します。

構文

```
FUNCTION getOwnerDocument( n DOMNode) RETURN DOMDocument;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

insertBefore()

説明

newChild ノードを既存の子ノード refChild の前に挿入します。refChild が null の場合は、子のリストの最後に newChild を挿入します。

newChild がドキュメント・フラグメント・オブジェクトの場合は、そのすべての子が同じ順序で refChild の前に挿入されます。newChild がツリー内にすでに存在する場合は、最初に削除されます。

構文

```
FUNCTION insertBefore( n DOMNode,
                       newChild IN DOMNode,
                       refChild IN DOMNode)
                       RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。
newChild	IN	n という DOM ノードに挿入する子ノード。
refChild	IN	newChild が前に挿入される参照ノード。

replaceChild()

説明

子のリスト内の子ノード oldChild を newChild で置換し、oldChild ノードを戻します。

newChild がドキュメント・フラグメント・オブジェクトの場合、oldChild はそのドキュメント・フラグメントのすべての子に置換されます。これらの子は、同じ順序で挿入されます。newChild がツリー内にすでに存在する場合は、最初に削除されます。

構文

```
FUNCTION replaceChild( n DOMNode,
                       newChild IN DOMNode,
                       oldChild IN DOMNode)
                       RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。
newChild	IN	oldChild と置換する新しい子ノード。
oldChild	IN	ノード n の置換される子ノード。

removeChild()

説明

oldChild に指定された子ノードを子のリストから削除します。

構文

```
FUNCTION removeChild( n DOMNode,
                      oldChild IN DOMNode)
                      RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。
oldChild	IN	ノード n の削除される子ノード。

appendChild()

説明

ノードの子のリストの最後に newChild ノードを追加します。newChild がツリー内にすでに存在する場合は、最初に削除されます。

構文

```
FUNCTION appendChild( n DOMNode,
                      newChild IN DOMNode)
                      RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。
newChild	IN	ノード n の子のリストに追加する子ノード。

hasChildNodes()

説明

ノードに子ノードが存在するかどうかを戻します。

構文

```
FUNCTION hasChildNodes( n DOMNode) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。

cloneNode()

説明

ノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。複製ノードには親ノードが存在しません。parentNode は null を戻します。

要素を複製すると、XML プロセッサによって生成されたデフォルト値の属性を含む、すべての属性およびその値がコピーされます。ただし、このメソッドでは、ノードに含まれるテキストは、子である Text ノードに含まれているため、ディープ・クローンの場合以外はコピーされません。要素の複製操作の一部として属性を複製した場合に対し、属性を直接複製すると、指定された属性が戻されます（true が指定されている場合）。他のタイプのノードを複製すると、単純にこのノードのコピーが戻されます。

不変サブツリーを複製すると、可変コピーが生成されますが、複製された実体参照の子は読込み専用であることに注意してください。未指定の Attr ノードの複製も指定されます。また、ドキュメント・ノード、ドキュメント・タイプ・ノード、エンティティ・ノードおよび表記法ノードの複製は、実装によって異なります。

構文

```
FUNCTION cloneNode( n DOMNode,
                    deep boolean)
                    RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOM ノード。
deep	IN	子を複製するかどうかを判別するブール値。

DOM 名前付きノード・マップ・メソッド

isNull()

説明

指定された DOM 名前付きノード・マップが null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( nnm DOMNamedNodeMap) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
nnm	(IN)	確認する DOM 名前付きノード・マップ。

getNamedItem()

説明

名前で指定されたノードを取得します。

構文

```
FUNCTION getNamedItem( nnm DOMNamedNodeMap,
                       name IN VARCHAR2)
                       RETURN DOMNode;
```

パラメータ	IN / OUT	説明
nnm	IN	DOM 名前付きノード・マップ。
name	IN	取得する項目の名前。

setNamedItem()

説明

ノードの nodeName 属性を使用して、ノードを追加します。その名前を持つノードがすでにこのマッピング内に存在する場合、そのノードは新しいノードに置換されます。

nodeName 属性はノードが格納される名前を派生させるために使用されるため、特定の型の複数のノード（特殊な文字列値を持つもの）は、名前が競合するため格納できません。これは、ノードに別名を許可するより優先されます。

構文

```
FUNCTION setNamedItem( nnm DOMNamedNodeMap,
                       arg IN DOMNode)
                       RETURN DOMNode;
```

パラメータ	IN / OUT	説明
nnm	IN	DOM 名前付きノード・マップ。
arg	IN	nodeName 属性を使用して追加するノード。

removeNamedItem()

説明

名前で指定されたノードを削除します。このマッピングが要素に連結された属性を含む場合、削除された属性の値がデフォルトであることがわかるときは、属性は、必要に応じて、対応する名前空間 URI、ローカル名、接頭辞およびデフォルト値とともにすぐに表示されます。

構文

```
FUNCTION removeNamedItem( nnm DOMNamedNodeMap,
                          name IN VARCHAR2)
                          RETURN DOMNode;
```

パラメータ	IN / OUT	説明
nnm	IN	DOM 名前付きノード・マップ。
name	IN	マッピングから削除する項目の名前。

item()

説明

index パラメータに対応する、マッピング内の項目を戻します。index がこのマッピング内のノード数以上である場合は、null を戻します。

構文

```
FUNCTION item( nnm DOMNamedNodeMap,
               index IN NUMBER)
               RETURN DOMNode;
```

パラメータ	IN / OUT	説明
nnm	IN	DOM 名前付きノード・マップ。
index	IN	取得する項目が存在する、ノード・マップ内のインデックス。

getLength()

説明

このマッピング内のノード数を戻します。有効な子ノードのインデックスの範囲は、0 ～ length-1（両方の数値を含む）です。

構文

```
FUNCTION getLength( nnm DOMNamedNodeMap) RETURN NUMBER;
```

パラメータ	IN / OUT	説明
nnm	IN	DOM 名前付きノード・マップ。

DOM ノード・リスト・メソッド

isNull()

説明

指定された DOM ノード・リストが null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( nl DOMNodeList) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
nl	(IN)	確認する DOM ノード・リスト。

item()

説明

index パラメータに対応する、コレクション内の項目を戻します。index がリスト内のノード数以上である場合は、null を戻します。

構文

```
FUNCTION item( nl DOMNodeList,
               index IN NUMBER)
               RETURN DOMNode;
```

パラメータ	IN / OUT	説明
nl	IN	DOM ノード・リスト。
index	IN	取得する項目が存在する、NodeList 内のインデックス。

getLength()

説明

リスト内のノード数を戻します。有効な子ノードのインデックスの範囲は、0 ～ length-1（両方の数値を含む）です。

構文

```
FUNCTION getLength( nl DOMNodeList) RETURN NUMBER;
```

パラメータ	IN / OUT	説明
nl	IN	DOM ノード・リスト。

DOM 属性メソッド

isNull()

説明

指定された DOM 属性が null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( a DOMAttr) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
a	(IN)	確認する DOM 属性。

makeNode()

説明

指定された DOM 属性を DOM ノードにキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( a DOMAttr) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
a	(IN)	キャストする DOM 属性。

getQualifiedName()

説明

DOM 属性の修飾名を戻します。

構文

```
FUNCTION getQualifiedName( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	(IN)	DOM 属性。

getNamespace()

説明

DOM 属性の名前空間を戻します。

構文

```
FUNCTION getNamespace( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	(IN)	DOM 属性。

getLocalName()

説明
DOM 属性のローカル名を戻します。

構文
```
FUNCTION getLocalName( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	(IN)	DOM 属性。

getExpandedName()

説明
DOM 属性の拡張名を戻します。

構文
```
FUNCTION getExpandedName( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	(IN)	DOM 属性。

getName()

説明
この属性の名前を戻します。

構文
```
FUNCTION getName( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	IN	DOM 属性。

getSpecified()

説明

元のドキュメントでこの属性に値が明示的に指定されている場合は true、それ以外の場合は false を戻します。

構文

```
FUNCTION getSpecified( a DOMAttr) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
a	IN	DOM 属性。

getValue()

説明

属性値を取得します。

構文

```
FUNCTION getValue( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	IN	DOM 属性。

setValue()

説明

属性値を設定します。

構文

```
PROCEDURE setValue( a DOMAttr,
                    value IN VARCHAR2);
```

パラメータ	IN / OUT	説明
a	IN	DOM 属性。
value	IN	属性に設定する値。

DOM CDATA セクション・メソッド

isNull()

説明

指定された DOM CDATA セクションが null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( cds DOMCDataSection) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
cds	(IN)	確認する DOM CDATA セクション。

makeNode()

説明

DOM CDATA セクションを DOM ノードにキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( cds DOMCDataSection) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
cds	(IN)	キャストする DOM CDATA セクション。

DOM 文字データ・メソッド

isNull()

説明

指定された DOM 文字データが null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( cd DOMCharacterData) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
cd	(IN)	確認する DOM 文字データ。

makeNode()

説明

指定された DOM 文字データを DOM ノードとしてキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( cd DOMCharacterData) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
cd	(IN)	キャストする DOM 文字データ。

getData()

説明

このインタフェースを実装するノードの文字データを取得します。

構文

```
FUNCTION getData( cd DOMCharacterData) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
cd	IN	DOM 文字データ。

setData()

説明

このインタフェースを実装するノードの文字データを設定します。

構文

```
PROCEDURE setData( cd DOMCharacterData,
                   data IN VARCHAR2);
```

パラメータ	IN / OUT	説明
cd	IN	DOM 文字データ。
data	IN	ノードに設定するデータ。

getLength()

説明

データおよび substringData() メソッドを介して使用できる 16 ビット単位の数を戻します。この値は 0（ゼロ）である場合があり、文字データ・ノードは空の場合があります。

構文

```
FUNCTION getLength( cd DOMCharacterData) RETURN NUMBER;
```

パラメータ	IN / OUT	説明
cd	IN	DOM 文字データ。

substringData()

説明

一定の範囲のデータをノードから抽出します。

構文

```
FUNCTION substringData( cd DOMCharacterData,
                        offset IN NUMBER,
                        cnt IN NUMBER)
                        RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
cd	IN	DOM 文字データ。
offset	IN	データ取得の開始点とするオフセット。
cnt	IN	取得するデータの（オフセットからの）文字数。

appendData()

説明

文字列をノードの文字データの最後に追加します。正常に追加されると、連続したデータと指定された文字列引数へのアクセスが可能になります。

構文

```
PROCEDURE appendData( cd DOMCharacterData,
                      arg IN VARCHAR2);
```

パラメータ	IN / OUT	説明
cd	IN	DOM 文字データ。
arg	IN	既存のデータに追加するデータ。

insertData()

説明

指定された 16 ビット単位のオフセットに文字列を挿入します。

構文

```
PROCEDURE insertData( cd DOMCharacterData,
                      offset IN NUMBER,
                      arg IN VARCHAR2);
```

パラメータ	IN / OUT	説明
cd	IN	DOM 文字データ。
offset	IN	データの挿入点とするオフセット。
arg	IN	挿入する値。

deleteData()

説明

一定の範囲の 16 ビット単位をノードから削除します。正常に削除されると、その変更がデータおよび長さに反映されます。

構文

```
PROCEDURE deleteData( cd DOMCharacterData,
                      offset IN NUMBER,
                      cnt IN NUMBER);
```

パラメータ	IN / OUT	説明
cd	IN	DOM 文字データ。
offset	IN	データ削除の開始点とするオフセット。
cnt	IN	削除する（オフセットからの）文字数。

replaceData()

説明

一定の範囲の 16 ビット単位をノードから置換します。正常に置換されると、その変更がデータおよび長さに反映されます。

構文

```
PROCEDURE replaceData( cd DOMCharacterData,
                       offset IN NUMBER,
                       cnt IN NUMBER,
                       arg IN VARCHAR2);
```

パラメータ	IN / OUT	説明
cd	IN	DOM 文字データ。
offset	IN	置換の開始点とするオフセット。
cnt	IN	置換する文字数。
arg	IN	置換後の値。

DOM コメント・メソッド

isNull()

説明

指定された DOM コメントが null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( com DOMComment) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
com	(IN)	確認する DOM コメント。

makeNode()

説明
指定された DOM コメントを DOM ノードにキャストし、その DOM ノードを戻します。

構文
```
FUNCTION makeNode( com DOMComment) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
com	(IN)	キャストする DOM コメント。

DOM インプリメンテーション・メソッド

isNull()

説明
指定された DOM インプリメンテーションが null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文
```
FUNCTION isNull( di DOMImplementation) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
di	(IN)	確認する DOM インプリメンテーション。

hasFeature()

説明

DOM インプリメンテーションが固有の機能を実装するかどうかをテストします。

構文

```
FUNCTION hasFeature( di DOMImplementation,
                     feature IN VARCHAR2,
                     version IN VARCHAR2)
                     RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
di	IN	DOM インプリメンテーション。
feature	IN	確認する機能。
version	IN	確認する DOM のバージョン。

DOM ドキュメント・フラグメント・メソッド

isNull()

説明

指定された DOM ドキュメント・フラグメントが null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( df DOMDocumentFragment) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
df	(IN)	確認する DOM ドキュメント・フラグメント。

makeNode()

説明

指定された DOM ドキュメント・フラグメントを DOM ノードにキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( df DOMDocumentFragment) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
df	(IN)	キャストする DOM ドキュメント・フラグメント。

DOM ドキュメント・タイプ・メソッド

isNull()

説明

指定された DOM ドキュメント・タイプが null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( dt DOMDocumentType) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
dt	(IN)	確認する DOM ドキュメント・タイプ。

makeNode()

説明

指定された DOM ドキュメント・タイプを DOM ノードにキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( dt DOMDocumentType) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
dt	(IN)	キャストする DOM ドキュメント・タイプ。

findEntity()

説明

指定された DTD のエンティティを検索し、検出された場合は、そのエンティティを戻します。

構文

```
FUNCTION findEntity( dt      DOMDocumentType,
                     name    VARCHAR2,
                     par     BOOLEAN)
                     RETURN  DOMEntity;
```

パラメータ	IN / OUT	説明
dt	(IN)	DTD。
name	(IN)	検索するエンティティ。
par	(IN)	エンティティの種類を示すフラグ。パラメータ・エンティティの場合は `true`、通常のエンティティの場合は `false` です。

findNotation()

説明

指定された DTD の表記法を検索し、検出された場合は、その表記法を戻します。

構文

```
FUNCTION findNotation( dt   DOMDocumentType,
                       name VARCHAR2)
                       RETURN DOMNotation;
```

パラメータ	IN / OUT	説明
dt	(IN)	DTD。
name	(IN)	検索する表記法。

getPublicId()

説明

指定された DTD の公開識別子を戻します。

構文

```
FUNCTION getPublicId( dt DOMDocumentType) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
dt	(IN)	DTD。

getSystemId()

説明

指定された DTD のシステム識別子を戻します。

構文

```
FUNCTION getSystemId( dt DOMDocumentType) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
dt	(IN)	DTD。

writeExternalDTDToFile()

説明

指定されたファイルに DTD を書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2);	データベース・キャラクタ・セットを使用して、指定されたファイルに DTD を書き込みます。
PROCEDURE writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2, charset VARCHAR2);	指定されたキャラクタ・セットを使用して、指定されたファイルに DTD を書き込みます。

パラメータ	IN / OUT	説明
dt	(IN)	DTD。
fileName	(IN)	書込み先のファイル。
charset	(IN)	キャラクタ・セット。

writeExternalDTDToBuffer()

説明

指定されたバッファに DTD を書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2);	データベース・キャラクタ・セットを使用して、指定されたバッファに DTD を書き込みます。
PROCEDURE writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2, charset VARCHAR2);	指定されたキャラクタ・セットを使用して、指定されたバッファに DTD を書き込みます。

パラメータ	IN / OUT	説明
dt	(IN)	DTD。
buffer	(IN/OUT)	書込み先のバッファ。
charset	(IN)	キャラクタ・セット。

writeExternalDTDToClob()

説明

指定された CLOB に DTD を書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB);	データベース・キャラクタ・セットを使用して、指定された CLOB に DTD を書き込みます。
PROCEDURE writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB, charset VARCHAR2);	指定されたキャラクタ・セットを使用して、指定された CLOB に DTD を書き込みます。

パラメータ	IN / OUT	説明
dt	(IN)	DTD。
cl	(IN/OUT)	書込み先の CLOB。
charset	(IN)	キャラクタ・セット。

getName()

説明

DTD の名前（DOCTYPE キーワードの直後の名前）を取得します。

構文

```
FUNCTION getName( dt DOMDocumentType) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
dt	IN	DOM ドキュメント・タイプ。

getEntities()

説明

DTD で宣言されている外部および内部の汎用エンティティを含む名前付きノード・マップを取得します。

構文

```
FUNCTION getEntities( dt DOMDocumentType) RETURN DOMNamedNodeMap;
```

パラメータ	IN / OUT	説明
dt	IN	DOM ドキュメント・タイプ。

getNotations()

説明

DTD で宣言されている表記法を含む名前付きノード・マップを取得します。

構文

```
FUNCTION getNotations( dt DOMDocumentType) RETURN DOMNamedNodeMap;
```

パラメータ	IN / OUT	説明
dt	IN	DOM ドキュメント・タイプ。

DOM 要素メソッド

isNull()

説明

指定された DOM 要素が null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( elem DOMElement) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
elem	(IN)	確認する DOM 要素。

makeNode()

説明

指定された DOM 要素を DOM ノードにキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( elem DOMElement) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
elem	(IN)	キャストする DOM 要素。

getQualifiedName()

説明

DOM 要素の修飾名を戻します。

構文

```
FUNCTION getQualifiedName( elem DOMElement) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。

getNamespace()

説明

DOM 要素の名前空間を戻します。

構文

```
FUNCTION getNamespace( elem DOMElement) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。

getLocalName()

説明

DOM 要素のローカル名を戻します。

構文

```
FUNCTION getLocalName( elem DOMElement) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。

getExpandedName()

説明

DOM 要素の拡張名を戻します。

構文

```
FUNCTION getExpandedName( elem DOMElement) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。

getChildrenByTagName()

説明

DOM 要素の子を戻します。次の表に、オプションを示します。

構文	説明
FUNCTION getChildrenByTagName(elem DOMElement, name VARCHAR2) RETURN DOMNodeList;	指定されたタグ名を持つ DOM 要素の子を戻します。
FUNCTION getChildrenByTagName(elem DOMElement, name VARCHAR2, ns VARCHAR2) RETURN DOMNodeList;	指定されたタグ名および名前空間を持つ DOM 要素の子を戻します。

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。
name	(IN)	タグ名。「*」を指定すると、すべてのタグに一致します。
ns	(IN)	名前空間。

getElementsByTagName()

説明

DOM 要素の子要素を戻します。次の表に、オプションを示します。

構文	説明
FUNCTION getElementsByTagName(elem DOMElement, name IN VARCHAR2) RETURN DOMNodeList;	指定されたタグ名を持つ DOM 要素の子要素を戻します。
FUNCTION getElementsByTagName(elem DOMElement, name VARCHAR2, ns VARCHAR2) RETURN DOMNodeList;	指定されたタグ名および名前空間を持つ DOM 要素の子要素を戻します。

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。
name	(IN)	タグ名。「*」を指定すると、すべてのタグに一致します。
ns	(IN)	名前空間。

resolveNamespacePrefix()

説明

指定された名前空間の接頭辞を解決し、解決済名前空間を戻します。

構文

```
FUNCTION resolveNamespacePrefix( elem DOMElement,
                                 prefix VARCHAR2)
                                 RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。
prefix	(IN)	名前空間の接頭辞。

getTagName()

説明

DOM 要素の名前を戻します。

構文

```
FUNCTION getTagName(elem DOMElement) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。

getAttribute()

説明

属性名を持つ、DOM 要素の任意の属性値を戻します。

構文

```
FUNCTION getAttribute( elem DOMElement,
                       name IN VARCHAR2)
                       RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。
name	(IN)	属性名。「*」を指定すると、すべての属性に一致します。

setAttribute()

説明

属性名を持つ、DOM 要素の任意の属性値を設定します。

構文

```
PROCEDURE setAttribute( elem DOMElement,
                        name IN VARCHAR2,
                        value IN VARCHAR2);
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。
name	(IN)	属性名。「*」を指定すると、すべての属性に一致します。
value	(IN)	属性値。

removeAttribute()

説明

DOM 要素から属性名を持つ任意の属性を削除します。

構文

```
PROCEDURE removeAttribute( elem DOMElement,
                           name IN VARCHAR2);
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。
name	(IN)	属性名。「*」を指定すると、すべての属性に一致します。

getAttributeNode()

説明

属性名を持つ、DOM 要素の任意の属性ノードを戻します。

構文

```
FUNCTION getAttributeNode( elem DOMElement,
                           name IN VARCHAR2)
                           RETURN DOMAttr;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。
name	(IN)	属性名。「*」を指定すると、すべての属性に一致します。

setAttributeNode()

説明

DOM 要素に新しい属性ノードを追加します。

構文

```
FUNCTION setAttributeNode( elem DOMElement,
                           newAttr IN DOMAttr)
                           RETURN DOMAttr;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。
newAttr	(IN)	新しい DOM 属性。

removeAttributeNode()

説明

指定された属性ノードを DOM 要素から削除します。

構文

```
FUNCTION removeAttributeNode( elem DOMElement,
                              oldAttr IN DOMAttr)
                              RETURN DOMAttr;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。
oldAttr	(IN)	古い DOM 属性。

normalize()

説明

DOM 要素の子である Text ノードを正規化します。

構文

```
PROCEDURE normalize( elem DOMElement);
```

パラメータ	IN / OUT	説明
elem	(IN)	DOM 要素。

DOM エンティティ・メソッド

isNull()

説明

指定された DOM エンティティが null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( ent DOMEntity) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
ent	(IN)	確認する DOM エンティティ。

makeNode()

説明

指定された DOM エンティティを DOM ノードにキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( ent DOMEntity) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
ent	(IN)	キャストする DOM エンティティ。

getPublicId()

説明

DOM エンティティの公開識別子を戻します。

構文

```
FUNCTION getPublicId( ent DOMEntity) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
ent	(IN)	DOM エンティティ。

getSystemId()

説明

DOM エンティティのシステム識別子を戻します。

構文

```
FUNCTION getSystemId( ent DOMEntity) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
ent	(IN)	DOM エンティティ。

getNotationName()

説明

DOM エンティティの表記法名を戻します。

構文

```
FUNCTION getNotationName( ent DOMEntity) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
ent	(IN)	DOM エンティティ。

DOM 実体参照メソッド

isNull()

説明

指定された DOM 実体参照が null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( eref DOMEntityReference) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
eref	(IN)	確認する DOM 実体参照。

makeNode()

説明

DOM 実体参照を DOM ノードにキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( eref DOMEntityReference) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
eref	(IN)	キャストする DOM 実体参照。

DOM 表記法メソッド

isNull()

説明

指定された DOM 表記法が null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( n DOMNotation) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
n	(IN)	確認する DOM 表記法。

makeNode()

説明

DOM 表記法を DOM ノードにキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( n DOMNotation) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	(IN)	キャストする DOM 表記法。

getPublicId()

説明

DOM 表記法の公開識別子を戻します。

構文

```
FUNCTION getPublicId( n DOMNotation) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
n	(IN)	DOM 表記法。

getSystemId()

説明

DOM 表記法のシステム識別子を戻します。

構文

```
FUNCTION getSystemId( n DOMNotation) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
n	(IN)	DOM 表記法。

DOM 処理命令メソッド

isNull()

説明

指定された DOM 処理命令が null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( pi DOMProcessingInstruction) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
pi	(IN)	確認する DOM 処理命令。

makeNode()

説明

DOM 処理命令を DOM ノードにキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( pi DOMProcessingInstruction) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
pi	(IN)	キャストする DOM 処理命令。

getData()

説明

DOM 処理命令のコンテンツ・データを戻します。

構文

```
FUNCTION  getData( pi DOMProcessingInstruction) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
pi	(IN)	DOM 処理命令。

getTarget()

説明

DOM 処理命令のターゲットを戻します。

構文

```
FUNCTION getTarget( pi DOMProcessingInstruction) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
pi	(IN)	DOM 処理命令。

setData()

説明
DOM 処理命令のコンテンツ・データを設定します。

構文
```
PROCEDURE setData( pi DOMProcessingInstruction,
                   data IN VARCHAR2);
```

パラメータ	IN / OUT	説明
pi	(IN)	DOM 処理命令。
data	(IN)	新しい処理命令のコンテンツ・データ

DOM テキスト・メソッド

isNull()

説明
指定された DOM テキストが null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文
```
FUNCTION isNull( t DOMText) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
t	(IN)	確認する DOM テキスト。

makeNode()

説明

DOM テキストを DOM ノードにキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( t DOMText) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
t	(IN)	キャストする DOM テキスト。

splitText()

説明

この DOMText ノードを、指定されたオフセットで 2 つの DOMText ノードに分割します。

構文

```
FUNCTION splitText( t DOMText,
                    offset IN NUMBER)
                    RETURN DOMText;
```

パラメータ	IN / OUT	説明
t	(IN)	DOM テキスト。
offset	(IN)	分割点となるオフセット。

DOM 文書メソッド

isNull()

説明

指定された DOM 文書が null かどうかを確認し、null の場合は true、null でない場合は false を戻します。

構文

```
FUNCTION isNull( doc DOMDocument) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
doc	(IN)	確認する DOM 文書。

makeNode()

説明

DOM 文書を DOM ノードにキャストし、その DOM ノードを戻します。

構文

```
FUNCTION makeNode( doc DOMDocument) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
doc	(IN)	キャストする DOM 文書。

newDOMDocument()

説明

新しい DOM 文書インスタンスを戻します。

構文

```
FUNCTION newDOMDocument RETURN DOMDocument;
```

freeDocument()

説明

DOMDocument オブジェクトを解放します。

構文

```
PROCEDURE freeDocument( doc DOMDocument);
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。

getVersion()

説明

XML 文書のバージョン情報を戻します。

構文

```
FUNCTION getVersion( doc DOMDocument) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。

setVersion()

説明

XML 文書のバージョン情報を設定します。

構文

```
PROCEDURE setVersion( doc     DOMDocument,
                      version VARCHAR2);
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
version	(IN)	バージョン情報。

getCharset()

説明

XML 文書のキャラクタ・セットを取得します。

構文

```
FUNCTION getCharset( doc DOMDocument) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。

setCharset()

説明

XML 文書のキャラクタ・セットを設定します。

構文

```
PROCEDURE setCharset( doc DOMDocument,
                      charset VARCHAR2);
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
charset	(IN)	キャラクタ・セット。

getStandalone()

説明

XML 文書のスタンドアロン情報を戻します。

構文

```
FUNCTION getStandalone( doc DOMDocument) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。

setStandalone()

説明

XML 文書のスタンドアロン情報を設定します。

構文

```
PROCEDURE setStandalone( doc DOMDocument,
                         value VARCHAR2);
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
value	(IN)	スタンドアロン情報。

writeToFile()

説明

指定されたファイルに XML 文書を書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeToFile(doc DOMDocument, fileName VARCHAR2);	データベース・キャラクタ・セットを使用して、指定されたファイルに XML 文書を書き込みます。
PROCEDURE writeToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2);	指定されたキャラクタ・セットを使用して、指定されたファイルに XML 文書を書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
filename	(IN)	書込み先のファイル。
charset	(IN)	キャラクタ・セット。

writeToBuffer()

説明

指定されたバッファに XML 文書を書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2);	データベース・キャラクタ・セットを使用して、指定されたバッファに XML 文書を書き込みます。
PROCEDURE writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2);	指定されたキャラクタ・セットを使用して、指定されたバッファに XML 文書を書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
buffer	(IN/OUT)	書込み先のバッファ。
charset	(IN)	キャラクタ・セット。

writeToClob()

説明

指定された CLOB に XML 文書を書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeToClob(doc DOMDocument, cl IN OUT CLOB);	データベース・キャラクタ・セットを使用して、指定された CLOB に XML 文書を書き込みます。
PROCEDURE writeToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2);	指定されたキャラクタ・セットを使用して、指定された CLOB に XML 文書を書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
cl	(IN/OUT)	書込み先の CLOB。
charset	(IN)	キャラクタ・セット。

writeExternalDTDToFile()

説明

指定されたファイルに外部 DTD を書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2);	データベース・キャラクタ・セットを使用して、指定されたファイルに外部 DTD を書き込みます。
PROCEDURE writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2);	指定されたキャラクタ・セットを使用して、指定されたファイルに外部 DTD を書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
fileName	(IN)	書込み先のファイル。
charset	(IN)	キャラクタ・セット。

writeExternalDTDToBuffer()

説明

指定されたバッファに外部 DTD を書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2);	データベース・キャラクタ・セットを使用して、指定されたバッファに外部 DTD を書き込みます。
PROCEDURE writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2);	指定されたキャラクタ・セットを使用して、指定されたバッファに外部 DTD を書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
buffer	(IN/OUT)	書込み先のバッファ。
charset	(IN)	キャラクタ・セット。

writeExternalDTDToClob()

説明

指定された CLOB に外部 DTD を書き込みます。次の表に、オプションを示します。

構文	説明
PROCEDURE writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB);	データベース・キャラクタ・セットを使用して、指定された CLOB に外部 DTD を書き込みます。
PROCEDURE writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2);	指定されたキャラクタ・セットを使用して、指定された CLOB に外部 DTD を書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
cl	(IN)	書込み先の CLOB。
charset	(IN)	キャラクタ・セット。

getDoctype()

説明

DOM 文書に対応する DTD を戻します。

構文

```
FUNCTION getDoctype( doc DOMDocument) RETURN DOMDocumentType;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。

getImplementation()

説明

この DOM 文書を処理する DOM インプリメンテーション・オブジェクトを戻します。

構文

```
FUNCTION getImplementation( doc DOMDocument) RETURN DOMImplementation;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。

getDocumentElement()

説明

DOM 文書の子ノード（ドキュメント要素）を戻します。

構文

```
FUNCTION getDocumentElement( doc DOMDocument) RETURN DOMElement;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。

createElement()

説明

DOM 要素を作成します。

構文

```
FUNCTION createElement( doc DOMDocument,
                        tagName IN VARCHAR2)
                        RETURN DOMElement;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
tagName	(IN)	新しい DOM 要素のタグ名。

createDocumentFragment()

説明

DOM ドキュメント・フラグメントを作成します。

構文

```
FUNCTION createDocumentFragment( doc DOMDocument) RETURN DOMDocumentFragment;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。

createTextNode()

説明

DOMText ノードを作成します。

構文

```
FUNCTION createTextNode( doc DOMDocument,
                         data IN VARCHAR2)
                         RETURN DOMText;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
data	(IN)	DOMText ノードのコンテンツ。

createComment()

説明

DOM コメント・ノードを作成します。

構文

```
FUNCTION createComment( doc DOMDocument,
                        data IN VARCHAR2)
                        RETURN DOMComment;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
data	(IN)	DOM コメント・ノードのコンテンツ。

createCDATASection()

説明

DOM CDATA セクション・ノードを作成します。

構文

```
FUNCTION createCDATASection( doc DOMDocument,
                             data IN VARCHAR2)
                             RETURN DOMCDATASection;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
data	(IN)	DOM CDATA セクション・ノードのコンテンツ。

createProcessingInstruction()

説明

DOM 処理命令ノードを作成します。

構文

```
FUNCTION createProcessingInstruction( doc DOMDocument,
                                      target IN VARCHAR2,
                                      data IN VARCHAR2)
                                      RETURN DOMProcessingInstruction;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
target	(IN)	新しい処理命令のターゲット。
data	(IN)	新しい処理命令のコンテンツ・データ。

createAttribute()

説明

DOM Attr ノードを作成します。

構文

```
FUNCTION createAttribute( doc DOMDocument,
                          name IN VARCHAR2)
                          RETURN DOMAttr;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
name	(IN)	新しい属性名。

createEntityReference()

説明

DOM 実体参照ノードを作成します。

構文

```
FUNCTION createEntityReference( doc DOMDocument,
                                name IN VARCHAR2)
                                RETURN DOMEntityReference;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
name	(IN)	新しい実体参照名。

getElementsByTagName()

説明

指定されたタグ名を持つすべての要素の DOM NodeList を戻します。

構文

```
FUNCTION getElementsByTagName( doc DOMDocument,
                               tagname IN VARCHAR2)
                               RETURN DOMNodeList;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOM 文書。
tagname	(IN)	一致させるタグ名。

26

PL/SQL Parser API for XMLType

DBMS_XMLPARSER パッケージの API を介して、XML 文書のコンテンツおよび構造にアクセスできます。

この章の内容は次のとおりです。

- DBMS_XMLPARSER の説明
- DBMS_XMLPARSER のサブプログラム

> **参照：** 次のマニュアルを参照してください。
>
> - 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
> - 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

DBMS_XMLPARSER パッケージ

DBMS_XMLPARSER の説明

eXtensible Markup Language（XML）は、XML 文書というデータ・オブジェクトのクラスを記述します。XML は、XML 文書を処理するコンピュータ・プログラムの動作を部分的に記述します。XML はアプリケーション・プロファイルであり、Standard Generalized Markup Language（SGML）の機能を一部制限したものです。構成的には、XML 文書は SGML ドキュメントに準拠します。

XML 文書は、エンティティという記憶単位で構成されます。エンティティには、解析対象データまたは解析対象外データのいずれかが格納されます。解析対象データは文字で構成され、文字データやマークアップを形成します。マークアップは、ドキュメントの記憶レイアウトおよび論理構造の記述をエンコードします。XML は、記憶レイアウトおよび論理構造に制約を適用するためのメカニズムを提供します。

XML 文書の読込み、およびそのコンテンツと構造へのアクセスには、XML プロセッサというソフトウェア・モジュールを使用します。XML プロセッサは、別のモジュール（アプリケーション）にかわって作業を行っていることを想定しています。この XML プロセッサ（またはパーサー）の PL/SQL 実装は、W3C の XML 仕様（改訂 REC-xml-19980210）に準拠し、XML データを読み込むために XML プロセッサに必要な動作、およびアプリケーションに提供する必要がある情報を含んでいます。

この PL/SQL XML パーサーのデフォルトの動作を次に示します。

- DOM API がアクセス可能な 1 つの解析ツリーが構築されます。
- パーサーは、DTD が検出されると検証を行い、検出されないと検証を行いません。
- エラー・ログを指定しなかった場合、エラーは記録されませんが、解析が正常に実行されなかった場合は、アプリケーション・エラーが発生します。

DBMS_XMLPARSER のサブプログラム

表 26-1 DBMS_XMLPARSER のサブプログラムの概要

サブプログラム	説明
parse()（26-4 ページ）	指定された URL/ ファイルに格納された XML を解析します。
newParser()（26-4 ページ）	新しいパーサー・インスタンスを戻します。
parseBuffer()（26-5 ページ）	指定されたバッファに格納された XML を解析します。
parseClob()（26-5 ページ）	指定された CLOB に格納された XML を解析します。
parseDTD()（26-6 ページ）	指定された URL/ ファイルに格納された DTD を解析します。
parseDTDBuffer()（26-6 ページ）	指定されたバッファに格納された DTD を解析します。
parseDTDClob()（26-7 ページ）	指定された CLOB に格納された DTD を解析します。
setBaseDir()（26-7 ページ）	関連 URL を解決するために使用するベース・ディレクトリを設定します。
showWarnings()（26-8 ページ）	警告を表示または非表示にします。
setErrorLog()（26-8 ページ）	指定されたファイルにエラーが送信されるように設定します。
setPreserveWhitespace()（26-9 ページ）	空白保持モードを設定します。
setValidationMode()（26-9 ページ）	検証モードを設定します。
getValidationMode()（26-10 ページ）	検証モードを戻します。
setDoctype()（26-10 ページ）	DTD を設定します。
getDoctype()（26-11 ページ）	DTD を取得します。
getDocument()（26-11 ページ）	DOM 文書を取得します。
freeParser()（26-12 ページ）	パーサー・オブジェクトを解放します。
getReleaseVersion()（26-12 ページ）	Oracle XML Parser for PL/SQL のバージョン番号を戻します。

parse()

説明

指定された URL/ ファイルに格納された XML を解析します。解析が正常に実行されなかった場合は、アプリケーション・エラーが発生します。次の表に、オプションを示します。

構文	説明
FUNCTION parse(url VARCHAR2) RETURN DOMDocument;	構築された DOM 文書を戻します。パーサーのデフォルトの動作が受け入れられ、URL/ ファイルのみを解析する必要がある場合に使用します。
PROCEDURE parse(p Parser, url VARCHAR2);	パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。

パラメータ	IN / OUT	説明
url	(IN)	解析する URL/ ファイルの完全なパス。
p	(IN)	パーサー・インスタンス。

newParser()

説明

新しいパーサー・インスタンスを戻します。パーサーのデフォルトの動作が変更される前で、他の解析メソッドを使用する必要がある場合にコールする必要があります。

構文

```
FUNCTION newParser RETURN Parser;
```

parseBuffer()

説明

指定されたバッファに格納された XML を解析します。パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。解析が正常に実行されなかった場合は、アプリケーション・エラーが発生します。

構文

```
PROCEDURE parseBuffer( p   Parser,
                       doc VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
doc	(IN)	解析する XML 文書バッファ。

parseClob()

説明

指定された CLOB に格納された XML を解析します。パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。解析が正常に実行されなかった場合は、アプリケーション・エラーが発生します。

構文

```
PROCEDURE parseClob( p   Parser,
                     doc CLOB);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
doc	(IN)	解析する XML 文書バッファ。

parseDTD()

説明

指定された URL/ ファイルに格納された DTD を解析します。パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。解析が正常に実行されなかった場合は、アプリケーション・エラーが発生します。

構文

```
PROCEDURE parseDTD( p     Parser,
                    url   VARCHAR2,
                    root  VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
url	(IN)	解析する URL/ ファイルの完全なパス。
root	(IN)	ルート要素名。

parseDTDBuffer()

説明

指定されたバッファに格納された DTD を解析します。パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。解析が正常に実行されなかった場合は、アプリケーション・エラーが発生します。

構文

```
PROCEDURE parseDTDBuffer( p    Parser,
                          dtd  VARCHAR2,
                          root VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
dtd	(IN)	解析する DTD バッファ。
root	(IN)	ルート要素名。

parseDTDClob()

説明

指定された CLOB に格納された DTD を解析します。パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。解析が正常に実行されなかった場合は、アプリケーション・エラーが発生します。

構文

```
PROCEDURE parseDTDClob( p    Parser,
                        dtd  CLOB,
                        root VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
dtd	(IN)	解析する DTD CLOB。
root	(IN)	ルート要素名。

setBaseDir()

説明

関連 URL を解決するために使用するベース・ディレクトリを設定します。解析が正常に実行されなかった場合は、アプリケーション・エラーが発生します。

構文

```
PROCEDURE setBaseDir( p   Parser,
                      dir VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
dir	(IN)	ベース・ディレクトリとして使用するディレクトリ。

showWarnings()

説明

警告を表示または非表示にします。

構文

```
PROCEDURE showWarnings( p   Parser,
                        yes BOOLEAN);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
yes	(IN)	設定するモード。警告を表示する場合は `true`、表示しない場合は `false` です。

setErrorLog()

説明

指定されたファイルにエラーが送信されるように設定します。

構文

```
PROCEDURE setErrorLog( p        Parser,
                       fileName VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
fileName	(IN)	エラー・ログとして使用するファイルの完全なパス。

setPreserveWhitespace()

説明

空白保持モードを設定します。

構文

```
PROCEDURE setPreserveWhitespace( p   Parser,
                                 yes BOOLEAN);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
yes	(IN)	設定するモード。空白を保持する場合は true、保持しない場合は false です。

setValidationMode()

説明

検証モードを設定します。

構文

```
PROCEDURE setValidationMode( p   Parser,
                             yes BOOLEAN);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
yes	(IN)	設定するモード。検証する場合は true、検証しない場合は false です。

getValidationMode()

説明

検証モードを取得します。検証する場合は true、検証しない場合は false を戻します。

構文

```
FUNCTION getValidationMode( p Parser)
                            RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。

setDoctype()

説明

パーサーが検証に使用する DTD を設定します。これは、ドキュメントが解析される前にコールする必要があります。

構文

```
PROCEDURE setDoctype( p   Parser,
                      dtd DOMDocumentType);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
dtd	(IN)	設定する DTD。

getDoctype()

説明

解析済 DTD を戻します。このファンクションのコールは、DTD の解析後のみ有効です。

構文

```
FUNCTION getDoctype( p Parser)
                    RETURN DOMDocumentType;
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。

getDocument()

説明

パーサーが構築した DOM ツリー文書のルートを戻します。このファンクションのコールは、ドキュメントの解析後のみ有効です。

構文

```
FUNCTION getDocument( p Parser)
                     RETURN DOMDocument;
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。

freeParser()

説明

パーサー・オブジェクトを解放します。

構文

```
PROCEDURE freeParser( p Parser);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。

getReleaseVersion()

説明

Oracle XML Parser for PL/SQL のバージョン番号を戻します。

構文

```
PROCEDURE getReleaseVersion RETURN VARCHAR2;
```

27

XMLType での PL/SQL XSLT 処理

DBMS_XSLPROCESSOR パッケージの eXtensible Stylesheet Language（XSL）パッケージ・プロセッサ API を使用すると、PL/SQL 開発者は XML 文書のコンテンツおよび構造にアクセスできます。

この章の内容は次のとおりです。

- DBMS_XSLPROCESSOR の説明
- DBMS_XSLPROCESSOR のサブプログラム

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

DBMS_XSLPROCESSOR パッケージ

DBMS_XSLPROCESSOR の説明

eXtensible Stylesheet Language Transformation（XSLT）は、ソース・ツリーを結果ツリーに変換する規則を記述しています。XSLT に記述されている変換をスタイルシートといいます。指定した変換は、スタイルシートに定義されたテンプレートを使用してパターンを対応付けることによって実行されます。テンプレートは、結果ツリーの一部を作成するためにインスタンス化されます。この XSLT プロセッサの PL/SQL 実装は、W3C の XSLT 草案（改訂 WD-xslt-19990813）に準拠しており、XSLT スタイルシートを読み込むために XML プロセッサに必要な動作、および実行する必要がある変換を含んでいます。

この PL/SQL XSLT プロセッサのデフォルトの動作を次に示します。

- DOM API がアクセス可能な 1 つの結果ツリーが構築されます。
- エラー・ログを指定しなかった場合、エラーは記録されませんが、解析が正常に実行されなかった場合は、アプリケーション・エラーが発生します。

DBMS_XSLPROCESSOR のサブプログラム

表 27-1 DBMS_XSLPROCESSOR のサブプログラムの概要

サブプログラム	説明
newProcessor()（27-3 ページ）	新しいプロセッサ・インスタンスを戻します。
processXSL()（27-3 ページ）	入力された XML 文書を変換します。
showWarnings()（27-6 ページ）	警告を表示または非表示にします。
setErrorLog()（27-6 ページ）	指定されたファイルにエラーが送信されるように設定します。
newStylesheet()（27-7 ページ）	指定された入力 URL および参照 URL を使用して、新しいスタイルシートを作成します。
transformNode()（27-8 ページ）	指定されたスタイルシートを使用して、DOM ツリーのノードを変換します。
selectNodes()（27-8 ページ）	DOM ツリーから、指定されたパターンに一致するノードを選択します。
selectSingleNode()（27-9 ページ）	ツリーから、指定されたパターンに一致する最初のノードを選択します。
valueOf()（27-9 ページ）	ツリーから、指定されたパターンに一致する最初のノードの値を取得します。
setParam()（27-10 ページ）	最上位のスタイルシートのパラメータを設定します。

表 27-1 DBMS_XSLPROCESSOR のサブプログラムの概要（続き）

サブプログラム	説明
removeParam()（27-10 ページ）	最上位のスタイルシートのパラメータを削除します。
resetParams()（27-11 ページ）	最上位のスタイルシートのパラメータをリセットします。
freeStylesheet()（27-11 ページ）	スタイルシート・オブジェクトを解放します。
freeProcessor()（27-11 ページ）	プロセッサ・オブジェクトを解放します。

newProcessor()

説明

新しいプロセッサ・インスタンスを戻します。このファンクションは、プロセッサのデフォルト動作を変更する前、および他のプロセッサ・メソッドを使用する必要がある場合にコールする必要があります。

構文

```
FUNCTION newProcessor RETURN Processor;
```

processXSL()

説明

入力された XML 文書を変換します。プロセッサのデフォルト動作への変更が有効になってから、このプロシージャをコールします。処理が正常に実行されなかった場合は、アプリケーション・エラーが発生します。次の表に、オプションを示します。

構文	説明
FUNCTION processXSL(p Processor, ss Stylesheet, xmldoc DOMDocument), RETURN DOMDocumentFragment;	指定された DOM 文書およびスタイルシートを使用して、入力された XML 文書を変換し、結果のドキュメント・フラグメントを戻します。

構文	説明
FUNCTION processXSL(p Processor, ss Stylesheet, url VARCHAR2, RETURN DOMDocumentFragment;	指定された URL としてのドキュメントおよびスタイルシートを使用して、入力された XML 文書を変換し、結果のドキュメント・フラグメントを戻します。
FUNCTION processXSL(p Processor, ss Stylesheet, clb CLOB) RETURN DOMDocumentFragment;	指定された CLOB としてのドキュメントおよびスタイルシートを使用して、入力された XML 文書を変換し、結果のドキュメント・フラグメントを戻します。
PROCEDURE processXSL(p Processor, ss Stylesheet, xmldoc DOMDocument, dir VARCHAR2, fileName VARCHAR2);	指定された DOM 文書およびスタイルシートを使用して、入力された XML 文書を変換し、出力を指定ファイルに書き込みます。
PROCEDURE processXSL(p Processor, ss Stylesheet, url VARCHAR2, dir VARCHAR2, fileName VARCHAR2);	指定された URL およびスタイルシートを使用して、入力された XML 文書を変換し、出力を指定ディレクトリ内の指定ファイルに書き込みます。
PROCEDURE processXSL(p Processor, ss Stylesheet, xmldoc DOMDocument, cl IN OUT CLOB);	指定された DOM 文書およびスタイルシートを使用して、入力された XML 文書を変換し、出力を CLOB に書き込みます。
FUNCTION processXSL(p Processor, ss Stylesheet, xmldf DOMDocumentFragment) RETURN DOMDocumentFragment;	指定された DOM ドキュメント・フラグメントおよびスタイルシートを使用して、入力された XML 文書のフラグメントを変換し、結果のドキュメント・フラグメントを戻します。

構文	説明
PROCEDURE processXSL(p Processor, ss Stylesheet, xmldf DOMDocumentFragment, dir VARCHAR2, fileName VARCHAR2);	指定された DOM ドキュメント・フラグメントおよびスタイルシートを使用して、入力された XML 文書のフラグメントを変換し、出力を指定ディレクトリ内の指定ファイルに書き込みます。
PROCEDURE processXSL(p Processor, ss Stylesheet, xmldf DOMDocumentFragment, buf IN OUT VARCHAR2);	指定された DOM ドキュメント・フラグメントおよびスタイルシートを使用して、入力された XML 文書のフラグメントを変換し、出力をバッファに書き込みます。
PROCEDURE processXSL(p Processor, ss Stylesheet, xmldf DOMDocumentFragment, cl IN OUT CLOB);	指定された DOM ドキュメント・フラグメントおよびスタイルシートを使用して、入力された XML 文書のフラグメントを変換し、出力を CLOB に書き込みます。

パラメータ	IN / OUT	説明
p	(IN)	プロセッサ・インスタンス。
ss	(IN)	スタイルシート・インスタンス。
xmldoc	(IN)	変換する XML 文書。
url	(IN)	変換する情報の URL。
clb	(IN)	変換する情報を含む CLOB。
dir	(IN)	処理出力ファイルの保存先のディレクトリ。
fileName	(IN)	処理出力ファイル。
cl	(IN/OUT)	処理出力の保存先の CLOB。
xmldf	(IN)	変換する XML 文書のフラグメント。

showWarnings()

説明

警告を表示または非表示にします。

構文

```
PROCEDURE showWarnings( p Processor,
                        yes BOOLEAN);
```

パラメータ	IN / OUT	説明
p	(IN)	プロセッサ・インスタンス。
yes	(IN)	設定するモード。警告を表示する場合は `true`、表示しない場合は `false`。

setErrorLog()

説明

指定されたファイルにエラーが送信されるように設定します。

構文

```
PROCEDURE setErrorLog( p Processor,
                       fileName VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	プロセッサ・インスタンス。
fileName	(IN)	エラー・ログとして使用するファイルの完全なパス。

newStylesheet()

説明

新しいスタイルシート・インスタンスを作成し、戻します。次の表に、オプションを示します。

構文	説明
FUNCTION newStylesheet(xmldoc DOMDocument, ref VARCHAR2) RETURN Stylesheet;	指定された DOM 文書および参照 URL を使用して、新しいスタイルシート・インスタンスを作成し、戻します。
FUNCTION newStylesheet(inp VARCHAR2, ref VARCHAR2) RETURN Stylesheet;	指定された入力 URL および参照 URL を使用して、新しいスタイルシート・インスタンスを作成し、戻します。

パラメータ	IN / OUT	説明
xmldoc	(IN)	作成に使用する DOM 文書。
inp	(IN)	作成に使用する入力 URL。
ref	(IN)	参照 URL。

transformNode()

説明

指定されたスタイルシートを使用して DOM ツリーのノードを変換し、変換の結果を DOM ドキュメント・フラグメントとして戻します。

構文

```
FUNCTION transformNode( n DOMNode,
                        ss Stylesheet)
                        RETURN DOMDocumentFragment;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOM ノード。
ss	(IN)	使用するスタイルシート。

selectNodes()

説明

DOM ツリーから、指定されたパターンに一致するノードを選択し、選択の結果を戻します。

構文

```
FUNCTION selectNodes( n DOMNode,
                      pattern VARCHAR2)
                      RETURN DOMNodeList;
```

パラメータ	IN / OUT	説明
n	(IN)	DOM ツリーのルート・ノード。
pattern	(IN)	使用するパターン。

selectSingleNode()

説明

ツリーから、指定されたパターンに一致する最初のノードを選択し、そのノードを戻します。

構文

```
FUNCTION selectSingleNode( n DOMNode,
                           pattern VARCHAR2)
                           RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	(IN)	DOM ツリーのルート・ノード。
pattern	(IN)	使用するパターン。

valueOf()

説明

ツリーから、指定されたパターンに一致する最初のノードの値を取得します。

構文

```
PROCEDURE valueOf( n DOMNode,
                   pattern VARCHAR2,
                   val OUT VARCHAR2);
```

パラメータ	IN / OUT	説明
n	(IN)	DOM ツリーのルート・ノード。
pattern	(IN)	使用するパターン。
val	(OUT)	取得された値。

setParam()

説明

最上位のスタイルシートのパラメータを設定します。パラメータ値は、有効な XPath 式である必要があります。したがって、文字列リテラル値を囲む必要があります。

構文

```
PROCEDURE setParam( ss Stylesheet,
                    name VARCHAR2,
                    value VARCHAR2);
```

パラメータ	IN / OUT	説明
ss	(IN)	スタイルシート。
name	(IN)	パラメータ名。
value	(IN)	パラメータの値。

removeParam()

説明

最上位のスタイルシートのパラメータを削除します。

構文

```
PROCEDURE removeParam( ss Stylesheet,
                       name VARCHAR2);
```

パラメータ	IN / OUT	説明
ss	(IN)	スタイルシート。
name	(IN)	パラメータ名。

resetParams()

説明

最上位のスタイルシートのパラメータをリセットします。

構文

```
PROCEDURE resetParams( ss Stylesheet);
```

パラメータ	IN / OUT	説明
ss	(IN)	スタイルシート。

freeStylesheet()

説明

スタイルシート・オブジェクトを解放します。

構文

```
PROCEDURE freeStylesheet( ss Stylesheet);
```

パラメータ	IN / OUT	説明
ss	(IN)	スタイルシート。

freeProcessor()

説明

プロセッサ・オブジェクトを解放します。

構文

```
PROCEDURE freeProccessor( p Processor);
```

パラメータ	IN / OUT	説明
p	(IN)	プロセッサ。

28

PL/SQL での DBMS_XMLSCHEMA およびカタログ・ビュー

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

DBMS_XMLSCHEMA パッケージ

DBMS_XMLSCHEMA の説明

このパッケージは、Oracle XML DB のインストール中にスクリプト dbmsxsch.sql によって作成されます。このパッケージは、XML Schema を登録および削除するプロシージャを提供します。

DBMS_XMLSCHEMA の定数

表 28-1 DBMS_XMLSCHEMA の定数

定数	説明
DELETE_RESTRICT	CONSTANT NUMBER := 1;
DELETE_INVALIDATE	CONSTANT NUMBER := 2;
DELETE_CASCADE	CONSTANT NUMBER := 3;
DELETE_CASCADE_FORCE	CONSTANT NUMBER := 4;

DBMS_XMLSCHEMA のファンクションおよびプロシージャ

表 28-2 DBMS_XMLSCHEMA のファンクションおよびプロシージャの概要

定数	説明
registerSchema()（28-3 ページ）	Oracle が使用する指定されたスキーマを登録します。このスキーマに準拠するドキュメントをこのスキーマに格納できるようになります。
registerURI()（28-5 ページ）	URI 名によって指定された XMLSchema を登録します。
deleteSchema()（28-7 ページ）	Oracle XML DB からスキーマを削除します。
compileSchema()（28-8 ページ）	登録済の XML Schema を再コンパイルします。これは、無効なスキーマを有効にする場合に役立ちます。
generateSchema()（28-8 ページ）	Oracle オブジェクト型の名前から XML Schema を生成します。

registerSchema()

説明

Oracle XML DB が使用する指定されたスキーマを登録します。次の表に、オプションを示します。

構文	説明
procedure registerSchema(schemaURL IN varchar2, schemaDoc IN VARCHAR2, local IN BOOLEAN := TRUE, genTypes IN BOOLEAN := TRUE, genbean IN BOOLEAN := FALSE, genTables IN BOOLEAN := TRUE, force IN BOOLEAN := FALSE, owner IN VARCHAR2 := null);	VARCHAR2 として指定されたスキーマを登録します。
procedure registerSchema(schemaURL IN varchar2, schemaDoc IN CLOB, local IN BOOLEAN := TRUE, genTypes IN BOOLEAN := TRUE, genbean IN BOOLEAN := FALSE, force IN BOOLEAN := FALSE, owner IN VARCHAR2 := null);	CLOB として指定されたスキーマを登録します。
procedure registerSchema(schemaURL IN varchar2, schemaDoc IN BFILE, local IN BOOLEAN := TRUE, genTypes IN BOOLEAN := TRUE, genbean IN BOOLEAN := FALSE, force IN BOOLEAN := FALSE, owner IN VARCHAR2 := null);	BFILE として指定されたスキーマを登録します。

構文	説明
procedure registerSchema(schemaURL IN varchar2, schemaDoc IN SYS.XMLType, local IN BOOLEAN := TRUE, genTypes IN BOOLEAN := TRUE, genbean IN BOOLEAN := FALSE, force IN BOOLEAN := FALSE, owner IN VARCHAR2 := null);	XMLType として指定されたスキーマを登録します。
procedure registerSchema(schemaURL IN varchar2, schemaDoc IN SYS.URIType, local IN BOOLEAN := TRUE, genTypes IN BOOLEAN := TRUE, genbean IN BOOLEAN := FALSE, force IN BOOLEAN := FALSE, owner IN VARCHAR2 := null);	URIType として指定されたスキーマを登録します。

パラメータ	IN / OUT	説明
schemaURL	(IN)	スキーマ・ドキュメントを一意に識別する URL。この値は、Oracle XML DB 階層内のスキーマ・ドキュメントのパス名を派生させるために使用します。
schemaDoc	(IN)	妥当な XML Schema 文書。
local	(IN)	このスキーマがローカルかグローバルか。 デフォルトでは、すべてのスキーマは `/sys/schemas/<username/...` にローカル・スキーマとして登録されます。 スキーマをグローバル・スキーマとして登録する場合、そのスキーマは `/sys/schemas/PUBLIC/....` に追加されます。 スキーマをグローバル・スキーマとして登録するには、`PUBLIC` ディレクトリに対する書込み権限が必要です。
genTypes	(IN)	スキーマ・コンパイラがオブジェクト型を生成する必要があるかどうか。デフォルトは `true` です。

パラメータ	IN / OUT	説明
genbean	(IN)	このリリースでは、常に false です。
genTables	(IN)	スキーマ・コンパイラがデフォルトの表を生成する必要があるかどうか。デフォルトは true です。
force	(IN)	true に設定されている場合、スキーマ登録によるエラーは発生しません。エラーが存在する場合は、かわりに無効な XML Schema オブジェクトが作成されます。デフォルトは false です。
owner	(IN)	XML Schema オブジェクトを所有するデータベース・ユーザーの名前を指定します。デフォルトでは、スキーマを登録するユーザーがその XML Schema オブジェクトを所有します。このパラメータを使用すると、別のデータベース・ユーザーが所有するように、XML Schema を登録できます。

registerURI()

説明

URI 名によって指定された XML Schema を登録します。

構文

```
procedure registerURI(schemaURL IN varchar2,
                      schemaDocURI IN varchar2,
                      local IN BOOLEAN := TRUE,
                      genTypes IN BOOLEAN := TRUE,
                      genbean IN BOOLEAN := FALSE,
                      genTables IN BOOLEAN := TRUE,
                      force IN BOOLEAN := FALSE,
                      owner IN VARCHAR2 := null);
```

パラメータ	IN / OUT	説明
schemaURL	(IN)	スキーマ・ドキュメントを一意に識別する名前。
schemaDocURI	(IN)	スキーマ・ドキュメントの物理的な位置に対応するパス名（URI）。URI パスは、HTTP、FTP、DB または Oracle XML DB プロトコルに基づいている場合があります。このファンクションは、URIFactory を使用して URIType インスタンスを作成し、`registerSchema()` ファンクションをコールします。
local	(IN)	このスキーマがローカルかグローバルか。 デフォルトでは、すべてのスキーマは `/sys/schemas/<username/...` にローカル・スキーマとして登録されます。 スキーマをローバル・スキーマとして登録する場合、そのスキーマは `/sys/schemas/PUBLIC/....` に追加されます。 スキーマをグローバル・スキーマとして登録するには、`PUBLIC` ディレクトリに対する書込み権限が必要です。
genTypes	(IN)	スキーマ・コンパイラがオブジェクト型を生成する必要があるかどうか。デフォルトは `true` です。
genbean	(IN)	このリリースでは、常に `false` です。
genTables	(IN)	スキーマ・コンパイラがデフォルトの表を生成する必要があるかどうか。デフォルトは `true` です。
force	(IN)	`true` に設定されている場合、スキーマ登録によるエラーは発生しません。エラーが存在する場合は、かわりに無効な XML Schema オブジェクトが作成されます。デフォルトは `false` です。
owner	(IN)	XML Schema オブジェクトを所有するデータベース・ユーザーの名前を指定します。デフォルトでは、スキーマを登録するユーザーがその XML Schema オブジェクトを所有します。このパラメータを使用すると、別のデータベース・ユーザーが所有するように、XML Schema を登録することができます。

deleteSchema()

説明

URL によって指定された XMLSchema を削除します。「ORA-31001 リソース・ハンドルまたはパス名 "*string*" が無効です」という例外が発生する可能性があります（*string* には文字列が入ります）。

構文

```
procedure deleteSchema( schemaURL IN varchar2,
                        delete_option IN pls_integer := DELETE_RESTRICT);
```

パラメータ	IN / OUT	説明
schemaURL	(IN)	削除するスキーマを識別する URL。
delete_option	(IN)	スキーマを削除するためのオプション。

delete_option パラメータのオプション

オプション	説明
DELETE_RESTRICT	このスキーマに依存する表またはスキーマが存在する場合、スキーマは正常に削除されません。
DELETE_INVALIDATE	依存性が存在する場合、スキーマは正常に削除されますが、すべての依存オブジェクトが無効になるのみです。
DELETE_CASCADE	スキーマを削除すると、すべてのデフォルトの SQL 型および表も削除されます。ただし、このスキーマに準拠するストアド・インスタンスが存在する場合は、削除が正常に実行されません。
DELETE_CASCADE_FORCE	DELETE_CASCADE に類似しています。ただし、このスキーマに準拠するストアド・インスタンスが存在するかどうかは確認しません。また、エラーも無視します。

compileSchema()

説明

登録済の XML Schema を再コンパイルします。これは、無効なスキーマを有効にする場合に役立ちます。「ORA-31001 リソース・ハンドルまたはパス名 "*string*" が無効です」という例外が発生する可能性があります（*string* には文字列が入ります）。

構文

```
procedure compileSchema( schemaURL IN varchar2);
```

パラメータ	IN / OUT	説明
schemaURL	(IN)	スキーマを識別する URL。

generateSchema()

説明

Oracle オブジェクト型の名前から XML Schema を生成します。「ORA-31001 リソース・ハンドルまたはパス名 "*string*" が無効です」という例外が発生する可能性があります（*string* には文字列が入ります）。次の表に、オプションを示します。

構文	説明
function generateSchemas(schemaName IN varchar2, typeName IN varchar2, elementName IN varchar2 := NULL, schemaURL IN varchar2 := NULL, annotate IN BOOLEAN := TRUE, embedColl IN BOOLEAN := TRUE) return sys.XMLSequenceType;	XMLType のコレクション（データベース・スキーマごとに 1 つの XML Schema 文書）を戻します。

構文	説明
function generateSchema(schemaName IN varchar2, typeName IN varchar2, elementName IN varchar2 := NULL, recurse IN BOOLEAN := TRUE, annotate IN BOOLEAN := TRUE, embedColl IN BOOLEAN := TRUE) return sys.XMLType;	すべてのものを 1 つのスキーマ（XMLType）にインラインに格納します。

パラメータ	IN / OUT	説明
schemaName	(IN)	この型を含むデータベース・スキーマの名前。
typeName	(IN)	Oracle オブジェクト型名。
elementName	(IN)	XMLSchema 内の最上位要素の名前。デフォルトは typeName です。
schemaURL	(IN)	インポート文用の最上位スキーマが必要とする、スキーマの格納先のベース URL。
recurse	(IN)	指定された型によって参照されるすべての型のスキーマも生成するかどうか。
annotate	(IN)	XMLSchema に SQL 注釈を挿入するかどうか。
embedColl	(IN)	コレクションを参照する型にそのコレクションを埋め込む必要があるか、または complexType を作成する必要があるか。注釈を有効にする場合は、`false` を指定できません。

カタログ・ビュー

表 28-3 カタログ・ビュー・スキーマの概要

スキーマ	説明
USER_XML_SCHEMAS（28-11 ページ）	ユーザーが所有するすべての登録済 XML Schema。
ALL_XML_SCHEMAS（28-11 ページ）	現行のユーザーが使用できるすべての登録済 XML Schema。
DBA_XML_SCHEMAS（28-12 ページ）	Oracle XML DB 内のすべての登録済 XML Schema。
DBA_XML_TABLES（28-12 ページ）	システム内のすべての XMLType 表。
USER_XML_TABLES（28-13 ページ）	現行のユーザーが所有するすべての XMLType 表。
ALL_XML_TABLES（28-13 ページ）	現行のユーザーが使用できるすべての XMLType 表。
DBA_XML_TAB_COLS（28-14 ページ）	システム内のすべての XMLType 表の列。
USER_XML_TAB_COLS（28-14 ページ）	現行のユーザーが所有する表内のすべての XMLType 表の列。
ALL_XML_TAB_COLS（28-15 ページ）	現行のユーザーが使用できる表内のすべての XMLType 表の列。
DBA_XML_VIEWS（28-15 ページ）	システム内のすべての XMLType ビュー。
USER_XML_VIEWS（28-16 ページ）	現行のユーザーが所有するすべての XMLType ビュー。
ALL_XML_VIEWS（28-16 ページ）	現行のユーザーが使用できるすべての XMLType ビュー。
DBA_XML_VIEW_COLS（28-16 ページ）	システム内のすべての XMLType ビューの列。
USER_XML_VIEW_COLS（28-17 ページ）	現行のユーザーが所有するビュー内のすべての XMLType ビューの列。
ALL_XML_VIEW_COLS（28-17 ページ）	現行のユーザーが使用できるビュー内のすべての XMLType ビューの列。

USER_XML_SCHEMAS

説明

現行のユーザーが所有するすべてのスキーマ（ローカルおよびグローバル）を示します。

列	データ型	説明
SCHEMA_URL	VARCHAR2	XML Schema URL。
LOCAL	VARCHAR2	ローカル・スキーマ（YES/NO）。
SCHEMA	XMLTYPE	XML Schema 文書。

ALL_XML_SCHEMAS

説明

現行のユーザーが所有するすべてのローカル・スキーマ、およびグローバル・スキーマを示します。

列	データ型	説明
OWNER	VARCHAR2	XML Schema を所有するデータベース・ユーザー。
SCHEMA_URL	VARCHAR2	XML Schema URL。
LOCAL	VARCHAR2	ローカル・スキーマ（YES/NO）。
SCHEMA	XMLTYPE	XML Schema 文書。

DBA_XML_SCHEMAS

説明

システム内のすべての登録済ローカル・スキーマおよびグローバル・スキーマを示します。

列	データ型	説明
OWNER	VARCHAR2	XML Schema を所有するデータベース・ユーザー。
SCHEMA_URL	VARCHAR2	XML Schema URL。
LOCAL	VARCHAR2	ローカル・スキーマ（YES/NO）。
SCHEMA	XMLTYPE	XML Schema 文書。

DBA_XML_TABLES

説明

システム内のすべての XMLType 表を示します。

列	データ型	説明
OWNER	VARCHAR2	表を所有するデータベース・ユーザー。
TABLE_NAME	VARCHAR2	XMLType 表の名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。
ELEMENT_NAME	VARCHAR2	XML Schema 要素。
STORAGE_TYPE	VARCHAR2	記憶域タイプ。CLOB またはオブジェクト・リレーショナルです。

USER_XML_TABLES

説明

現行のユーザーが所有するすべてのローカル XMLType 表を示します。

列	データ型	説明
TABLE_NAME	VARCHAR2	XMLType 表の名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。
ELEMENT_NAME	VARCHAR2	XML Schema 要素。
STORAGE_TYPE	VARCHAR2	記憶域タイプ。CLOB またはオブジェクト・リレーショナルです。

ALL_XML_TABLES

説明

現行のユーザーが所有するすべてのローカル XMLType 表、および現行のユーザーが参照できるすべてのグローバル表を示します。

列	データ型	説明
OWNER	VARCHAR2	表を所有するデータベース・ユーザー。
TABLE_NAME	VARCHAR2	XMLType 表の名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。
ELEMENT_NAME	VARCHAR2	XML Schema 要素。
STORAGE_TYPE	VARCHAR2	記憶域タイプ。CLOB またはオブジェクト・リレーショナルです。

DBA_XML_TAB_COLS

説明

システム内のすべての XMLType 列を示します。

列	データ型	説明
OWNER	VARCHAR2	表を所有するデータベース・ユーザー。
TABLE_NAME	VARCHAR2	表の名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。
ELEMENT_NAME	VARCHAR2	XML Schema 要素。
STORAGE_TYPE	VARCHAR2	記憶域タイプ。CLOB またはオブジェクト・リレーショナルです。

USER_XML_TAB_COLS

説明

現行のユーザーが所有する表内のすべての XMLType 列を示します。

列	データ型	説明
TABLE_NAME	VARCHAR2	表の名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。
ELEMENT_NAME	VARCHAR2	XML Schema 要素。
STORAGE_TYPE	VARCHAR2	記憶域タイプ。CLOB またはオブジェクト・リレーショナルです。

ALL_XML_TAB_COLS

説明

現行のユーザーが所有する表内のすべての XMLType 列、および現行のユーザーが参照できるすべてのグローバル表を示します。

列	データ型	説明
OWNER	VARCHAR2	表を所有するデータベース・ユーザー。
TABLE_NAME	VARCHAR2	表の名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。
ELEMENT_NAME	VARCHAR2	XML Schema 要素。
STORAGE_TYPE	VARCHAR2	記憶域タイプ。CLOB またはオブジェクト・リレーショナルです。

DBA_XML_VIEWS

説明

システム内のすべての XMLType ビューを示します。

列	データ型	説明
OWNER	VARCHAR2	ビューを所有するデータベース・ユーザー。
VIEW_NAME	VARCHAR2	XMLType ビューの名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。
ELEMENT_NAME	VARCHAR2	XML Schema 要素。

USER_XML_VIEWS

説明

現行のユーザーが所有するすべてのローカル XMLType ビューを示します。

列	データ型	説明
VIEW_NAME	VARCHAR2	XMLType ビューの名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。
ELEMENT_NAME	VARCHAR2	XML Schema 要素。

ALL_XML_VIEWS

説明

現行のユーザーが所有するすべてのローカル XMLType ビュー、および現行のユーザーが参照できるすべてのグローバル・ビューを示します。

列	データ型	説明
OWNER	VARCHAR2	ビューを所有するデータベース・ユーザー。
VIEW_NAME	VARCHAR2	XMLType ビューの名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。
ELEMENT_NAME	VARCHAR2	XML Schema 要素。

DBA_XML_VIEW_COLS

説明

システム内のすべての XMLType 列を示します。

列	データ型	説明
OWNER	VARCHAR2	ビューを所有するデータベース・ユーザー。
VIEW_NAME	VARCHAR2	ビューの名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。

列	データ型	説明
ELEMENT_NAME	VARCHAR2	XML Schema 要素。

USER_XML_VIEW_COLS

説明

現行のユーザーが所有するビュー内のすべての XMLType 列を示します。

列	データ型	説明
VIEW_NAME	VARCHAR2	ビューの名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。
ELEMENT_NAME	VARCHAR2	XML Schema 要素。

ALL_XML_VIEW_COLS

説明

現行のユーザーが所有するビュー内のすべての XMLType 列、および現行のユーザーが参照できるすべてのグローバル・ビューを示します。

列	データ型	説明
OWNER	VARCHAR2	ビューを所有するデータベース・ユーザー。
VIEW_NAME	VARCHAR2	ビューの名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema URL。
ELEMENT_NAME	VARCHAR2	XML Schema 要素。

29

PL/SQL でのリソース管理およびアクセス制御

DBMS_XDB パッケージには、PL/SQL 用のリソース管理 API およびアクセス制御 API が含まれています。

この章の内容は次のとおりです。

- DBMS_XDB の説明
- DBMS_XDB のファンクションおよびプロシージャ

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

DBMS_XDB パッケージ

DBMS_XDB の説明

DBMS_XDB パッケージは、PL/SQL アプリケーション開発者に、Oracle XML DB 階層でのリソース管理、Oracle XML DB のアクセス制御リスト（ACL）・セキュリティのサポート、および Oracle XML DB 構成のセッション管理を可能にする API を提供します。

Oracle XML DB のリソース管理機能では、`Link()`、`LockResource()`、`GetLockToken()`、`UnlockResource()`、`CreateResource()`、`CreateFolder()`、`DeleteResource()` および `Link()` ファンクションが提供されます。これらのメソッドは、リソース・ビューによって提供される機能を補完します。

ACL ベースのセキュリティ・メカニズムは、階層内 ACL（Oracle XML DB Resource API によって格納された ACL）またはメモリー内 ACL（Oracle XML DB 外のユーザーによって格納される場合があります）のどちらでも使用できます。これらのメソッドの一部は、Oracle XML DB リソースと任意のデータベース・オブジェクトの両方に使用できます。

アクセス制御セキュリティ機能では、Oracle XML DB リソース用に `checkPrivileges()`、`getAclDocument()`、`changePrivileges()` および `getPrivileges()` ファンクションが提供されます。`AclCheckPrivileges()` ファンクションを使用すると、データベース・ユーザーは、オブジェクトを Oracle XML DB 階層に格納することなく、Oracle XML DB の ACL ベースのセキュリティ・メカニズムにアクセスできます。

Oracle XML DB 構成のセッション管理では、`cfg_refresh()`、`cfg_get()` および `cfg_update()` が提供されます。

DBMS_XDB のファンクションおよびプロシージャ

表 29-1 DBMS_XDB のファンクションおよびプロシージャの概要

ファンクションまたはプロシージャ	説明
getAclDocument()（29-3 ページ）	パス名が指定されたリソースを保護する ACL ドキュメントを取得します。
getPrivileges()（29-4 ページ）	現行のユーザーに付与されている、指定された Oracle XML DB リソースに対するすべての権限を取得します。
changePrivileges()（29-4 ページ）	指定されたアクセス制御エントリ（ACE）を指定されたリソースの ACL に追加します。
checkPrivileges()（29-5 ページ）	現行のユーザーに付与されている、指定された Oracle XML DB リソースに対するすべてのアクセス権を確認します。
setacl()（29-6 ページ）	指定された Oracle XML DB リソースの ACL を、指定された ACL になるように設定します。

表 29-1 DBMS_XDB のファンクションおよびプロシージャの概要（続き）

ファンクションまたはプロシージャ	説明
AclCheckPrivileges()（29-6 ページ）	指定された ACL ドキュメントによって、所有者が owner パラメータに指定されているリソースに対する、現行ユーザーのアクセス権を確認します。
LockResource()（29-7 ページ）	パスが指定されたリソースの WebDAV 形式のロックを取得します。
GetLockToken()（29-8 ページ）	リソースへのパスが指定された現行ユーザーに対するロック・トークンを戻します。
UnlockResource()（29-8 ページ）	ロック・トークンおよびパスが指定されたリソースのロックを解除します。
CreateResource()（29-9 ページ）	新しいリソースを作成します。
CreateFolder()（29-10 ページ）	階層内に新しいフォルダ・リソースを作成します。
DeleteResource()（29-10 ページ）	階層からリソースを削除します。
Link()（29-11 ページ）	既存のリソースへのリンクを作成します。
cfg_refresh()（29-11 ページ）	セッションの構成情報を最新の構成にリフレッシュします。
cfg_get()（29-11 ページ）	セッションの構成情報を取得します。
cfg_update()（29-12 ページ）	構成情報を更新します。

getAclDocument()

説明

パス名が指定されたリソースを保護する ACL ドキュメントを取得し、ACL ドキュメントの XMLType を戻します。

構文

```
FUNCTION getAclDocument( abspath  IN  VARCHAR2)
                         RETURN sys.xmltype;
```

パラメータ	IN / OUT	説明
abspath	(IN)	ACL ドキュメントがリクエストされるリソースのパス名。

getPrivileges()

説明

現行のユーザーに付与されている、指定された Oracle XML DB リソースに対するすべての権限を取得します。現行のユーザーに付与されている、このリソースに対するすべてのリーフ権限がリストされる <privilege> 要素の XMLType インスタンスを戻します。次に例を示します。

```
<privilege xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                              http://xmlns.oracle.com/xdb/acl.xsd"
      <read-contents/>
      <read-properties/>
      <resolve/>
      <read-acl/>
 </privilege>
```

構文

```
FUNCTION getPrivileges( res_path IN VARCHAR2) RETURN sys.xmltype;
```

パラメータ	IN / OUT	説明
res_path	(IN)	階層内における Oracle XML DB リソースの絶対パス。

changePrivileges()

説明

指定された ACE を指定されたリソースの ACL に追加します。ACL が正常に変更された場合は、正の整数を戻します。次に例を示します。

```
<ace  xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:dav="DAV:"
          xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                              http://xmlns.oracle.com/xdb/acl.xsd
                              DAV:http://xmlns.oracle.com/xdb/dav.xsd"
```

```
        <grant>true</grant>
        <principal>SCOTT</principal>
        <privilege>
             <read-contents/>
             <read-properties/>
             <resolve/>
             <dav:waste/>
        </privilege>
   </ace>
```

構文

```
FUNCTION changePrivileges( res_path  IN  VARCHAR2,
                           ace       IN  xmltype)
                           RETURN pls_integer;
```

パラメータ	IN / OUT	説明
res_path	(IN)	権限を変更する必要がある Oracle XML DB リソースのパス名。
ace	(IN)	<principal>、<grant> 操作および権限リストを指定する <ace> 要素の XMLType インスタンス。前述のコード例を参照してください。

同じプリンシパルおよび操作（grant または deny）を含む ACE が ACL 内に存在しない場合は、新しい ACE が ACL の最後に追加されます。

checkPrivileges()

説明

現行のユーザーに付与されている、指定された Oracle XML DB リソースに対するアクセス権を確認します。リクエストされたすべての権限が付与されている場合は、正の整数を戻します。たとえば、次の <privilege> の XMLType インスタンスを使用して、<read.contents>、<read.properties> および <dav:waste> 権限が付与されているかどうかを確認します。

```
   <privilege xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:dav="DAV:"
            xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                                http://xmlns.oracle.com/xdb/acl.xsd
                                DAV: http://xmlns.oracle.com/xdb/dav.xsd"
        <read-contents/>
```

```
        <read-properties/>
        <resolve/>
        <dav:waste/>
    </privilege>
```

構文

```
FUNCTION checkPrivileges( res_path   IN  VARCHAR2,
                          privs      IN  xmltype)
                          RETURN pls_integer;
```

パラメータ	IN / OUT	説明
res_path	(IN)	階層内における Oracle XML DB リソースの絶対パス。
privs	(IN)	リクエストされた一連のアクセス権を指定する privilege 要素の XMLType インスタンス。前述のコード例を参照してください。

setacl()

説明

指定された Oracle XML DB リソースの ACL を、パスによって指定された ACL になるように設定します。リソースに対する <write-acl> 権限が必要です。

構文

```
PROCEDURE setacl( res_path   IN  VARCHAR2,
                  acl_path   IN  VARCHAR2);
```

パラメータ	IN / OUT	説明
res_path	(IN)	階層内における Oracle XML DB リソースの絶対パス。
acl_path	(IN)	階層内における Oracle XML DB の ACL の絶対パス。

AclCheckPrivileges()

説明

指定された ACL ドキュメントによって、所有者が owner パラメータに指定されているリソースに対する、現行ユーザーのアクセス権を確認します。リクエストされたすべての権限が付与されている場合は、正の整数を戻します。

構文

```
FUNCTION AclCheckPrivileges( acl_path  IN  VARCHAR2,
                             owner     IN  VARCHAR2,
                             privs     IN  xmltype)
                             RETURN pls_integer;
```

パラメータ	IN / OUT	説明
acl_path	(IN)	階層内における ACL ドキュメントの絶対パス。
owner	(IN)	リソースの所有者名。ACL 権限の解決中に、このユーザーによって擬似ユーザー「DAV:owner」が置換されます。
privs	(IN)	リクエストされた一連のアクセス権を指定する privilege 要素の XMLType インスタンス。checkPrivileges() の説明を参照してください。

LockResource()

説明

リソースへのパスを指定すると、そのリソースの WebDAV 形式のロックが取得されます。操作が正常に実行された場合は true、エラーが発生した場合は false を戻します。リソースに対する UPDATE 権限が必要です。

構文

```
FUNCTION LockResource( path      IN  VARCHAR2,
                       depthzero IN  BOOLEAN,
                       shared    IN  boolean)
                       RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
path	(IN)	ロックするリソースのパス名。
depthzero	(IN)	現在はサポートされていません。現在、このファンクションでは、指定されたリソースのみがロックされます。将来のリリースでは、false を指定すると、無限深度のロックが取得される予定です。
shared	(IN)	true を指定すると、共有書込みロックが取得されます。

GetLockToken()

説明

リソースへのパスが指定されると、現行のユーザーに対する、そのリソースのロック・トークンを戻します。リソースに対する READPROPERTIES 権限が必要です。

構文

```
PROCEDURE GetLockToken( path      IN  VARCHAR2,
                        locktoken OUT VARCHAR2);
```

パラメータ	IN / OUT	説明
path	(IN)	リソースのパス名。
locktoken	(OUT)	リソースに対するログイン・ユーザーのロック・トークン。

UnlockResource()

説明

ロック・トークンおよびパスが指定されたリソースのロックを解除します。操作が正常に実行された場合は true、エラーが発生した場合は false を戻します。リソースに対する UPDATE 権限が必要です。

構文

```
FUNCTION UnlockResource( path     IN  VARCHAR2,
                         deltoken IN  VARCHAR2)
                         RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
path	(IN)	リソースのパス名。
deltoken	(IN)	削除するロック・トークン。

CreateResource()

説明

新しいリソースを作成します。操作が正常に実行された場合は `true`、エラーが発生した場合は `false` を戻します。次の表に、オプションを示します。

構文	説明
FUNCTION CreateResource(path IN VARCHAR2, data IN VARCHAR2) RETURN BOOLEAN;	指定された文字列をコンテンツとして含む新しいリソースを作成します。
FUNCTION CreateResource(path IN VARCHAR2, data IN SYS.XMLTYPE) RETURN BOOLEAN;	指定された XMLType データをコンテンツとして含む新しいリソースを作成します。
FUNCTION CreateResource(path IN VARCHAR2, datarow IN REF SYS.XMLTYPE) RETURN BOOLEAN;	既存の XMLType 行への参照が指定されると、その行を指すコンテンツを含むリソースを作成します。別のリソース内に存在していない行を指定する必要があります。
FUNCTION CreateResource(path IN VARCHAR2, data IN CLOB) RETURN BOOLEAN;	指定された CLOB をコンテンツとして含むリソースを作成します。
FUNCTION CreateResource(path IN VARCHAR2, data IN BFILE) RETURN BOOLEAN;	指定された BFILE をコンテンツとして含むリソースを作成します。

パラメータ	IN / OUT	説明
path	(IN)	作成するリソースのパス名。パス名の親フォルダがすでに階層内に存在している必要があります。「/foo/bar.txt」を指定する場合は、フォルダ「/foo」がすでに存在している必要があります。

パラメータ	IN / OUT	説明
data	(IN)	新しいリソースのコンテンツ。このデータは、スキーマベースの XML 文書を含んでいるかどうかを確認するために解析され、その内容がスキーマのデフォルトの表にスキーマベースのデータとして格納されます。スキーマベースの XML 文書を含まない場合は、バイナリ・データとして保存されます。
datarow	(IN)	コンテンツとして使用する XMLType 行への参照。

CreateFolder()

説明

階層内に新しいフォルダ・リソースを作成します。操作が正常に実行された場合は `true`、エラーが発生した場合は `false` を戻します。指定されたパス名の親フォルダがすでに階層内に存在している必要があります。たとえば、path パラメータに「`/folder1/folder2`」を指定する場合は、「`/folder1`」がすでに存在している必要があります。

構文

```
FUNCTION CreateFolder( path   IN  VARCHAR2)
                     RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
path	(IN)	新しいフォルダのパス名。

DeleteResource()

説明

階層からリソースを削除します。

構文

```
PROCEDURE DeleteResource( path   IN  VARCHAR2);
```

パラメータ	IN / OUT	説明
path	(IN)	削除するリソースのパス名。

Link()

説明

既存のリソースへのリンクを作成します。UNIX でハード・リンクを作成する場合と同様です。

構文

```
PROCEDURE Link( srcpath    IN  VARCHAR2,
                linkfolder IN  VARCHAR2,
                linkname   IN  VARCHAR2);
```

パラメータ	IN / OUT	説明
srcpath	(IN)	リンク先とするリソースのパス名。
linkfolder	(IN)	新しいリンクを置くフォルダ。
linkname	(IN)	新しいリンクの名前。

cfg_refresh()

説明

セッションの構成情報を最新の構成にリフレッシュします。

構文

```
PROCEDURE cfg_refresh;
```

cfg_get()

説明

セッションの構成情報を XMLType インスタンスとして取得します。

構文

```
FUNCTION cfg_get RETURN SYS.XMLType;
```

cfg_update()

説明

構成情報を更新し、コミットします。

構文

```
PROCEDURE cfg_update( xdbconfig   IN  SYS.XMLTYPE);
```

パラメータ	IN / OUT	説明
xdbconfig	(IN)	新しい構成データ。

30

PL/SQL での DBMS_XMLGEN を使用した問合せの生成

DBMS_XMLGEN パッケージを使用すると、SQL 問合せの結果を正規の XML 形式に変換できます。

この章の内容は次のとおりです。

- DMS_XMLGEN の説明
- DBMS_XMLGEN のファンクションおよびプロシージャ

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

DBMS_XMLGEN パッケージ

DMS_XMLGEN の説明

DBMS_XMLGEN は、SQL 問合せの結果を正規の XML 形式に変換します。このパッケージは、入力として任意の SQL 問合せを取り、XML 形式に変換し、結果を CLOB として戻します。

このパッケージは、DBMS_XMLQUERY パッケージと類似しています。ただし、DBMS_XMLGEN は C で作成され、カーネルにコンパイルされます。このパッケージは、データベース内でのみ実行できます。

DBMS_XMLGEN のファンクションおよびプロシージャ

表 30-1 DBMS_XMLGEN のファンクションおよびプロシージャの概要

ファンクションまたはプロシージャ	説明
newContext()（30-3 ページ）	新しいコンテキスト・ハンドルを作成します。
setRowTag()（30-4 ページ）	結果の各行を囲む要素の名前を設定します。デフォルトのタグは ROW です。
setRowSetTag ()（30-4 ページ）	結果全体を囲む要素の名前を設定します。デフォルトのタグは ROWSET です。
getXML()（30-5 ページ）	XML 文書を取得します。
getNumRowsProcessed()（30-6 ページ）	getXML を最後にコールしたときに処理された SQL の行数を取得します。
setMaxRows()（30-6 ページ）	getXML のコールごとにフェッチする行の最大数を設定します。
setSkipRows()（30-7 ページ）	getXML をコールするたびに、XML を生成する前にスキップする行数を設定します。デフォルトは 0（ゼロ）です。
setConvertSpecialChars()（30-7 ページ）	非 XML 文字である $ などの特殊文字をエスケープ文字に変換する必要があるかどうかを設定します。デフォルトでは、変換が実行されます。
convert()（30-8 ページ）	XML を等価のエスケープされた XML またはエスケープされていない XML に変換します。
useItemTagsForColl()（30-9 ページ）	コレクション要素に対して、_ITEM タグを追加したコレクション列名を強制的に使用します。デフォルトでは、コレクションのベース要素に対する実際の実装となるオブジェクトの型名が設定されます。
restartQuery()（30-9 ページ）	問合せを再度開始して、最初からフェッチを開始します。

表 30-1 DBMS_XMLGEN のファンクションおよびプロシージャの概要（続き）

ファンクションまたはプロシージャ	説明
closeContext()（30-9 ページ）	コンテキストをクローズし、すべてのリソースを解放します。

newContext()

説明

新しいコンテキスト・ハンドルを生成し、戻します。このコンテキスト・ハンドルを getXML() およびその他のファンクションで使用して、結果から XML を戻します。次の表に、オプションを示します。

構文	説明
DBMS_XMLGEN.newContext (query IN VARCHAR2) RETURN ctxHandle;	問合せから新しいコンテキスト・ハンドルを生成します。
DBMS_XMLGEN.newContext (queryString IN SYS_REFCURSOR) RETURN ctxHandle;	PL/SQL REF カーソル形式の問合せ文字列から新しいコンテキスト・ハンドルを生成します。

パラメータ	IN / OUT	説明
query	(IN)	結果を XML に変換する必要がある VARCHAR 形式の問合せ。
queryString	(IN)	結果を XML に変換する必要がある PL/SQL REF カーソル形式の問合せ文字列。

setRowTag()

説明

すべての行を区切る要素の名前を設定します。デフォルト名は ROW です。この名前を null に設定して、ROW 要素自体を出力しないようにすることができます。ROW および ROWSET の両方が null で、出力に複数の列または行がある場合は、エラーが発生します。これは、生成された XML が最上位の囲みタグを持たず、無効であるためです。

構文

```
DBMS_XMLGEN.setRowTag (
   ctx     IN ctxHandle,
   rowTag  IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctx	(IN)	newContext をコールして取得されるコンテキスト・ハンドル。
rowTag	(IN)	ROW 要素名。ROW 要素が必要ない場合は、null を指定します。

setRowSetTag ()

説明

文書のルート要素の名前を設定します。デフォルト名は ROWSET です。rowSetTag を null に設定して、この要素を出力しないようにすることができます。ROW および ROWSET の両方が null で、出力に複数の列または行がある場合は、エラーが発生します。これは、生成された XML が最上位の囲みタグを持たず、無効であるためです。

構文

```
DBMS_XMLGEN.setRowSetTag (
   ctx        IN ctxHandle,
   rowSetTag  IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctx	(IN)	newContext をコールして取得されるコンテキスト・ハンドル。
rowSetTag	(IN)	文書要素名。ROWSET 要素が必要ない場合は、null を指定します。

getXML()

説明

XML 文書を取得します。setSkipRows() コールに指定された行がスキップされ、setMaxRows() コールに指定された最大数の行（または指定されていない場合は結果全体）がフェッチされ、XML に変換されます。getNumRowsProcessed() を使用して、行が取り出されたかどうかを確認します。次の表に、オプションを示します。

構文	説明
FUNCTION DBMS_XMLGEN.getXML (ctx IN ctxHandle, clobval IN OUT NCOPY clob, dtdOrSchema IN number := NONE) RETURN boolean;	指定されている最大数の行をフェッチすることによって、XML 文書を取得します。XML 文書を渡された CLOB に追加します。このバージョンの getXML() を使用すると、余分な CLOB コピーの生成を回避したり、後続のコールで同じ CLOB を再利用できます。CLOB を再利用するため、この getXML() コールはより効率的です。
FUNCTION DBMS_XMLGEN.getXML (ctx IN ctxHandle, dtdOrSchema IN number := NONE) RETURN clob;	XML 文書を生成し、それを一時 CLOB として戻します。DBMS_LOB.FREETEMPORARY をコールして、このファンクションから取得した一時 CLOB を解放する必要があります。
FUNCTION DBMS_XMLGEN.getXML (sqlQuery IN VARCHAR2, dtdOrSchema IN number := NONE) RETURN clob;	SQL 問合せ文字列の結果を XML 形式に変換し、その XML を一時 CLOB として戻します。次に、DBMS_LOB.FREETEMPORARY をコールして、この一時 CLOB を解放する必要があります。
FUNCTION DBMS_ XMLGEN.getXMLType (ctx IN ctxhandle, dtdOrSchema IN number := NONE) RETURN sys.XMLType;	XML 文書を生成し、それを sys.XMLType として戻します。結果に対して、ExistsNode や Extract などの XMLType 操作を実行できます。これは、結果サイズが 4KB 未満の場合に、getStringVal() ファンクションを使用することによって、結果を文字列として取得する方法も提供します。
FUNCTION DBMS_ XMLGEN.getXMLType (sqlQuery IN VARCHAR2, dtdOrSchema IN number := NONE) RETURN sys.XMLType	SQL 問合せ文字列の結果を XML 形式に変換し、その XML を sys.XMLType として戻します。結果に対して、ExistsNode や Extract などの XMLType 操作を実行できます。これは、結果サイズが 4KB 未満の場合に、getStringVal() ファンクションを使用することによって、結果を文字列として取得する方法も提供します。

パラメータ	IN / OUT	説明
ctx	(IN)	newContext をコールして取得されるコンテキスト・ハンドル。
clobval	(IN/OUT)	XML 文書を追加する CLOB。
sqlQuery	(IN)	SQL 問合せ文字列。
dtdOrSchema	(IN)	DTD またはスキーマの生成を示すブール値。現在サポートされているオプションは NONE のみです。

getNumRowsProcessed()

説明

getXML をコールして XML を生成するときに処理される、SQL の行数を取得します。この行数には、XML の生成前にスキップされる行数は含まれません。ループで getXML() をコールする場合、このファンクションを使用して終了条件を判別します。getXML() は、行がない場合でも常に XML 文書を生成することに注意してください。

構文

```
DBMS_XMLGEN.getNumRowsProcessed (
   ctx  IN ctxHandle)
RETURN NUMBER;
```

パラメータ	IN / OUT	説明
ctx	(IN)	newContext をコールして取得されるコンテキスト・ハンドル。

setMaxRows()

説明

getXML のコールごとに、SQL 問合せ結果からフェッチする行の最大数を設定します。結果のページ区切りを生成するときに使用します。たとえば、XML または HTML データのページを生成する場合、maxRows パラメータを設定することによって、XML または HTML に変換する行数を制限します。

構文

```
DBMS_XMLGEN.setMaxRows (
   ctx      IN ctxHandle,
   maxRows  IN NUMBER);
```

パラメータ	IN / OUT	説明
ctx	(IN)	実行された問合せに対応するコンテキスト・ハンドル。
maxRows	(IN)	getXML をコールするたびに取得する最大行数。

setSkipRows()

説明

getXML ルーチンをコールするたびに、XML 出力を生成する前に指定された数の行をスキップします。このユーティリティを使用して、ステートレスな Web ページの結果のページ区切りを生成するときに使用します。たとえば、XML または HTML データの最初のページを生成する場合は、skipRows を 0（ゼロ）に設定します。次に設定するときは、skipRows を最初に指定した行数に設定できます。getNumRowsProcessed() も参照してください。

構文

```
DBMS_XMLGEN.setSkipRows (
   ctx       IN ctxHandle,
   skipRows  IN NUMBER);
```

パラメータ	IN / OUT	説明
ctx	(IN)	実行された問合せに対応するコンテキスト・ハンドル。
skipRows	(IN)	getXML をコールするたびにスキップする行数。

setConvertSpecialChars()

説明

XML データの特殊文字を、XML の等価のエスケープ文字に変換する必要があるかどうかを設定します。たとえば、「<」符号は < に変換されます。デフォルトでは、変換が実行されます。入力データに、エスケープする必要がある <、>、"、' などの特殊文字を含めることができない場合にこのファンクションを使用すると、XML 処理のパフォーマンスが向上します。特に大量のデータを処理する場合、文字データをスキャンして特殊文字を置換するとコストがかかります。

構文

```
DBMS_XMLGEN.setConvertSpecialChars (
   ctx   IN ctxHandle,
   conv  IN boolean);
```

パラメータ	IN / OUT	説明
ctx	(IN)	newContext をコールして取得されるコンテキスト・ハンドル。
conv	(IN)	変換が必要かどうか。

convert()

説明

XML データを等価のエスケープされた XML データまたはエスケープされていない XML データに変換し、XML CLOB データをエンコードされた形式またはデコードされた形式で戻します。ENTITY_ENCODE が指定されている場合、XML データをエスケープします。たとえば、文字 < のエスケープされた形式は < です。エスケープの解除は逆の変換です。次の表に、オプションを示します。

構文	説明
DBMS_XMLGEN.convert (xmlData IN VARCHAR2, flag IN NUMBER := ENTITY_ENCODE) RETURN VARCHAR2;	XML データを文字列形式（VARCHAR2）で使用します。
DBMS_XMLGEN.convert (xmlData IN CLOB, flag IN NUMBER := ENTITY_ENCODE) RETURN CLOB;	XML データを CLOB 形式で使用します。

パラメータ	IN / OUT	説明
xmlData	(IN)	エンコードまたはデコードする XML CLOB データ。
flag	(IN)	フラグ設定。エンコードする場合は ENTITY_ENCODE（デフォルト）、デコードする場合は ENTITY_DECODE です。

useItemTagsForColl()

説明

コレクション要素のデフォルト名をオーバーライドします。コレクション要素のデフォルト名は、型名自体です。このファンクションを使用すると、列名を、列名に _ITEM タグを追加した名前にオーバーライドできます。NUMBER というコレクションがある場合、コレクション要素のデフォルト・タグ名は NUMBER になります。このプロシージャを使用すると、このデフォルト動作をオーバーライドして、デフォルト名に _ITEM タグを追加したコレクション列名を生成できます。

構文

```
DBMS_XMLGEN.useItemTagsForColl (
   ctx  IN ctxHandle);
```

パラメータ	IN / OUT	説明
ctx	(IN)	コンテキスト・ハンドル。

restartQuery()

説明

問合せを再度開始し、最初の行から XML を生成します。このプロシージャを使用すると、新しいコンテキストを作成することなく、問合せの実行を再度開始できます。

構文

```
DBMS_XMLGEN.restartQuery (ctx  IN ctxHandle);
```

パラメータ	IN / OUT	説明
ctx	(IN)	現行の問合せに対応するコンテキスト・ハンドル。

closeContext()

説明

指定されたコンテキストをクローズし、SQL カーソル、バインドや定義バッファなど、そのコンテキストに対応付けられているすべてのリソースを解放します。このファンクションをコールした後は、後続の DBMS_XMLGEN ファンクションのコールでこのハンドルを使用できなくなります。

構文

```
DBMS_XMLGEN.closeContext ( ctx  IN ctxHandle);
```

パラメータ	IN / OUT	説明
ctx	(IN)	クローズするコンテキスト・ハンドル。

31

Oracle XML DB Resource View API for PL/SQL

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』

Oracle XML DB Resource View API

Oracle XML DB Resource View パッケージの説明

リソース・ビューおよびパス・ビューでは、Oracle XML DB リポジトリに格納されたデータに SQL を介してアクセスするためのメカニズムが提供されます。これらのビューを使用すると、FTP や WebDAV などのプロトコルまたは JNDI などのプログラム API を介して Oracle XML DB リポジトリに格納されたデータに SQL でアクセスしたり、その逆の操作も行うことができます。リソース・ビューおよびパス・ビューを一部の PL/SQL パッケージとともに使用すると、プログラム API を介して使用できるすべての問合せおよび DML 機能が提供されます。パス・ビューにはリポジトリ内の一意のパスごとに 1 つの行があり、リソース・ビューにはリポジトリ内のリソースごとに 1 つの行があります。

Oracle XML DB Resource View パッケージの演算子

表 31-1 Oracle XML DB Resource View パッケージの演算子の概要

演算子	説明
UNDER_PATH（31-2 ページ）	Oracle XML DB の階層インデックスを使用して、特定のパスのサブパスを戻します。
EQUALS_PATH（31-4 ページ）	指定されたパス名を持つリソースを検索します。
PATH（31-4 ページ）	pathname 引数に指定されたパス名の下に存在するリソースの相対パス名を戻します。
DEPTH（31-5 ページ）	指定された開始パスの下に存在するリソースのフォルダの深度を戻します。

UNDER_PATH

説明

UNDER_PATH 演算子は、Oracle XML DB の階層インデックスを使用して、特定のパスの下に存在するパスを戻します。通常、階層インデックスは、下位層のパスへのアクセスを高速化するために使用します。問合せ述語の条件が他の部分と多くマッチしてしまう場合がありますが、UNDER_PATH の機能の実装では、特定のリソースを基準として上位に階層を検索します。検索する必要があるリンクの数が大幅に減少する場合があるため、これによって効率が向上する可能性があります。次の表に、オプションを示します。

構文	説明
INTEGER UNDER_PATH(resource_column, pathname);	指定されたパスの下にリソースが存在するかどうかを判別します。
INTEGER UNDER_PATH(resource_column, depth, pathname);	検索するレベル数を制限する depth 引数を使用して、指定されたパスの下にリソースが存在するかどうかを判別します。
INTEGER UNDER_PATH(resource_column, pathname, correlation)	補助演算子に対する correlation 引数を使用して、指定されたパスの下にリソースが存在するかどうかを判別します。
INTEGER UNDER_PATH(resource_column, depth, pathname, correlation)	検索するレベル数を制限する depth 引数、および補助演算子に対する correlation 引数を使用して、指定されたパスの下にリソースが存在するかどうかを判別します。 1 つのリソースを戻すには、リソースにアクセスできるパスのいずれかが、path 引数に指定されたパスの下に存在する必要があります。

パラメータ	説明
resource_column	path_view または resource_view 内の resource 列の列名または別名。
pathname	解決するパス名。
depth	検索する最大深度。0（ゼロ）未満の深度は、0（ゼロ）として処理されます。
correlation	UNDER_PATH 演算子（主演算子）を補助演算子（PATH および DEPTH）と相関させるために使用できる整数。

EQUALS_PATH

説明

指定されたパス名を持つリソースを検索します。EQUALS_PATH 演算子は、機能上、深度が 0（ゼロ）に制限されている UNDER_PATH と同じです。

構文

```
EQUALS_PATH INTEGER EQUALS_PATH( resource_column,
                                 pathname);
```

パラメータ	説明
resource_column	path_view または resource_view 内の resource 列の列名または別名。
pathname	解決するパス名。

PATH

説明

pathname 引数に指定されたパス名の下に存在するリソースの相対パス名を戻す補助演算子です。リソース・ビュー内の path 列は常にリソースの絶対パスを含むことに注意してください。

構文

```
PATH VARCHAR2 PATH( correlation);
```

パラメータ	説明
correlation	UNDER_PATH 演算子（主演算子）を補助演算子（PATH および DEPTH）と相関させるために使用できる整数。

DEPTH

説明

指定された開始パス下に存在するリソースのフォルダの深度を戻す補助演算子です。

構文

```
DEPTH  INTEGER DEPTH( correlation);
```

パラメータ	説明
correlation	UNDER_PATH 演算子（主演算子）を補助演算子（PATH および DEPTH）と相関させるために使用できる整数。

32

Oracle XML DB バージョニング API for PL/SQL

Oracle XML DB バージョニング API は、DBMS_XDB_VERSION パッケージに含まれています。

この章の内容は次のとおりです。

- DBMS_XDB_VERSION の説明
- DBMS_XDB_VERSION のファンクションおよびプロシージャ

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

DBMS_XDB_VERSION パッケージ

DBMS_XDB_VERSION の説明

DBMS_XDB_VERSION のファンクションおよびプロシージャは、VCR を作成し、バージョン履歴のバージョンを管理する場合に有効です。

DBMS_XDB_VERSION のファンクションおよびプロシージャ

表 32-1 DBMS_XDB_VERSION のファンクションおよびプロシージャの概要

ファンクションまたはプロシージャ	説明
MakeVersioned()（32-3 ページ）	指定されたパス名を持つ通常のリソースをバージョン管理されたリソースに変換します。
Checkout()（32-3 ページ）	VCR を更新または削除する前に、その VCR をチェックアウトします。
Checkin()（32-4 ページ）	チェックアウトされた VCR をチェックインし、新しく作成されたバージョンのリソース ID を戻します。
Uncheckout()（32-4 ページ）	チェックアウトされたリソースをチェックインし、リソースがチェックアウトされる前にバージョンのリソース ID を戻します。
GetPredecessors()（32-5 ページ）	パス名によって先行バージョンのリストを取得します。
GetPredsByResId()（32-5 ページ）	リソース ID によって先行バージョンのリストを取得します。
GetResourceByResId()（32-5 ページ）	リソース・オブジェクト ID を指定して、リソースを XMLType として取得します。
GetSuccessors()（32-5 ページ）	パス名によって後続バージョンのリストを取得します。
GetSuccsByResId()（32-7 ページ）	リソース ID によって後続バージョンのリストを取得します。

MakeVersioned()

説明

指定されたパス名を持つ通常のリソースをバージョン管理されたリソースに変換します。2 つ以上のパス名が同じリソースにバインドされている場合、そのリソースのコピーが作成され、指定されたパス名が新しく作成されたコピーにバインドされます。その後、この新しいリソースはバージョン管理されます。他のすべてのパス名は、元のリソースを参照したままになります。このファンクションは、VCR の最初のバージョン（ルート）のリソース ID を戻します。これは、自動コミットの SQL 操作ではありません。

- VCR に対して MakeVersioned() をコールできます。例外や警告は発生しません。
- フォルダ、バージョン履歴、バージョン・リソースおよび ACL に対して MakeVersioned() をコールできます。
- スキーマベースのリソースはサポートされません。

リソースが存在しない場合は、例外が発生します。

構文

```
FUNCTION MakeVersioned( pathname VARCHAR2) RETURN dbms_xdb.resid_type;
```

パラメータ	説明
pathname	バージョン管理するリソースのパス名

Checkout()

説明

VCR を更新または削除する前に、その VCR をチェックアウトします。これは、自動コミットの SQL 操作ではありません。同じ作業領域を持つ 2 人のユーザーは、同時に同じ VCR に対して Checkout() 操作を実行できません。同時に実行した場合は、どちらか片方のユーザーがロールバックする必要があります。そのため、リソースを更新する前に Checkout() 操作をコミットし、トランザクションがロールバックされた場合の更新の損失を回避することをお薦めします。指定されたリソースが VCR でない場合、VCR がすでにチェックアウトされている場合、またはリソースが存在しない場合は、例外が発生します。

構文

```
PROCEDURE Checkout(  pathname VARCHAR2);
```

パラメータ	説明
pathname	チェックアウトする VCR のパス名。

Checkin()

説明

チェックアウトされた VCR をチェックインし、新しく作成されたバージョンのリソース ID を戻します。これは、自動コミットの SQL 操作ではありません。Checkin() は、Checkout() 操作に渡されたパス名と同じパス名を取る必要はありません。ただし、操作を正常に行うには、Checkin() のパス名と Checkout() のパス名が同じリソースのものである必要があります。リソースの名前が変更された場合、古い名前は無効であるか、または現在別のリソースにバインドされているため、Checkin() に対して新しい名前を使用する必要があります。パス名が存在しない場合は、例外が発生します。パス名が変更されている場合は、新しいパス名を使用して、リソースに対する Checkin() 操作を行う必要があります。

構文

```
FUNCTION Checkin( pathname VARCHAR2) RETURN dbms_xdb.resid_type;
```

パラメータ	説明
pathname	チェックアウトされたリソースのパス名。

Uncheckout()

説明

チェックアウトされたリソースをチェックインし、リソースがチェックアウトされる前にバージョンのリソース ID を戻します。これは、自動コミットの SQL 操作ではありません。Uncheckout() は、Checkout() 操作に渡されたパス名と同じパス名を取る必要はありません。ただし、操作を正常に行うには、Uncheckout() のパス名と Checkout() のパス名が同じリソースのものである必要があります。リソースの名前が変更された場合、古い名前は無効であるか、または現在別のリソースにバインドされているため、チェックアウトを解除するために新しい名前を使用する必要があります。パス名が存在しない場合は、例外が発生します。パス名が変更されている場合は、新しいパス名を使用して、リソースに対する Uncheckout() 操作を行う必要があります。

構文

```
FUNCTION Uncheckout( pathname VARCHAR2) RETURN dbms_xdb.resid_type;
```

パラメータ	説明
pathname	チェックアウトされたリソースのパス名。

GetPredecessors()

説明

パス名によって先行バージョンのリストを取得します。pathname が無効な場合は、例外が発生します。

構文

```
FUNCTION GetPredecessors( pathname VARCHAR2) RETURN resid_list_type;
```

パラメータ	説明
pathname	リソースのパス名。

GetPredsByResId()

説明

リソース ID によって先行バージョンのリストを取得します。resid によって先行バージョンを取得すると、パス名によって取得するより効率的です。resid が無効な場合は、例外が発生します。

構文

```
FUNCTION GetPredsByResId( resid resid_type) RETURN resid_list_type;
```

パラメータ	説明
resid	リソース ID。

GetResourceByResId()

説明

リソース・オブジェクト ID を指定して、リソースを XMLType として取得します。システムではバージョンのパス名が作成されないため、このファンクションは、リソース ID を使用してリソースを取得する場合に有効です。

構文

```
FUNCTION GetResourceByResId( resid resid_type) RETURN XMLType;
```

パラメータ	説明
resid	リソース ID。

GetSuccessors()

説明

バージョン・リソースまたは VCR を指定して、パス名によってリソースの後続バージョンのリストを取得します。resid によって後続のバージョンを取得すると、pathname によって取得するより効率的です。pathname が無効な場合は、例外が発生します。

構文

```
FUNCTION GetSuccessors( pathname VARCHAR2) RETURN resid_list_type;
```

パラメータ	説明
pathname	リソースのパス名。

GetSuccsByResId()

説明

バージョン・リソースまたは VCR を指定して、リソース ID によってリソースの後続バージョンのリストを取得します。resid によって後続のバージョンを取得すると、パス名によって取得するより効率的です。resid が無効な場合は、例外が発生します。

構文

```
FUNCTION GetSuccsByResId( resid resid_type) RETURN resid_list_type;
```

パラメータ	説明
resid	リソース ID。

33

ConText インデックスの管理 : PL/SQL の DBMS_XDBT

DBMS_XDBT パッケージを使用すると、管理者は XML DB 階層に対して ConText インデックスを作成し、それを自動メンテナンスできるように構成できます。

この章の内容は次のとおりです。

- DBMS_XDBT の説明
- DBMS_XDBT のファンクションおよびプロシージャ
- DBMS_XDBT パッケージのカスタマイズ

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

DBMS_XDBT パッケージ

DBMS_XDBT の説明

DBMS_XDBT パッケージは、管理者が Oracle XML DB 階層に対して ConText インデックスを設定する場合に有効なメカニズムを提供します。このパッケージには、デフォルトのプリファレンスを作成するプロシージャ、インデックスを作成するプロシージャ、および ConText インデックスの自動同期を設定するプロシージャが含まれます。

また、DBMS_XDBT パッケージには、インデックスの構成設定を定義する一連のパッケージ変数も含まれます。これらの変数は、インストールに必要な可能性がある基本的なカスタマイズの処理に使用されますが、完全な変数セットではありません。

DBMS_XDBT パッケージは、次の方法で使用できます。

- パッケージをカスタマイズして、適切な構成を設定します。
- dropPreferences() プロシージャを使用して、既存のすべての索引プリファレンスを削除します。
- createPreferences() プロシージャを使用して、新しい索引プリファレンスを作成します。
- createIndex() プロシージャを使用して、ConText インデックスを作成します。
- configureAutoSync() プロシージャを使用して、インデックスの自動同期を設定します。

DBMS_XDBT のファンクションおよびプロシージャ

表 33-1 DBMS_XDBT のファンクションおよびプロシージャの概要

ファンクションまたはプロシージャ	説明
dropPreferences()（33-3 ページ）	既存のすべてのプリファレンスを削除します。
createPreferences()（33-3 ページ）	XML DB 階層に対する ConText インデックスに必要なプリファレンスを作成します。
createDatastorePref()（33-4 ページ）	ConText インデックス用の USER データストア・プリファレンスを作成します。
createFilterPref()（33-4 ページ）	ConText インデックス用のフィルタ・プリファレンスを作成します。
createLexerPref()（33-5 ページ）	ConText インデックス用のレクサー・プリファレンスを作成します。

表 33-1 DBMS_XDBT のファンクションおよびプロシージャの概要（続き）

ファンクションまたはプロシージャ	説明
createWordlistPref()（33-5 ページ）	ConText インデックス用のストップリストを作成します。
createStoplistPref()（33-5 ページ）	ConText インデックス用のセクション・グループを作成します。
createStoragePref()（33-6 ページ）	ConText インデックス用のワードリスト・プリファレンスを作成します。
createSectiongroupPref()（33-6 ページ）	ConText インデックス用の記憶域プリファレンスを作成します。
createIndex()（33-7 ページ）	XML DB 階層に対して ConText インデックスを作成します。
configureAutoSync()（33-7 ページ）	ConText インデックスを自動メンテナンス（SYNC）できるように構成します。

dropPreferences()

説明

XML DB 階層に対する ConText インデックス用に以前作成したすべてのプリファレンスを削除します。

構文

```
PROCEDURE dropPreferences;
```

createPreferences()

説明

構成設定に基づいて、一連のデフォルトのプリファレンスを作成します。

構文

```
PROCEDURE createPreferences;
```

createDatastorePref()

説明

XML DB 階層に対する ConText インデックス用の USER データストア・プリファレンスを作成します。

- データストア・プリファレンスの名前は変更できます。DatastorePref 構成設定の説明を参照してください。
- デフォルトの USER データストア・プロシージャは、受信したドキュメントもフィルタします。DBMS_XDBT パッケージは、フィルタ処理を制御する一連の構成設定を提供します。
- SkipFilter_Types 配列には、正規表現のリストが含まれます。これらのいずれかの表現に一致する MIME タイプを持つドキュメントは、インデックス付けされません。ドキュメント・メタデータの一部のプロパティ（author など）は、インデックス付けされないままになります。
  - NullFilter_Types 配列には、正規表現のリストが含まれます。これらのいずれかの表現に一致する MIME タイプを持つドキュメントは、フィルタされません。ただし、その場合も、インデックス付けはされます。これは、HTML、XML、プレーン・テキストなどのテキストベースのドキュメントに使用されます。
  - 他のすべてのドキュメントは、IFilter API を介して INSO フィルタを使用します。

構文

```
PROCEDURE createDatastorePref;
```

createFilterPref()

説明

XML DB 階層に対する ConText インデックス用の null のフィルタ・プリファレンスを作成します。

- フィルタ・プリファレンスの名前は変更できます。FilterPref 構成設定の説明を参照してください。
- USER データストア・プロシージャは、受信したドキュメントをフィルタします。詳細は、createDatastorePref を参照してください。

構文

```
PROCEDURE createFilterPref;
```

createLexerPref()

説明

XML DB 階層に対する ConText インデックス用の基本的なレクサー・プリファレンスを作成します。

- レクサー・プリファレンスの名前は変更できます。LexerPref 構成設定の説明を参照してください。その他の構成設定は提供されません。
- マルチレクサー・プリファレンスはサポートされていません。
- デフォルトでは、基本文字変換が有効になります。

構文

```
PROCEDURE createLexerPref;
```

createWordlistPref()

説明

XML DB 階層に対する ConText インデックス用のワードリスト・プリファレンスを作成します。

- ワードリスト・プリファレンスの名前は変更できます。WordlistPref 構成設定の説明を参照してください。その他の構成設定は提供されません。
- FUZZY_MATCH 属性および STEMMER 属性は、AUTO（自動言語検出）に設定されます。

構文

```
PROCEDURE createWordlistPref;
```

createStoplistPref()

説明

XML DB 階層に対する ConText インデックス用のストップリストを作成します。

- ストップリストの名前は変更できます。StoplistPref 構成設定の説明を参照してください。
- 数値はインデックス付けされません。
- StopWords 配列は、ストップワードの構成可能なリストです。このリストのストップワードは、CTXSYS.DEFAULT_STOPLIST に指定された一連のストップワード以外のストップワードである必要があります。

構文

```
PROCEDURE createStoplistPref;
```

createStoragePref()

説明

XML DB 階層に対する ConText インデックス用の基本記憶域プリファレンスを作成します。

- 記憶域プリファレンスの名前は変更できます。StoragePref 構成設定の説明を参照してください。
- ConText インデックスを構成する表およびインデックスに表領域を指定できます。IndexTablespace の構成設定を参照してください。
- デフォルトでは、接頭辞および部分文字列のインデックス付けは有効になりません。
- I_INDEX_CLAUSE は、キー圧縮を使用します。

構文

```
PROCEDURE createStoragePref;
```

createSectiongroupPref()

説明

XML DB 階層に対する ConText インデックス用のセクション・グループを作成します。

- セクション・グループの名前は変更できます。SectiongroupPref 構成設定の説明を参照してください。
- デフォルトでは、HTML セクションが使用され、ゾーン・セクションは作成されません。ほとんどの文書が XML の場合は、AUTO_SECTION_GROUP または PATH_SECTION_GROUP の使用を検討してください（SectionGroup 構成設定の説明を参照）。

構文

```
PROCEDURE createSectiongroupPref;
```

createIndex()

説明

XML DB 階層に対して ConText インデックスを作成します。

構文

```
PROCEDURE createIndex;
```

注意

- インデックスの名前は変更できます。IndexName 構成設定の説明を参照してください。
- インデックスの作成中の ROWID ロギングを有効化するには、LogFile 構成パラメータを設定します。
- インデックスの作成およびその後の SYNC に使用されるメモリー量を判別するには、IndexMemory 構成パラメータを設定します。

configureAutoSync()

説明

ConText インデックスの自動 SYNC のためのジョブを設定します。

- 自動同期のためのジョブ・キューを使用できるように、システムを構成する必要があります。ジョブは、USER_JOBS カタログ・ビューを使用して参照できます。
- AutoSyncPolicy 構成パラメータを設定して、適切な同期ポリシーを選択できます。

同期は、次のいずれかの基準に基づいて実行できます。

SYNC_BY_PENDING_COUNT	SYNC は、ペンディング・キュー内のドキュメント数がしきい値（MaxPendingCount 構成設定の説明を参照）を超えた場合にトリガーされます。ペンディング・キューは、ドキュメント数がしきい値を超えたかどうかを判別するために、定期的（CheckPendingCountInterval 構成パラメータの説明を参照）にポーリングされます。
SYNC_BY_TIME	SYNC は、定期的にトリガーされます。SyncInterval 構成パラメータの説明を参照してください。
SYNC_BY_PENDING_COUNT_AND_TIME	SYNC_BY_PENDING_COUNT と SYNC_BY_TIME の組合せ。

構文

```
PROCEDURE configureAutoSync;
```

DBMS_XDBT パッケージのカスタマイズ

DBMS_XDBT パッケージは、次の 2 つの方法のいずれかでカスタマイズできます。

- PL/SQL プロシージャまたは無名ブロックを使用して、関連するパッケージ変数（構成設定）を設定し、このパッケージのプロシージャを実行します。
- より一般的な方法では、既存のこのパッケージを変更するか、またはコピーとして変更することによって、適切なカスタマイズを行います。

システムは、ジョブ・キューを使用できるように構成する必要があります。ジョブは、USER_JOBS カタログ・ビューを介して参照できます。

この項では、DBMS_XDBT パッケージのカスタマイズに使用できる構成設定（パッケージ変数）について説明します。

一般的なインデックス付けの設定

次の表に、一般的なインデックス付けに関連する構成設定を示します。

パラメータ	デフォルト値	説明
IndexName	XDB$CI	ConText インデックス名。
IndexTablespace	XDB$RESINFO	ConText インデックスを構成する表およびインデックスによって使用される表領域。
IndexMemory	128M	インデックスの作成および SYNC によって使用されるメモリー。これは、MAX_INDEX_MEMORY システム・パラメータの値以下である必要があります。MAX_INDEX_MEMORY システム・パラメータ（CTX_ADMIN パッケージを参照）の値は、IndexMemory の設定値以上である必要があります。
LogFile	'XdbCtxLog'	インデックスの作成中に ROWID ロギングに使用されるログ・ファイル。LOG_DIRECTORY システム・パラメータが設定済である必要があります。ROWID ロギングを無効にするには、このパラメータを null に設定します。ROWID ロギングを有効にするには、LOG_DIRECTORY システム・パラメータ（CTX_ADMIN パッケージを参照）を設定する必要があります。

フィルタリングの設定

次の表に、XML DB 階層内のドキュメントのフィルタ処理を制御する構成設定を示します。

パラメータ	デフォルト値	説明
SkipFilter_Types	image/%, audio/%, video/%, model/%	インデックス付けの必要がない MIME タイプのリスト。
NullFilter_Types	text/plain, text/html, text/xml	INSO フィルタを使用する必要がない MIME タイプのリスト。テキストベースのドキュメントには、このパラメータを使用します。
FilterPref	XDB$CI_FILTER	フィルタ・プリファレンス名。

セクショニングおよびセクション・グループの設定

次の表に、セクショナに関連する構成設定を示します。

パラメータ	デフォルト値	説明
SectionGroup	HTML_SECTION_GROUP	使用するデフォルトのセクショナ。リポジトリが主に XML 文書を含んでいる場合は、PATH_SECTION_GROUP または AUTO_SECTION_GROUP の使用を検討してください。
SectiongroupPref	XDB$CI_SECTIONGROUP	セクション・グループ名。

ストップリストの設定

次の表に、ストップリストの構成設定を示します。

パラメータ	デフォルト値	説明
StoplistPref	XDB$CI_STOPLIST	ストップリスト名。
StopWords	0..9 'a'..'z' 'A'..'Z'	CTXSYS.DEFAULT_STOPLIST に指定されたストップワード以外のストップワードのリスト。

その他のプリファレンスの設定

次の表に、その他の索引プリファレンスの設定を示します。

パラメータ	デフォルト値	説明
DatastorePref	XDB$CI_DATASTORE	データストア・プリファレンス名。
StoragePref	XDB$CI_STORAGE	記憶域プリファレンス名。
WordlistPref	XDB$CI_WORDLIST	ワードリスト・プリファレンス名。
DefaultLexerPref	XDB$CI_DEFAULT_LEXER	デフォルトのレクサー・プリファレンスの名前。

インデックス SYNC の設定

次の表に、ConText インデックスを同期化する場合および方法を制御する設定を示します。

パラメータ	デフォルト値	説明
AutoSyncPolicy	SYNC_BY_PENDING_COUNT	インデックスの同期化のタイミングを指定します。次のいずれかの値を設定できます。 - SYNC_BY_PENDING_COUNT - SYNC_BY_TIME - SYNC_BY_PENDING_COUNT_AND_TIME
MaxPendingCount	2	インデックス SYNC がトリガーされる前の、インデックスに対する CTX_USER_PENDING キュー内のドキュメントの最大数。AutoSyncPolicy が次のどちらかの値である場合にのみ適用されます。 - SYNC_BY_PENDING_COUNT - SYNC_BY_PENDING_COUNT_AND_TIME
CheckPendingCount Interval	10 minutes	ペンディング・キューの確認の頻度（分単位）を指定します。AutoSyncPolicy が次のどちらかの値である場合にのみ適用されます。 - SYNC_BY_PENDING_COUNT - SYNC_BY_PENDING_COUNT_AND_TIME
SyncInterval	60 minutes	インデックスの同期化の頻度（分単位）を指定します。AutoSyncPolicy が次のどちらかの値である場合にのみ適用されます。 - SYNC_BY_TIME - SYNC_BY_PENDING_COUNT_AND_TIME

第 VII 部

XML データベース・サポート : SQLX

第 VII 部に含まれる章は、次のとおりです。

34

SQLX 関数および演算子

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* SQL リファレンス』

SQLX パッケージ

SQLX の説明

このリリースでサポートされている SQLX ファンクションは、SQLX 標準に準拠しています。

SQLX の関数および演算子

表 34-1 SQLX の関数および演算子の概要

関数または演算子	説明
XMLElement()（34-3 ページ）	XML 要素を作成します。
XMLForest()（34-4 ページ）	指定されたコンポーネントから XML フラグメントを作成します。
XMLColAttVal()（34-4 ページ）	XML フラグメントの作成後、各 XML フラグメントが name 属性によって column という名前を持つように、結果の XML を展開します。
ExtractValue()（34-5 ページ）	XMLType インスタンスおよび XPath 式を引数として取り、結果のノードのスカラー値を戻します。
XMLTransform()（34-6 ページ）	XMLType インスタンスおよび XSL スタイルシート（それ自体が XMLType インスタンス形式）を引数として取り、スタイルシートをインスタンスに適用して XMLType を戻します。
XMLSequence()（34-6 ページ）	入力を取り、XMLType の最上位ノードまたはカーソルの各行に対する XMLSequence 型の XML 文書を戻します。
XMLConcat()（34-7 ページ）	一連の XMLType インスタンスを入力として取り、各行の一連の要素を連結して、その連結された一連の要素を戻します。
UpdateXML()（34-8 ページ）	XMLType インスタンスと XPath の値の組を引数として取り、更新された値とともに XMLType インスタンスを戻します。

XMLElement()

説明

識別子の要素名、要素に対するオプションの属性のコレクションおよび要素の内容を構成する引数を取ります。XMLType 型のインスタンスを戻します。XMLElement は、戻された XML に属性を含めることができること以外は SYS_XMLGen と同様です。ただし、XMLFormat オブジェクトを使用したフォーマットは受け入れません。

XMLElement ファンクションは、次の項の例に示すように、通常、ネストした構造をもつ XML 文書を生成するためにネストされます。

識別子に対する値を指定する必要があります。Oracle がこの識別子を囲みタグとして使用します。この識別子は、列名または列の参照である必要はありませんが、式または null にすることはできません。

XML_attributes_clause では、*value_expr* が null の場合、その値式に対して属性は作成されません。*value_expr* の型は、オブジェクト型またはコレクションにはできません。

要素の内容を構成するオブジェクトは、XMLATTRIBUTES キーワードに従います。

- *value_expr* がスカラー式の場合、AS 句は省略できます。Oracle では、列名を要素名として使用します。
- *value_expr* がオブジェクト型またはコレクションの場合、AS 句は必須です。Oracle では、指定した *c_alias* を囲みタグとして使用します。
- *value_expr* が null の場合、その値式に要素は作成されません。

構文

```
XMLELEMENT(   elementname IN VARCHAR2,
              [XMLATTRIBUTES(),]
               value_expr,
               ...)
               RETURN XMLType

XMLATTRIBUTES( value_expr [AS identifier],
               ...)
               RETURN XMLType
```

パラメータ	説明
elementname	生成する XML 要素の名前。
value_expr	XMLATTRIBUTES のスカラー値。XMLELEMENT に対するネストした XMLELEMENT コールにできます。

XMLForest()

説明

各引数のパラメータを XML に変換後、変換した引数の連結である XML フラグメントを戻します。

- *value_expr* がスカラー式の場合、AS 句は省略できます。Oracle では、列名を要素名として使用します。
- *value_expr* がオブジェクト型またはコレクションの場合、AS 句は必須です。Oracle では、指定した *c_alias* を囲みタグとして使用します。
- *value_expr* が null の場合、その *value_expr* に要素は作成されません。

構文

```
XMLFOREST( value_expr [AS identifier],
           ...)
           RETURN XMLType
```

パラメータ	説明
value_expr	スカラー、オブジェクトまたはネストしたコール。

XMLColAttVal()

説明

XML フラグメントの作成後、各 XML フラグメントが name 属性によって column という名前を持つように、結果 XML を展開します。AS *c_alias* 句を使用して、name 属性の値を列名以外の値に変更できます。

value_expr の値を指定する必要があります。*value_expr* が null の場合、要素は戻されません。

制限： *value_expr* にオブジェクト型の列は指定できません。

構文

```
XMLCOLATTVAL( value_expr [AS identifier],
              ...)
              RETURN XMLType
```

パラメータ	説明
value_expr	スカラー、オブジェクトまたはネストしたコール。

ExtractValue()

説明

XMLType インスタンスおよび XPath 式を引数として取り、結果のノードのスカラー値を戻します。結果はシングルノードで、Text ノード、属性または要素のいずれかである必要があります。結果が要素の場合は、その要素にシングル Text ノードが子として含まれる必要があります。この値がファンクションによって戻されます。指定した XPath が複数の子を持つノードを指す場合、または指しているノードが非 Text ノードの子を持つ場合は、エラーが発生します。

XML Schema に基づく文書で、Oracle が戻り値の型を推論できる場合、適切な型のスカラー値が戻されます。推論できない場合、結果は VARCHAR2 型です。XML Schema に基づかない文書の場合、戻り型は常に VARCHAR2 です。

構文

```
EXTRACTVALUE( instance   IN  XMLType,
              xpath      IN  VARCHAR2,
             [namespace  IN  VARCHAR2])
             RETURN value
```

パラメータ	説明
instance	データの抽出元の XMLType。
xpath	文字列としての XPath 式。
namespace	名前空間宣言を指定するオプションの引数。

XMLTransform()

説明

XMLType インスタンスおよび XSL スタイルシート（それ自体が XMLType インスタンス形式）を引数として取り、スタイルシートをインスタンスに適用して XMLType を戻します。

このファンクションは、データベースからのデータの取得時に、スタイルシートに従ってデータを構成する場合に有効です。

構文

```
XMLTRANSFORM( instance    IN   XMLType,
              xslt        IN   XMLType)
              RETURN XMLType
```

パラメータ	説明
instance	変換する XMLType インスタンス。
xslt	インスタンスを変換する XSLT 文書。

XMLSequence()

説明

XMLSequence には、次の 2 つの形式があります。

- 1 つ目の形式では、XMLType インスタンスを入力として取り、XMLType の最上位のノードの VARRAY を戻します。
- 2 つ目の形式では、XMLFormat オブジェクトのオプションのインスタンスとともに、REFCURSOR インスタンスを入力として取り、カーソルの各行の XML 文書を XMLSequence 型として戻します。

XMLSequence は XMLType のコレクションを戻すため、このファンクションを TABLE 句で使用して、コレクション値を複数の行にネスト解除できます。その後、それらの行を SQL 問合せでさらに処理できます。

構文

```
XMLSEQUENCE( { input IN XMLType | input IN SYS_REFCURSOR,
               [format IN XMLFormat] })
             RETURN XMLType
```

パラメータ	説明
input	分割するデータ。
format	カーソルからの出力のフォーマットに使用されるオプションの XMLFormat。

XMLConcat()

説明

一連の XMLType インスタンスを入力として取り、各行の一連の要素を連結して、その連結した一連の要素を戻します。XMLConcat は、XMLSequence の逆です。

null 式は結果から削除されます。すべての値式が null の場合は、null を戻します。

次の表に、オプションを示します。

構文	説明
XMLCONCAT(data IN XMLSEQUENCETYPE) RETURN XMLType	XMLSequenceType 型の 1 つの引数を取り、すべての XMLType の連結である単一の XMLType を戻します。
XMLCONCAT (arg1 IN XMLType, ...) RETURN XMLType	引数から多くの XML 要素を構成する n 項ファンクションです。引数は、null 値の場合、結果から暗黙的に削除されます。すべての引数が null 値の場合は、null 値を戻します。

パラメータ	説明
data	連結する XMLSequenceType。
arg1, arg2, ...	連結する XMLType の引数。

UpdateXML()

XMLType インスタンスと XPath の値の組を引数として取り、更新された値とともに XMLType インスタンスを戻します。*XPath_string* が XML 要素の場合、対応する *value_expr* は、XMLType インスタンスである必要があります。*XPath_string* が属性または Text ノードの場合、*value_expr* は、任意のスカラー・データ型にできます。*XPath_string* と *value_expr* のターゲットのデータ型は一致する必要があります。

XML 要素を null に更新する場合、要素の属性および子が削除され、その要素は空になります。要素の Text ノードを null に更新する場合、要素のテキスト値が削除され、要素自体は残りますが空になります。

ほとんどの場合、この関数は XML 文書をメモリーに実体化して、その値を更新します。ただし、UPDATEXML は、列内で直接値を更新するように、オブジェクト・リレーショナル列に対する UPDATE 文用に最適化されます。この最適化には、次の条件が必要です。

- *XMLType_instance* は、UPDATE ... SET 句の列と同じである必要があります。
- *XPath_string* は、スカラー・コンテンツに変換する必要があります。

構文

```
UPDATEXML( instance  IN XMLType,
           [xpath     IN VARCHAR2,
           value_expr)], ...,
           [namespace IN VARCHAR2] )
          RETURN XMLType
```

パラメータ	説明
instance	更新する XMLType。
xpath	変更する XMLType の位置。
value_expr	位置を更新する値。指定する位置に応じて、XMLType または文字列にできます。
namespace	名前空間宣言を指定するオプションの引数。

第VIII部

XML データベース・サポート：データベース URIType

第 VIII 部に含まれる章は、次のとおりです。

35

データベース URI 型

この章の内容は次のとおりです。

参照： 次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』

URI のサポート

説明

Oracle9*i* は、データベースに URIRef を格納し、問い合せることが可能な URIType 型のグループをサポートします。URIType 自体は抽象オブジェクト型で、HTTPUriType、XDBUriType および DBUriType はそのサブタイプです。

URIType 列を作成し、それに、DBUriType、XDBUriType または HTTPUriType のインスタンスを格納できます。

また、URIType の独自のサブタイプを定義して、様々な URL プロトコルを処理することもできます。

Oracle9*i* は URIFactory パッケージも提供します。これは、http:// や /oradb などの接頭辞をスキャンすることによって、これらの URIType のインスタンスを自動的に生成可能なファクトリ・メソッドとして使用できます。また、ユーザー独自のサブタイプを登録し、ユーザーがサポートする接頭辞を指定することもできます。たとえば、gopher プロトコルを処理するためのサブタイプを URIFactory に登録し、gopher:// という接頭辞を持つ URL をそのサブタイプで処理するように指定できます。その結果、URIFactory は、接頭辞 gopher で始まるすべての URL に対して、登録したサブタイプのインスタンスを生成します。

表 35-1 URI をサポートする型

型	説明
URIType スーパータイプ（35-3 ページ）	データベース URI 型のベース型である抽象型。
HTTPUriType サブタイプ（35-7 ページ）	URI 型の HTTP 実装。
DBUriType サブタイプ（35-12 ページ）	データベース・オブジェクトを参照する URI 型の実装。
XDBUriType サブタイプ（35-18 ページ）	Oracle XML DB オブジェクトを参照する URI 型の実装。
URIFactory パッケージ（35-23 ページ）	指定された URI に適切な URI 型のインスタンスを生成するパッケージ。

URIType スーパータイプ

URIType の説明

URIType は、抽象スーパータイプです。これは、URI が指す値を取得するための、一連の標準ファンクションを提供します。実際のプロトコル実装は、この型のサブタイプによって定義される必要があります。

この型のインスタンスは直接作成できません。ただし、この型の列を作成し、それにサブタイプ・インスタンスを格納することはできます。

次に例を示します。

```
    create table uri_tab ( url uritype);

insert into uri_tab values
  (httpuritype.createuri('http://www.oracle.com'));
insert into uri_tab values
  (dburitype.createuri('/SCOTT/EMPLOYEE/ROW[ENAME="Jack"]'));
```

これで、実際に格納されている URL のインスタンスがわからなくても、列から選択できます。次のコードの行は、HTTP の URL と DBUriRef の両方を取得します。

```
select e.url.getclob() from uri_tab e;
```

UriType のファンクションおよびプロシージャ

表 35-2 UriType のファンクションおよびプロシージャの概要

ファンクション	説明
getBlob()（35-4 ページ）	URL によって指定されたアドレスに格納されている BLOB を戻します。
getClob()（35-4 ページ）	URL によって指定されたアドレスに格納されている CLOB を戻します。
getContentType()（35-5 ページ）	URIType インスタンス内に格納された、エスケープされた形式の URL を戻します。
getExternalUrl()（35-5 ページ）	URIType インスタンス内に格納された、エスケープされた形式の URL を戻します。
getUrl()（35-6 ページ）	URIType インスタンス内に格納された、エスケープされていない形式の URL を戻します。
getXML()（35-6 ページ）	URL によって指定されたアドレスに格納されている XMLType を戻します。

getBlob()

説明

URL によって指定されたアドレスに格納されている BLOB を戻します。このファンクションは、サブタイプ・インスタンスでオーバーライドできます。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getBlob() RETURN blob;	URL によって指定されたアドレスに格納されている BLOB を戻します。
MEMBER FUNCTION getBlob(content OUT VARCHAR2) RETURN blob;	URL およびコンテンツ・タイプによって指定されたアドレスに格納されている BLOB を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

getClob()

説明

URL によって指定されたアドレスに格納されている CLOB を戻します。このファンクションは、サブタイプ・インスタンスでオーバーライドできます。このファンクションは、永続 CLOB または一時 CLOB のいずれかを戻します。一時 CLOB が戻される場合は、その一時 CLOB を解放する必要があります。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getClob() RETURN clob;	URL によって指定されたアドレスに格納されている CLOB を戻します。
MEMBER FUNCTION getClob(content OUT VARCHAR2) RETURN clob;	URL およびコンテンツ・タイプによって指定されたアドレスに格納されている CLOB を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

getContentType()

説明

URI が指すドキュメントのコンテンツ・タイプを戻します。このファンクションは、サブタイプ・インスタンスでオーバーライドできます。このファンクションは、コンテンツ・タイプを VARCHAR2 として戻します。

構文

```
MEMBER FUNCTION getContentType RETURN VARCHAR2;
```

getExternalUrl()

説明

URIType インスタンス内に格納された、エスケープされた形式の URL を戻します。サブタイプ・インスタンスは、このメンバー・ファンクションをオーバーライドして、その他のセマンティクスを提供します。たとえば、HTTPUriType ファンクションは、URI 自体に接頭辞 http:// を格納しません。外部 URL を生成する場合、HTTPUriType は接頭辞を追加し、外部 URL を生成します。そのため、URIType インスタンスに存在する属性を使用するのではなく、getExternalUrl ファンクションまたは getUrl ファンクションを使用して URL 値を取得する必要があります。

構文

```
MEMBER FUNCTION getExternalUrl RETURN varchar2;
```

getUrl()

説明

URIType インスタンス内に格納された、エスケープされていない形式の URL を戻します。サブタイプ・インスタンスは、このメンバー・ファンクションをオーバーライドして、その他のセマンティクスを提供します。たとえば、HTTPUriType ファンクションは、URI 自体に接頭辞 http:// を格納しません。外部 URL を生成する場合、HTTPUriType は接頭辞を追加し、外部 URL を生成します。そのため、URIType インスタンスに存在する属性を使用するのではなく、getExternalUrl ファンクションまたは getUrl ファンクションを使用して URL 値を取得する必要があります。

構文

```
MEMBER FUNCTION getUrl RETURN varchar2;
```

getXML()

説明

URL によって指定されたアドレスに格納されている XMLType を戻します。このファンクションは、サブタイプ・インスタンスでオーバーライドできます。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getXML RETURN XMLType;	URL によって指定されたアドレスに格納されている XMLType を戻します。
MEMBER FUNCTION getXML(content OUT VARCHAR2) RETURN XMLType;	URL およびコンテンツ・タイプによって指定されたアドレスに格納されている XMLType を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

HTTPUriType サブタイプ

HTTPUriType の説明

HTTPUriType は、HTTP プロトコルをサポートする、URIType のサブタイプです。HTTPUriType は、UTL_HTTP パッケージを使用して、HTTP の URL にアクセスします。今回のリリースでは、プロキシおよび保護 Wallet はサポートされていません。

次の例では、URI 表を作成して HTTP インスタンスを格納します。

```
create table uri_tab ( url httpuritype);
```

HTTP インスタンスを挿入します。

```
insert into uri_tab values
    (httpuritype.createUri('http://www.oracle.com'));
```

HTML を生成します。

```
select e.url.getclob() from uri_tab e;
```

HTTPUriType のファンクションおよびプロシージャ

表 35-3 HTTPUriType のファンクションおよびプロシージャの概要

ファンクション	説明
createUri()（35-8 ページ）	指定された URI から HTTPUriType インスタンスを作成します。
getBlob()（35-8 ページ）	URL によって指定されたアドレスに格納されている BLOB を戻します。
getClob()（35-9 ページ）	URL によって指定されたアドレスに格納されている CLOB を戻します。
getContentType()（35-9 ページ）	URI が指すドキュメントのコンテンツ・タイプを戻します。
getExternalUrl()（35-10 ページ）	URIType インスタンス内に格納された、エスケープされた形式の URL を戻します。
getUrl()（35-10 ページ）	URIType インスタンス内に格納された、エスケープされていない形式の URL を戻します。
getXML()（35-10 ページ）	URL によって指定されたアドレスに格納されている XMLType を戻します。
httpUriType()（35-11 ページ）	指定された URI から HTTPUriType インスタンスを作成します。

createUri()

説明

HTTPUriType インスタンスを作成します。HTTPUriType インスタンスは、格納された URL に接頭辞 http:// を含めません。

構文

```
STATIC FUNCTION createUri(url IN varchar2) RETURN HttpUriType;
```

パラメータ	IN / OUT	説明
url	(IN)	有効な HTTP の URL を含む URL 文字列。URL 文字列は、エスケープされた形式である必要があります。たとえば、非 URL 文字は、これらの文字の UTF-8 エンコーディングの 16 進数値として表されます。

getBlob()

説明

HTTP の URL によって指定されたアドレスに格納されている BLOB を戻します。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getBlob RETURN blob;	HTTP の URL によって指定されたアドレスに格納されている BLOB を戻します。
MEMBER FUNCTION getBlob(content OUT VARCHAR2) RETURN blob;	HTTP の URL およびコンテンツ・タイプによって指定されたアドレスに格納されている BLOB を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

getClob()

説明

HTTP の URL によって指定されたアドレスに格納されている CLOB を戻します。このファンクションは、永続 CLOB または一時 CLOB のいずれかを戻します。一時 CLOB が戻される場合は、その一時 CLOB を解放する必要があります。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getClob RETURN clob;	HTTP の URL によって指定されたアドレスに格納されている CLOB を戻します。
MEMBER FUNCTION getClob(content OUT VARCHAR2) RETURN clob;	HTTP の URL およびコンテンツ・タイプによって指定されたアドレスに格納されている CLOB を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

getContentType()

説明

URI が指すドキュメントのコンテンツ・タイプを戻します。

構文

```
MEMBER FUNCTION getContentType() RETURN VARCHAR2;
```

getExternalUrl()

説明

HTTPUriType インスタンス内に格納された、エスケープされた形式の URL を戻します。サブタイプ・インスタンスは、このメンバー・ファンクションをオーバーライドして、その他のセマンティクスを提供します。HTTPUriType ファンクションは、URI 自体に接頭辞 http:// を格納しません。HTTPUriType は、外部 URL を生成する場合、接頭辞を追加して生成します。

構文

```
MEMBER FUNCTION getExternalUrl RETURN varchar2;
```

getUrl()

説明

HTTPUriType インスタンス内に格納された、エスケープされていない形式の URL を戻します。

構文

```
MEMBER FUNCTION getUrl RETURN varchar2;
```

getXML()

説明

URL によって指定されたアドレスに格納されている XMLType を戻します。アドレスが有効な XML 文書を指していない場合は、エラーが発生します。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getXML RETURN XMLType;	URL によって指定されたアドレスに格納されている XMLType を戻します。
MEMBER FUNCTION getXML(content OUT VARCHAR2) RETURN XMLType;	URL およびコンテンツ・タイプによって指定されたアドレスに格納されている XMLType を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

httpUriType()

説明

HTTPUriType インスタンスを作成します。HTTPUriType インスタンスは、格納された URL に接頭辞 http:// を含めません。

構文

```
CONSTRUCTOR FUNCTION httpUriType(url IN varchar2);
```

パラメータ	IN / OUT	説明
url	(IN)	有効な HTTP の URL を含む URL 文字列。URL 文字列は、エスケープされた形式である必要があります。たとえば、非 URL 文字は、これらの文字の UTF-8 エンコーディングの 16 進数値として表されます。

DBUriType サブタイプ

DBUriType の説明

DBUriType は、DBUriRef をサポートする URIType のサブタイプです。DBUriRef は、データベース内の任意の行または行 - 列のデータを参照するために使用可能なデータベース内 URL です。URL は、データベースの XML 表示に対する XPath 式として指定されます。スキーマは、表およびビューを含む要素になります。これらの表およびビューは、さらに内部に行および列を含みます。

たとえば、ユーザー scott は次のような仮想ドキュメントを参照できます。

```
<?xml version="1.0"?>
<ORADB>
  <SCOTT>
    <EMPLOYEE>
      <ROWSET>
         <ROW>
          <EMPNO>100</EMPNO>
          <ENAME>John</ENAME>
          <ADDRESS>
             <STREET>100 Main Street</STREET>
             <CITY>Jacksonville</CITY>
             <STATE>FL</STATE>
             <ZIP>32607</ZIP>
           </ADDRESS>
         </ROW>
         <ROW>
           <EMPNO>200</EMPNO>
           <ENAME>Jack</ENAME>
           <ADDRESS>
             <STREET>200 Front Street</STREET>
             <CITY>San Francisco</CITY>
             <STATE>CA</STATE>
             <ZIP>94011</ZIP>
           </ADDRESS>
         </ROW>
       </ROWSET>
     </EMPLOYEE>
   </SCOTT>
 </ORADB>
```

そのため、従業員表内の STATE 属性を参照するには、DBUriRef を次のとおり表現します。

```
/ORADB/SCOTT/EMPLOYEE/ROW[ENAME="Jack"]/ADDRESS/STATE
```

DBUriType を使用し、これらのインスタンスを作成して列内に格納できます。

表を作成します。

```
create table dburi_tab (dburl dburitype);
```

値を挿入します。

```
insert into dburi_tab values (
   dburitype.createUri(
        '/ORADB/SCOTT/EMPLOYEE/ROW[ENAME="Jack"]/ADDRESS/STATE'));
select e.dburl.getclob() from dburi_tab e;
```

次の結果が戻されます。

```
<?xml version="1.0"?>
<STATE>CA</STATE>
```

また、SQL ファンクション SYS_DBURIGEN を使用して、DBUriRef を動的に生成することもできます。

たとえば、次に示すとおり、STATE 属性に対する DBUriRef を生成できます。

```
select sys_dburigen( e.ename, e.address.state) AS urlcol
     from scott.employee e;
```

DBUriType のファンクションおよびプロシージャ

表 35-4 DBUriType のファンクションおよびプロシージャの概要

ファンクション	説明
createUri()（35-14 ページ）	DBUriType インスタンスを作成します。
DBUriType()（35-15 ページ）	指定された URI から DBUriType のインスタンスを作成します。
getBlob()（35-15 ページ）	DBUriType インスタンスによって指定されたアドレスに格納されている BLOB を戻します。
getClob()（35-16 ページ）	DBUriType インスタンスによって指定されたアドレスに格納されている CLOB を戻します。
getContentType()（35-16 ページ）	URI が指すドキュメントのコンテンツ・タイプを戻します。
getExternalUrl()（35-17 ページ）	DBUriType インスタンス内に格納された、エスケープされた形式の URL を戻します。
getUrl()（35-17 ページ）	DBUriType インスタンス内に格納された、エスケープされていない形式の URL を戻します。
getXML()（35-17 ページ）	URL によって指定されたアドレスに格納されている XMLType を戻します。

createUri()

説明

DBUriType インスタンスを作成します。指定された URL を解析して、DBUriType インスタンスを作成します。

構文

```
STATIC FUNCTION createUri( url IN varchar2) RETURN DBUriType;
```

パラメータ	IN / OUT	説明
url	(IN)	有効な DBUriRef を含む、エスケープされた形式の URL 文字列。

DBUriType()

説明

DBUriType インスタンスを作成します。

構文

```
CONSTRUCTOR FUNCTION DBUriType( url IN varchar2, spare IN raw := null);
```

パラメータ	IN / OUT	説明
url	(IN)	有効な DBUriRef を含む URL 文字列。URL 文字列は、エスケープされた形式である必要があります。たとえば、非 URL 文字は、これらの文字の UTF-8 エンコーディングの 16 進数値として表されます。

getBlob()

説明

URL によって指定されたアドレスに格納されている BLOB を戻します。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getBlob RETURN blob;	URL によって指定されたアドレスに格納されている BLOB を戻します。
MEMBER FUNCTION getBlob(content OUT VARCHAR2) RETURN blob;	URL およびコンテンツ・タイプによって指定されたアドレスに格納されている BLOB を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

getClob()

説明

DBUriType インスタンスによって指定されたアドレスに格納されている CLOB を戻します。一時 CLOB が戻される場合は、その一時 CLOB を解放する必要があります。戻されるドキュメントは、XML 文書またはテキスト・ドキュメントです。DBUriRef が XPath の要素を識別すると、結果は整形式の XML 文書になります。また、（text() ファンクションを使用して）Text ノードを識別すると、列または属性の内容のテキストのみが戻されます。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getClob RETURN clob;	DBUriType インスタンスによって指定されたアドレスに格納されている CLOB を戻します。
MEMBER FUNCTION getClob(content OUT VARCHAR2) RETURN clob;	DBUriType インスタンスおよびコンテンツ・タイプによって指定されたアドレスに格納されている CLOB を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

getContentType()

説明

URI が指すドキュメントのコンテンツ・タイプを戻します。

構文

```
MEMBER FUNCTION getContentType RETURN VARCHAR2;
```

getExternalUrl()

説明

DBUriType インスタンス内に格納された、エスケープされた形式の URL を戻します。DBUriType を処理する DBUri サーブレット URL は、エスケープされた URL を Web ページで使用する前に追加する必要があります。

構文

```
MEMBER FUNCTION getExternalUrl RETURN varchar2;
```

getUrl()

説明

DBUriType インスタンス内に格納された、エスケープされていない形式の URL を戻します。

構文

```
MEMBER FUNCTION getUrl RETURN varchar2;
```

getXML()

説明

URL によって指定されたアドレスに格納されている XMLType を戻します。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getXML RETURN XMLType;	URL によって指定されたアドレスに格納されている XMLType を戻します。
MEMBER FUNCTION getXML(content OUT VARCHAR2) RETURN XMLType;	URL およびコンテンツ・タイプによって指定されたアドレスに格納されている XMLType を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

XDBUriType サブタイプ

XDBUriType の説明

XDBUriType は、URIType の新しいサブタイプです。XDBUriType を使用すると、Oracle XML DB 階層のドキュメントを、表内のすべての URIType 列に埋込み可能な URI として公開できます。URI の URL 部分は、その URI が参照する XML 文書の階層名です。オプションのフラグメント部分には、XPath 構文を使用します。この部分は、「#」によって URL 部分と区切られます。フラグメントを指定するためのより一般的な XPointer 構文は、現在サポートされていません。

XDBUriType のファンクションおよびプロシージャ

表 35-5 XDBUriType のファンクションおよびプロシージャの概要

ファンクション	説明
createUri()（35-19 ページ）	指定された URL に対応する URIType を戻します。
getBlob()（35-19 ページ）	XDBUriType インスタンスによって指定されたドキュメントのコンテンツに対応する BLOB を戻します。
getClob()（35-16 ページ）	XDBUriType インスタンスによって指定されたドキュメントのコンテンツに対応する CLOB を戻します。
getContentType()（35-20 ページ）	URI が指すドキュメントのコンテンツ・タイプを戻します。
getExternalUrl()（35-17 ページ）	XDBUriType インスタンス内に格納された、エスケープされた形式の URL を戻します。
getUrl()（35-17 ページ）	XDBUriType インスタンス内に格納された、エスケープされていない形式の URL を戻します。
getXML()（35-21 ページ）	URL インスタンスによって指定されたドキュメントのコンテンツに対応する XMLType を戻します。
XDBUriType()（35-22 ページ）	指定された URI から XDBUriType インスタンスを作成します。

createUri()

説明

XDBUriType インスタンスを作成します。指定された URL を解析して、XDBUriType インスタンスを作成します。

構文

```
STATIC FUNCTION createUri(url IN varchar2) RETURN XDBUriType;
```

パラメータ	IN / OUT	説明
url	(IN)	有効な XDBUriRef を含む、エスケープされた形式の URL 文字列。

getBlob()

説明

XDBUriType インスタンスによって指定されたアドレスに格納されている BLOB を戻します。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getBlob RETURN blob;	URL によって指定されたアドレスに格納されている BLOB を戻します。
MEMBER FUNCTION getBlob(content OUT VARCHAR2) RETURN blob;	URL およびコンテンツ・タイプによって指定されたアドレスに格納されている BLOB を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

getClob()

説明

XDBUriType インスタンスによって指定されたアドレスに格納されている CLOB を戻します。一時 CLOB が戻される場合は、その一時 CLOB を解放する必要があります。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getClob RETURN clob;	DBUriType インスタンスによって指定されたアドレスに格納されている CLOB を戻します。
MEMBER FUNCTION getClob(content OUT VARCHAR2) RETURN clob;	DBUriType インスタンスおよびコンテンツ・タイプによって指定されたアドレスに格納されている CLOB を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

getContentType()

説明

URI が指すドキュメントのコンテンツ・タイプを戻します。このファンクションは、コンテンツ・タイプを VARCHAR2 として戻します。

構文

```
MEMBER FUNCTION getContentType RETURN VARCHAR2;
```

getExternalUrl()

説明

XDBUriType インスタンス内に格納された、エスケープされた形式の URL を戻します。

構文

```
MEMBER FUNCTION getExternalUrl RETURN varchar2;
```

getUrl()

説明

XDBUriType インスタンス内に格納された、エスケープされていない形式の URL を戻します。

構文

```
MEMBER FUNCTION getUrl RETURN varchar2;
```

getXML()

説明

URL によって指定されたアドレスに格納されている XMLType を戻します。次の表に、オプションを示します。

構文	説明
MEMBER FUNCTION getXML RETURN XMLType;	URL によって指定されたアドレスに格納されている XMLType を戻します。
MEMBER FUNCTION getXML(content OUT VARCHAR2) RETURN XMLType;	URL およびコンテンツ・タイプによって指定されたアドレスに格納されている XMLType を戻します。

パラメータ	IN / OUT	説明
content	(OUT)	URI が指すドキュメントのコンテンツ・タイプ。

XDBUriType()

説明

XDBUriType インスタンスを作成します。

構文

```
CONSTRUCTOR FUNCTION XDBUriType( url IN varchar2);
```

パラメータ	IN / OUT	説明
url	(IN)	有効な XDBUriRef を含む URL 文字列。URL 文字列は、エスケープされた形式である必要があります。たとえば、非 URL 文字は、これらの文字の UTF-8 エンコーディングの 16 進数値として表されます。

URIFactory パッケージ

URIFactory の説明

URIFactory パッケージには、ファクトリ・メソッドが含まれます。このファクトリ・メソッドを使用すると、プログラムで実装をハードコードすることなく、URIType の適切なインスタンスを生成できます。

URIFactory パッケージは、Oracle9*i* で現在サポートされていない他の様々なプロトコルを処理するために、URIType の新しいサブタイプを登録する機能も提供します。たとえば、新しいプロトコル ecom:// を作成し、そのプロトコルを処理するための URIType のサブタイプを定義して、URIFactory にそのサブタイプを登録できます。登録後は、すべてのファクトリ・メソッドが、接頭辞 ecom:// を検出するたびに、その新しいサブタイプのインスタンスを生成します。

次に例を示します。

```
create table url_tab (urlcol varchar2(20));
```

HTTP 参照を挿入します。

```
insert into url_tab values ('http://www.oracle.com');
```

DBUriRef を挿入します。

```
insert into url_tab values ('/SCOTT/EMPLOYEE/ROW[ENAME="Jack"]');
```

ecom:// という新しいプロトコルを処理する新しい型を作成します。

```
create type EComUriType under UriType
(
  overriding member function getClob() return clob,
  overriding member function getBlob() return blob,
     -- not supported
  overriding member function getExternalUrl() return varchar2,
  overriding member function getUrl() return varchar2,

  -- MUST NEED THIS for registering with the url handler
  static member function createUri(url in varchar2)
         return EcomUriType
);
```

新しいプロトコル・ハンドラを登録します。

```
begin
     -- register a new handler for ecom:// prefixes. The handler
     -- type name is ECOMURITYPE, schema is SCOTT
     -- Ignore the prefix case, when comparing and also strip
```

```
        -- the prefix before calling the createUri function
        urifactory.registerHandler('ecom://','SCOTT','
                                   ECOMURITYPE', true,true);
end;

insert into url_tab values ('ECOM://company1/company2=22/comp');
```

ファクトリを使用して、インスタンスを生成します。

```
select urifactory.getUri(urlcol) from url_tab;
```

次のインスタンスが生成されます。

```
HttpUriType('www.oracle.com');
        -- an Http uri type instance

DBUriType('/SCOTT/EMPLOYEE/ROW[ENAME="Jack"],null);
        -- a DBUriType

EComUriType('company1/company2=22/comp');
        -- a EComUriType instance
```

URIFactory のファンクションおよびプロシージャ

表 35-6 URIFactory のファンクションおよびプロシージャの概要

メソッド	説明
getUri()（35-25 ページ）	指定された URL 文字列の適切な URL ハンドラを戻します。
escapeUri()（35-25 ページ）	エスケープされた形式で URL を戻します。
unescapeUri()（35-26 ページ）	エスケープされていない形式で URL を戻します。
registerUrlHandler()（35-26 ページ）	特定の URL を処理するための、特定の型名を登録します。
unRegisterUrlHandler()（35-27 ページ）	URL ハンドラの登録を解除します。

getUri()

説明

指定された URL 文字列の適切な URL ハンドラを戻します。また、プロトコルを処理できる URIType のサブタイプ・インスタンスを戻します。デフォルトでは、URL が解決できない場合、常に XDBUriType インスタンスが作成されます。URL ハンドラは、registerUrlHandler() ファンクションを使用して、特定の接頭辞に対して登録できます。接頭辞が一致すると、getUrl() がそのサブタイプを使用します。

構文

```
FUNCTION getUrl(url IN Varchar2) RETURN UriType;
```

パラメータ	IN / OUT	説明
url	(IN)	有効な HTTP の URL を含む、エスケープされた形式の URL 文字列。

escapeUri()

説明

エスケープされた形式で URL を戻します。サブタイプ・インスタンスは、このメンバー ファンクションをオーバーライドして、その他のセマンティクスを提供します。たとえば、HTTPUriType ファンクションは、URL 自体に接頭辞 http:// を格納しません。外部 URL を生成する場合、HTTPUriType は接頭辞を追加し、外部 URL を生成します。そのため、URIType に存在する属性を使用するのではなく、getExternalUrl ファンクションまたは getUrl ファンクションを使用して URL 値を取得する必要があります。

構文

```
FUNCTION escapeUri(url IN Varchar2) RETURN varchar2;
```

パラメータ	IN / OUT	説明
url	(IN)	エスケープされた形式で戻される URL 文字列。

unescapeUri()

説明

エスケープされていない形式で URL を戻します。このファンクションは、escapeUri ファンクションの逆です。このファンクションは、文字列をスキャンして、16 進数の非 URL 文字を同等の UTF-8 文字に変換します。戻り型が VARCHAR2 であるため、この文字は、データベース・キャラクタ・セットによって定義されている同等の文字に変換されます。

構文

```
FUNCTION unescapeUri(url IN Varchar2) RETURN varchar2;
```

パラメータ	IN / OUT	説明
url	(IN)	エスケープされていない形式で戻される URL 文字列。

registerUrlHandler()

説明

特定の URL を処理するための特定の型名を登録します。指定された型は有効であり、URIType のサブタイプまたはそのサブタイプの 1 つである必要があります。また、次の静的メンバー・ファンクションを実装する必要があります。

```
STATIC FUNCTION createUri(url IN varchar2) RETURN <typename>;
```

このファンクションは、getUrl() ファンクションによって、その型のインスタンスを生成するためにコールされます。stripprefix パラメータは、このファンクションをコールする前にその接頭辞を削除する必要があることを示します。

構文

```
PROCEDURE registerUrlHandler( prefix IN varchar2,
                              schemaName IN varchar2,
                              typename IN varchar2,
                              ignoreCase IN boolean := true,
                              stripprefix IN boolean := true);
```

パラメータ	IN / OUT	説明
prefix	(IN)	処理する接頭辞（http:// など）。
schemaName	(IN)	型が存在するスキーマ名。大 / 小文字が区別されます。
typename	(IN)	URL を処理する型名。大 / 小文字が区別されます。
ignoreCase	(IN)	接頭辞の一致検索時に、大 / 小文字を無視します。
stripprefix	(IN)	型のインスタンスを生成する前に、接頭辞を削除します。

unRegisterUrlHandler()

説明

URL ハンドラの登録を解除します。これは、ユーザーが登録したハンドラの接頭辞のみを登録解除し、システムで事前定義されている http:// などの接頭辞は登録解除しません。

構文

```
PROCEDURE unregisterUrlHandler(prefix IN varchar2);
```

パラメータ	IN / OUT	説明
prefix	(IN)	登録を解除する接頭辞。

索引

か

く

は