Oracle®

XML リファレンス

10*g*（9.0.4）

部品番号 : B12342-01

2004 年 6 月

ORACLE®

Oracle XML リファレンス , 10*g*（9.0.4）

部品番号 : B12342-01

原本名 : Oracle XML Reference, 10*g* (9.0.4)

原本部品番号 : B10926-01

原本著者 : Roza Leyderman

目次

2 Document Object Model（DOM）

3 Java 用の XML 処理（JAXP）

4 Java 用の XSLT 処理

5 XML Schema の処理

6 Java 用の XML クラス生成

7 XML SQL Utility for Java

8 XSQL Pages Publishing Framework for Java

9 TransX Utility for Java

10 Oracle XML JavaBeans

11 Java の圧縮

12 Simple Object Access Protocol（SOAP）

17 Oracle XML Class Generator for C++

索引

はじめに

ここでは、次の項目について説明します。

対象読者

『Oracle XML リファレンス』は、最新のインターネット・テクノロジ、特に XML テクノロジの Oracle サポートに関心を持つすべてのユーザー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」

この部の各章では、JAVA API for XML について説明します。

第 1 章「XML Parser for Java」

第 2 章「Document Object Model（DOM）」

第 3 章「Java 用の XML 処理（JAXP）」

第 4 章「Java 用の XSLT 処理」

第 5 章「XML Schema の処理」

第 6 章「Java 用の XML クラス生成」

第 7 章「XML SQL Utility for Java」

第 8 章「XSQL Pages Publishing Framework for Java」

第 9 章「TransX Utility for Java」

第 10 章「Oracle XML JavaBeans」

第 11 章「Java の圧縮」

第 12 章「Simple Object Access Protocol（SOAP）」

第 II 部「XML の C サポート」

この部の各章では、C API for XML について説明します。

第 13 章「XML Schema Processor for C」

第 14 章「XML Parser for C」

第 III 部「XML の C++ サポート」

この部の各章では、C++ API for XML について説明します。

第 15 章「XML Schema Processor for C++」

第 16 章「XML Parser for C++」

第 17 章「Oracle XML Class Generator for C++」

関連ドキュメント

詳細は、データベース・ドキュメント・ライブラリの各種タイトルを参照してください。

リリース・ノート、インストール関連ドキュメント、ホワイト・ペーパーまたはその他の関連ドキュメントは、OTN-J（Oracle Technology Network Japan）から、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。登録は、次の Web サイトから無償で行えます。

`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している場合は、次の URL で OTN -J Web サイトのドキュメントのセクションに直接接続できます。

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

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てい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。

本文の表記規則

本文では、特定の項目が一目でわかるように、次の表記規則を使用します。次の表に、その規則と使用例を示します。

規則	意味	例
太字	太字は、本文中で定義されている用語および用語集に記載されている用語を示します。	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、システム指定の要素を示します。このような要素には、パラメータ、権限、データ型、RMAN 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、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作成できます。 `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指定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あります。また、ユーザーが指定するデータベース・オブジェクトとデータベース構造、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。	`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作成します。 `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。
	注意 : プログラム要素には、大文字と小文字を組み合せて使用するものもあります。これらの要素は、記載されているとおりに入力してください。	`JRepUtil` クラスが次のメソッドを実装します。

規則	意味	例
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の例です。次のように固定幅フォントで表示され、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とその使用例を示します。

規則	意味	例
[ ]	大カッコは、カッコ内の項目を任意に選択することを表します。大カッコは、入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコは、カッコ内の項目のうち、1 つが必須であることを表します。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複数の選択項目の区切りに使用します。項目のうちの 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]
. . .	水平の省略記号は、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。	
その他の記号	大カッコ、中カッコ、縦線および省略記号以外の記号は、記載されているとおりに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	イタリック体は、特定の値を指定する必要があるプレースホルダや変数を示します。	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*

規則	意味	例

大文字	大文字は、システム指定の要素を示します。これらの要素は、ユーザー定義の要素と区別するために大文字で示されます。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、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、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名などです。	`SELECT last_name, employee_id FROM employees;` `sqlplus hr/hr`
	注意 : プログラム要素には、大文字と小文字を組み合わせて使用するものもあります。これらの要素は、記載されているとおりに入力してください。	`CREATE USER mjones IDENTIFIED BY ty3MU9;`

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 解析は、XML 文書の処理を行いこれらの文書に含まれる情報を様々な API で利用可能にします。これにより、既存のアプリケーションを XML に対応するための拡張が容易になります。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

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

DefaultXMLDocumentHandler クラス

このクラスは、XMLDocumentHandler インタフェースのデフォルト動作を実装します。アプリケーション開発者は、このインタフェースの一部を実装する場合に、このクラスを拡張できます。

構文

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

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を構成します。
cDATASection()（1-3 ページ）	CDATA セクションの通知を受け取ります。
comment()（1-3 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）	接頭辞 -URI マッピング有効範囲の終わりです。
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()（1-5 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()（1-6 ページ）	テキスト宣言の通知を受け取ります。
setXMLDecl()（1-7 ページ）	XML 宣言の通知を受け取ります。
setXMLSchema()（1-7 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()（1-8 ページ）	要素の始まりの通知を受け取ります。
startPrefixMapping()（1-9 ページ）	接頭辞 -URI 名前空間マッピング有効範囲の始まりです。

DefaultXMLDocumentHandler()

デフォルトのドキュメントを構成します。

構文

```
public  DefaultXMLDocumentHandler();
```

cDATASection()

CDATA セクションの通知を受け取ります。パーサーは、CDATA セクションが検出されるたびに、このメソッドをコールします。発生する org.xml.sax.SAXException は SAX の例外であり、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
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

comment()

コメントの通知を受け取ります。パーサーは、コメントが検出されるたびにこのメソッドをコールします。コメントは、主なドキュメント要素の前後で発生する場合があることに注意してください。発生する org.xml.sax.SAXException は SAX の例外であり、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()

DTD の終わりの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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()

要素の終わりの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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、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	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か、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（修飾名が使用不可の場合は空の文字列）

endPrefixMapping()

接頭辞 -URI マッピング有効範囲の終わりです。発生する org.xml.sax.SAXException は SAX の例外であり、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()

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()

DTD の通知をすぐ後に受け取ることができるように、DTD を設定します。発生する org.xml.sax.SAXException は SAX の例外であり、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()

XMLError ハンドラの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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 err);
```

パラメータ	説明
err	XMLError オブジェクト

setHandler()

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。発生する org.xml.sax.SAXException は SAX の例外であり、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()

テキスト宣言の通知を受け取ります。パーサーは、テキスト宣言ごとにこのメソッドをコールします。発生する org.xml.sax.SAXException は SAX の例外であり、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（指定されていない場合は NULL）

setXMLDecl()

XML 宣言の通知を受け取ります。 パーサーは、XML 宣言ごとに、このメソッドをコールします。発生する org.xml.sax.SAXException は SAX の例外であり、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（指定されていない場合は NULL）

setXMLSchema()

XMLSchema オブジェクトの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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()

スキップされたエンティティの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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。

startElement()

要素の始まりの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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、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	NSName オブジェクト
attlist	要素の SAXAttrList
namespaceURI	名前空間 URI（要素に名前空間 URI が設定されていないか、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）

パラメータ	説明
qName	接頭辞付き修飾名（修飾名が使用不可の場合は空の文字列）
atts	要素に結び付けられた属性。属性がない場合は、空の属性オブジェクトになります。

startPrefixMapping()

接頭辞 -URI 名前空間マッピング有効範囲の始まりです。org.xml.sax.SAXException が発生し、さらに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 URI

DocumentBuilder クラス

このクラスは、SAX 2.0 イベントから DOM ツリーを構築するために XMLDocumentHandler（廃止）および ContentHandler を実装します。下位互換性のために XMLDocumentHandler イベントがサポートされています。

構文

```
public class DocumentBuilder
  oracle.xml.parser.v2.DocumentBuilder
```

表 1-2 DocumentBuilder のメソッドの概要

メソッド	説明
DocumentBuilder()（1-12 ページ）	XMLDocumentHandler として使用可能なドキュメント・ビルダを作成します。
attributeDecl()（1-12 ページ）	属性型の宣言を通知します。
cDATASection()（1-13 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()（1-13 ページ）	要素内の文字データの通知を受け取ります。
comment()（1-14 ページ）	コメントの通知を受け取ります。
elementDecl()（1-14 ページ）	要素型の宣言を通知します。
endCDATA()（1-15 ページ）	CDATA セクションの終わりを通知します。
endDoctype()（1-15 ページ）	DTD の終わりの通知を受け取ります。
endDocument()（1-15 ページ	ドキュメントの終わりの通知を受け取ります。
endDTD()（1-15 ページ）	DTD 宣言の終わりを通知します。
endElement()（1-16 ページ）	要素の終わりの通知を受け取ります。
endEntity()（1-16 ページ）	エンティティの終わりを通知します。
externalEntityDecl()（1-17 ページ）	解析した外部エンティティ宣言を通知します。
getCurrentNode()（1-17 ページ）	現在作成中のノードを戻します。
getDocument()（1-17 ページ）	作成中のドキュメントを取得します。

表 1-2 DocumentBuilder のメソッドの概要（続き）

メソッド	説明
ignorableWhitespace()（1-18 ページ）	要素内容内の無視できる空白の通知を受け取ります。
internalEntityDecl()（1-18 ページ）	内部エンティティ宣言を通知します。
processingInstruction()（1-19 ページ）	処理命令の通知を受け取ります。
retainCDATASection()（1-19 ページ）	CDATA セクションを保持するフラグを設定します。
setDebugMode()（1-19 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()（1-20 ページ）	DTD の通知を受け取ります。DTD を設定します。
setDocumentLocator()（1-20 ページ）	ドキュメント・イベントの 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、別のドキュメント・イベントで使用するためにロケータを格納する場合、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()（1-20 ページ）	オプションの NodeFactory を設定します。これは、カスタム DOM ツリーの作成に使用します。
setTextDecl()（1-21 ページ）	テキスト宣言の通知を受け取ります。パーサーは、テキスト宣言ごとにこのメソッドをコールします。
setXMLDecl()（1-21 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、このメソッドをコールします。
startCDATA()（1-22 ページ）	CDATA セクションの始まりを通知します。
startDocument()（1-22 ページ）	ドキュメントの始まりの通知を受け取ります。
startDTD()（1-22 ページ）	DTD 宣言（ある場合）の始まりを通知します。
startElement()（1-23 ページ）	要素の始まりの通知を受け取ります。
startEntity()（1-24 ページ）	XML の内部および外部エンティティの始まりを通知します。startEntity および endEntity イベントはすべて正しくネストされている必要があります。

DocumentBuilder()

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

構文

```
public  DocumentBuilder();
```

attributeDecl()

属性型の宣言を通知します。発生する org.xml.sax.SAXException は SAX の例外であり、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	対応付けされた要素の名前
aName	属性の名前
type	属性型を表す文字列
valueDefault	属性のデフォルト（「#IMPLIED」、「#REQUIRED」または「#FIXED」）を表す文字列。これらが該当しない場合は NULL になります。
value	属性のデフォルト値を表す文字列。存在しない場合は NULL になります。

cDATASection()

要素内の CDATA セクション・データの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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 文字列
start	配列の開始位置
length	配列のうちの使用する文字数

characters()

要素内の文字データの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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	文字データが入っている配列
start	配列の開始位置
length	配列のうちの使用する文字数

comment()

コメントの通知を受け取ります。ドキュメント内に XML コメントがあれば通知します。発生する org.xml.sax.SAXException は SAX の例外であり、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	コメント内に文字の入った配列
start	配列の開始位置
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()

要素型の宣言を通知します。発生する org.xml.sax.SAXException は SAX の例外であり、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	要素型の名前
model	内容モデルの正規化された文字列

endCDATA()

CDATA セクションの終わりを通知します。発生する org.xml.sax.SAXException は SAX の例外であり、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()

DTD の終わりの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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()

ドキュメントの終わりの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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()

DTD 宣言の終わりを通知します。発生する org.xml.sax.SAXException は SAX の例外であり、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()

要素の終わりの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、別の例外をラップし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、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	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か、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（修飾名が使用不可の場合は空の文字列）

endEntity()

エンティティの終わりを通知します。発生する org.xml.sax.SAXException は SAX の例外であり、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	終了するエンティティの名前

externalEntityDecl()

解析した外部エンティティ宣言を通知します。発生する org.xml.sax.SAXException は SAX の例外であり、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	エンティティの名前。パラメータ・エンティティの場合は、「%」で始まります。
publicId	宣言されたエンティティ宣言の公開識別子（宣言がない場合は NULL）。
systemId	宣言されたエンティティのシステム識別子。

getCurrentNode()

現在作成中の XMLNode を戻します。

構文

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

getDocument()

作成中のドキュメントを戻します。

構文

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

ignorableWhitespace()

要素内容内の無視できる空白の通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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	文字配列の開始位置
length	文字配列のうちの使用する文字数

internalEntityDecl()

内部エンティティ宣言を通知します。発生する org.xml.sax.SAXException は SAX の例外であり、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	エンティティの名前。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()

処理命令の通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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	処理命令データ（指定されていない場合は NULL）

retainCDATASection()

CDATA セクションを保持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 セクションを保持するかどうかを決定します。格納する場合は TRUE、格納しない場合は FALSE になります。

setDebugMode()

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(b oolean 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決定します。格納する場合は TRUE、格納しない場合は FALSE になります。

setDoctype()

DTD の通知をすぐ後に受け取ることができるように、DTD を設定します。発生する org.xml.sax.SAXException は SAX の例外であり、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	ドキュメントの DTD

setDocumentLocator()

ドキュメント・イベント用の Locator オブジェクトの通知をすぐ後に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、別のドキュメント・イベントで使用するためにロケータを格納する場合、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・イベントのロケータ

setNodeFactory()

オプションの NodeFactory を設定します。これは、カスタム DOM ツリーの作成に使用します。

構文

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

パラメータ	説明
f	NodeFactory/

setTextDecl()

テキスト宣言の通知を受け取ります。パーサーは、テキスト宣言ごとにこのメソッドをコールします。発生する org.xml.sax.SAXException は SAX の例外であり、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（指定されていない場合は NULL）

setXMLDecl()

XML 宣言の通知を受け取ります。パーサーは、XML 宣言ごとに、このメソッドをコールします。発生する org.xml.sax.SAXException は SAX の例外であり、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（指定されていない場合は NULL）

startCDATA()

CDATA セクションの始まりを通知します。発生する org.xml.sax.SAXException は SAX の例外であり、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()

ドキュメントの始まりの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、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()

DTD 宣言（ある場合）の始まりを通知します。発生する org.xml.sax.SAXException は SAX の例外であり、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	宣言された外部 DTD サブセットの公開識別子（宣言がない場合は NULL）
systemId	宣言された外部 DTD サブセットのシステム識別子（宣言がない場合は NULL）

startElement()

要素の始まりの通知を受け取ります。発生する org.xml.sax.SAXException は SAX の例外であり、別の例外が隠されている可能性があります。使用方法を次の表の中から選んでください。

構文	説明

`public void startElement(`
`  NSName elem,`
`  SAXAttrList attrlist);`
	要素を使用して、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	NSName オブジェクト
attlist	要素の SAXAttrList
namespaceURI	名前空間 URI（要素に名前空間 URI が設定されていないか、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（修飾名が使用不可の場合は空の文字列）
atts	要素に結び付けられた属性。属性がない場合は、空の属性オブジェクトになります。

startEntity()

XML の内部および外部エンティティの始まりを通知します。startEntity および endEntity イベントはすべて正しくネストされている必要があります。発生する `org.xml.sax.SAXException` は SAX の例外であり、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]」になります。

DOMParser クラス

W3C 勧告に従って XML 1.0 パーサーを実装し、XML 文書の解析および DOM ツリーの構築を行います。

構文

```
public class DOMParser
   oracle.xml.parser.v2.DOMParser
```

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 を設定します。
ERROR_ENCODING	public static final java.lang.String ERROR_ENCODING	エラー・ストリームによるエラー・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	エラーを通知するためのエラー・ストリーム。オブジェクトは OutputStream または PrintWriter です。この属性は、ErrorHandler が設定されている場合は無視されます。
NODE_FACTORY	public static final java.lang.String NODE_FACTORY	カスタム・ノードを構築するには NodeFactory を設定します。
SHOW_WARNINGS	public static final java.lang.String SHOW_WARNINGS	警告を無視するブーリアン Boolean.TRUE または Boolean.FALSE。

表 1-4 DOMParser のメソッドの概要

メソッド	説明
DOMParser()（1-26 ページ）	新しいパーサー・オブジェクトを作成します。
getAttribute()（1-26 ページ）	基になる実装の特定の属性を戻します。
getDoctype()（1-27 ページ）	DTD を取得します。

表 1-4 DOMParser のメソッドの概要（続き）

メソッド	説明
getDocument()（1-27 ページ）	ドキュメントを取得します。
parseDTD()（1-27 ページ）	与えられた入力ソースから外部 DTD を解析します。
reset()（1-28 ページ）	パーサーの状態をリセットします
retainCDATASection()（1-28 ページ）	CDATA セクションを保持するかどうかを決定します
setAttribute()（1-29 ページ）	基になる実装の特定の属性を設定します。
setDebugMode()（1-29 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()（1-30 ページ）	エラーおよび警告のための出力ストリームを作成します。
setNodeFactory()（1-30 ページ）	ノード・ファクトリを設定します。
showWarnings()（1-31 ページ）	警告を出力するかどうかを決定します

DOMParser()

新しいパーサー・オブジェクトを作成します。

構文

```
public  DOMParser();
```

getAttribute()

基になる実装の特定の属性を戻します。基になる実装が属性を認識しない場合に、`IllegalArgumentException` が発生します。

構文

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

パラメータ	説明
name	属性の名前

getDoctype()

DTD を戻します。

構文

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

getDocument()

解析中のドキュメントを戻します。

構文

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

parseDTD()

与えられた入力ストリームから外部 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起きた場合に発生します。
- `SAXException` - SAX の例外であり、別の例外が隠されている可能性があります。
- `IOException` - I/O エラーの場合に発生します。

次の表に、オプションを示します。

構文	説明
`public final void parseDTD( org.xml.sax.InputSource in, String rootName);`	入力ソースから外部 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外部 DTD を解析します。外部エンティティおよび DTD を変換するために、ベース URL を設定する必要があります。
`public final void parseDTD( java.io.Reader r, String rootName);`	Reader から外部 DTD を解析します。外部エンティティおよび DTD を変換するために、ベース URL を設定する必要があります。

構文	説明
`public final void parseDTD( String in, String rootName);`	文字列から外部 DTD を解析します。
`public final void parseDTD( java.net.URL url, String rootName);`	与えられた URL が指す外部 DTD ドキュメントを解析し、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
`rootName`	ルート要素として使用される要素
`r`	解析する XML データを含む Reader
`url`	解析する XML 文書を指す URL

reset()

パーサーの状態をリセットします

構文

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

retainCDATASection()

CDATA セクションを保持するかどうかを決定します

構文

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

パラメータ	説明
`flag`	TRUE は CDATASection を保持し（デフォルト）、FALSE は CDATASection をテキスト・ノードに変換します。

setAttribute()

基になる実装の特定の属性を設定します。基になる実装が属性を認識しない場合に、IllegalArgumentException が発生します。

構文

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

パラメータ	説明
name	属性の名前
value	属性の値

setDebugMode()

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断します。

setErrorStream()

エラーおよび警告を出力するための出力ストリームを作成します。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、標準のエラー出力ストリームである 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作成します。
`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作成します。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、IOEexception が発生します。
`public final void setErrorStream( java.io.PrintWriter out);`	エラーおよび警告の出力用のプリント・ライターを作成します。エラー・ストリームの設定中に I/O エラーが起きた場合に、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に使用します。
enc	使用するエンコーディング。

setNodeFactory()

ノード・ファクトリを設定します。アプリケーションは、ノード・ファクトリを拡張し、このメソッドを使用してノード・ファクトリを登録できます。パーサーは、ユーザーが提供したノード・ファクトリを使用して、DOM ツリーのノードを作成します。無効なファクトリが設定されると、XMLParseException が発生します。

構文

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

パラメータ	説明
factory	設定するノード・ファクトリ

showWarnings()

警告を出力するかどうかを決定します。

構文

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

パラメータ	説明
flag	警告を表示するかどうかを決定します。

NodeFactory クラス

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・クラスを作成できます。アプリケーションは、DOMParser 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。このメソッドに null を指定した場合、ノードは DOM ツリーに追加されません。

構文

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

表 1-5 NodeFactory のメソッドの概要

メソッド	説明
NodeFactory()（1-33 ページ）	クラス・コンストラクタです。
createAttribute()（1-33 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()（1-34 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()（1-34 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()（1-34 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()（1-35 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()（1-35 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。
createElementNS()（1-35 ページ）	指定したローカル名、接頭辞、名前空間 URI を含む要素ノードを作成して戻します。
createEntityReference()（1-36 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()（1-36 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()（1-36 ページ）	指定したテキストを含むテキスト・ノードを作成して戻します。

NodeFactory()

クラス・コンストラクタです。

構文

```
public  NodeFactory();
```

createAttribute()

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`	ノードの名前
`text`	ノードに対応するテキスト
`localName`	ノードの名前
`prefix`	ノードの接頭辞
`namespaceURI`	ノードの名前空間
`value`	ノードに対応する値

createCDATASection()

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()

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()

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()

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()

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	要素の名前

createElementNS()

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	要素の名前
prefix	要素の接頭辞
namespaceURI	要素の名前空間

createEntityReference()

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	ノードの名前

createProcessingInstruction()

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	ノードの名前
text	ノードに対応するテキスト

createTextNode()

指定したテキストを含むテキスト・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 クラスは、XML ファイルを確認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 のコマンドライン・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調べます。
-dtd	入力ファイルを DTD 検証で確認します。
-schema	入力ファイルをスキーマ検証で確認します。
-log <logfile>	エラー / 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	入力した圧縮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。

表 1-7 oraxml のメソッドの概要

メソッド	説明
oraxml()（1-38 ページ）	クラス・コンストラクタです。
main()（1-38 ページ）	操作ループです。

oraxml()

構文

```
public oraxml();
```

main()

構文

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

SAXAttrList クラス

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。また属性（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 つの方法で属性のリストにアクセスできます。

- 属性索引を使用
- 名前空間修飾名を使用
- 修飾名（接頭辞付き）を使用

このインタフェースは、現在は廃止された名前空間をサポートしない SAX1 インタフェースに代わるものです。名前空間をサポートするだけでなく、getIndex メソッドが追加されています。

属性のリスト内の順序は指定されておらず、実装ごとに異なります。

構文

```
public class SAXAttrList
   oracle.xml.parser.v2.SAXAttrList
```

表 1-8 SAXAttrList のメソッドの概要

メソッド	説明
SAXAttrList()（1-40 ページ）	クラス・コンストラクタです。
addAttr()（1-40 ページ）	親要素ノードに属性を追加します。
getExpandedName()（1-41 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()（1-41 ページ）	属性の索引を戻します。
getLength()（1-42 ページ）	リスト内の属性数を戻します。
getLocalName()（1-42 ページ）	属性のローカル名を索引で戻します。
getPrefix()（1-42 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()（1-43 ページ）	属性の修飾名を索引で戻します。
getType()（1-43 ページ）	属性の型を戻します。

表 1-8 SAXAttrList のメソッドの概要（続き）

メソッド	説明
getURI()（1-44 ページ）	属性の名前空間 URI を索引で戻します。
getValue()（1-44 ページ）	文字列としての属性値を戻します。
reset()（1-45 ページ）	SAXAttrList をリセットします。

SAXAttrList()

クラス・コンストラクタです。

構文

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

addAttr()

親要素ノードに属性を追加します。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、specified フラグおよび属性型を使用して属性を追加します。
`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、specified フラグ、属性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 qname
value	属性値
spec	specified フラグ
type	属性型
nmsp	名前空間

getExpandedName()

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	リスト内の属性の索引番号

getIndex()

属性の索引を戻します。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);`	属性の索引を修飾名で戻します。
`public int getIndex( String uri, String localName);`	属性の索引を名前空間 URI とローカル名で戻します。

パラメータ	説明
qName	修飾名（接頭辞付き）。
uri	名前空間 URI。名前に名前空間 URI が設定されていない場合は空の文字列。
localName	属性のローカル名。

getLength()

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、それらを任意の順序で提供できます。属性数が 0（ゼロ）の場合もあります。

構文

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

getLocalName()

属性のローカル名を索引で戻します。名前空間処理が行われない場合は空の文字列を、その索引が範囲外の場合は NULL を戻します。

構文

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

パラメータ	説明
index	属性索引（ゼロ基準）

getPrefix()

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	属性索引（ゼロ基準）

getQName()

属性の XML 1.0 修飾名を索引で戻します。何もない場合は空の文字列を、その索引が範囲外の場合は NULL を戻します。

構文

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

パラメータ	説明
index	属性索引（ゼロ基準）

getType()

属性の型を戻します。属性型は、「CDATA」、「ID」、「IDREF」、「IDREFS」、「NMTOKEN」、「NMTOKENS」、「ENTITY」、「ENTITIES」または「NOTATION」のうちのいずれかの文字列です（常に大文字）。パーサーが属性の宣言を読み込んでいない場合、またはパーサーが属性型を通知しない場合は、XML 1.0 勧告（3.3.3 節「属性値の正規化」）で説明するように必ず値「CDATA」を戻します。表記法でない列挙型属性の場合、パーサーはタイプを「NMTOKEN」として通知します。次の表に、オプションを示します。

構文	説明
`public String getType( int index);`	属性のタイプを索引で戻します。その索引が範囲外の場合は NULL を戻します。
`public String getType( String qName);`	属性のタイプを修飾名で戻します。その属性がリスト内にない場合や修飾名がない場合は NULL を戻します。
`public String getType( String uri, String localName);`	属性のタイプを名前空間名で戻します。その属性がリスト内にない場合や名前空間処理が行われない場合は NULL を戻します。

パラメータ	説明
index	属性索引（ゼロ基準）
qName	XML 1.0 修飾名
uri	名前空間 URI。名前に名前空間 URI が設定されていない場合は空の文字列
localName	属性のローカル名

getURI()

属性の名前空間 URI を索引で戻します。何もない場合は空の文字列を、その索引が範囲外の場合は NULL を戻します。

構文

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

パラメータ	説明
index	属性索引（ゼロ基準）

getValue()

文字列としての属性値を戻します（索引が範囲外の場合は NULL）。属性値がトークン（IDREFS、ENTITIES または NMTOKENS）のリストである場合は、これらのトークンは単一の文字列に結合され、各トークンはシングル・スペースで区切られます。次の表に、オプションを示します。

構文	説明
`public String getValue( int index);`	属性の値を索引で戻します。
`public String getValue( String qName);`	属性の値を修飾名で戻します。
`public String getValue( String uri, String localName);`	属性の値を名前空間名で戻します。

パラメータ	説明
index	属性索引（ゼロ基準）
qName	XML 1.0 修飾名
uri	名前空間 URI。名前に名前空間 URI が設定されていない場合は空の文字列
localName	属性のローカル名

reset()

SAXAttrList をリセットします。

構文

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

SAXParser クラス

このクラスは、W3C 勧告に従って、XML 1.0 SAX パーサーを実装します。アプリケーションでは、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 ドライバに実装する必要のあるインタフェースです。アプリケーションはこのインタフェースを使用して、パーサー内の機能とプロパティの設定および問合せ、文書を処理するためのイベント・ハンドラの登録および文書解析を行うことができます。

すべての SAX インタフェースは同期であると見なされます。解析メソッドは、解析が完了するまで戻ることができず、Reader はイベント・ハンドラ・コールバックが戻るのを待ってから次のイベントを通知する必要があります。

このインタフェースは、（現在非推奨の）SAX 1.0 パーサー・インタフェースに代わるものです。XMLReader インタフェースには、従来のパーサー・インタフェースに比べて重要な 2 つの機能強化があります。

- 機能およびプロパティを問い合せ、設定する標準手段が加わりました。
- XML 上位標準の多くで必要となる名前空間がサポートされました。

構文

```
public class SAXParser
  oracle.xml.parser.v2.SAXParser
```

表 1-9 SAXParser のメソッドの概要

メソッド	説明
SAXParser()（1-47 ページ）	新しいパーサー・オブジェクトを作成します。
getContentHandler()（1-47 ページ）	現行のコンテンツ・ハンドラを戻します。
getDTDHandler()（1-47 ページ）	現行の DTD ハンドラを戻します。
getFeature()（1-47 ページ）	機能の現在の状態（true または false）を戻します。
getProperty()（1-48 ページ）	プロパティの値を戻します。
setContentHandler()（1-49 ページ）	内容イベント・ハンドラを登録します。
setDTDHandler()（1-49 ページ）	DTD イベント・ハンドラを登録します。
setFeature()（1-50 ページ）	機能の状態を設定します。
setProperty()（1-50 ページ）	プロパティの値を設定します。

SAXParser()

新しいパーサー・オブジェクトを作成します。

構文

```
public  SAXParser()
```

getContentHandler()

現行のコンテンツ・ハンドラを戻します（何も登録されていない場合は NULL）。

構文

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

getDTDHandler()

現行の DTD ハンドラを戻します（何も登録されていない場合は NULL）。

構文

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

getFeature()

機能の現在の状態（true または false）を戻します。あらゆる完全修飾 URI が機能名になります。XMLReader は機能名を認識してもその値を戻せないという可能性があり、これが特に当てはまるのが SAX1 パーサーのアダプタの場合です。この場合、基になるパーサーが検証中なのか外部エンティティを拡張中なのかを知ることはできません。一部の機能値が、解析前、解析中または解析後といった特定の状況でのみ入手可能です。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に発生します。
- `org.xml.sax.SAXNotSupportedException` - XMLReader が機能名を認識するが、この時点ではその値を判断できない場合に発生します。

入力値が 2 進数であるため、機能は DTD 検証の操作のみ行います。値 true は DTD 検証を TRUE に設定します。この機能で、XML スキーマ型検証を操作することはできません。

実装者は、それぞれ自分の URI 上で作成した名前を使用して、独自の機能を自由に開発することをお薦めします。

構文

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

パラメータ	説明
name	機能名

getProperty()

プロパティの値を戻します。あらゆる完全修飾 URI がプロパティ名になります。XMLReader はプロパティ名を認識してもその状態を戻せないという可能性があり、これが特に当てはまるのが SAX1 パーサーのアダプタの場合です。XMLReader があらゆるプロパティ名を認識することは必須ではありませんが、初期コア・セットが SAX2 に文書化されます。一部のプロパティ値が、解析前、解析中、解析後といった特定の状況でのみ入手可能です。実装者は、それぞれ自分の URI 上で作成した名前を使用して、独自のプロパティを自由に開発することをお薦めします。次の例外が発生します。

- `org.xml.sax.SAXNotRecognizedException` - XMLReader が機能名を認識していない場合に発生します。
- `org.xml.sax.SAXNotSupportedException` - XMLReader が機能名を認識するが、この時点ではその値を判断できない場合に発生します。

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 です。

setContentHandler()

内容イベント・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 パーサーが通知するすべての内容イベントは自動的に無視されます。アプリケーションは解析の途中で新規または別のハンドラを登録する場合があり、SAX パーサーはただちに新規ハンドラを使用する必要があります。ハンドラ引数が NULL の場合に、`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()

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 パーサーが通知するすべての DTD イベントは自動的に無視されます。アプリケーションは解析の途中で新規または別のハンドラを登録する場合があり、SAX パーサーはただちに新規ハンドラを使用する必要があります。ハンドラ引数が NULL の場合に、`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()

機能の状態を設定します。あらゆる完全修飾 URI が機能名になります。XMLReader は機能名を認識してもその値を設定できないという可能性があり、これが特に当てはまるのが SAX1 パーサーのアダプタの場合です。この場合、たとえば基になるパーサーが検証中であるかどうかに影響を及ぼすことはありません。一部の機能値は、解析前、解析中または解析後といった特定の状況でのみ、不変または可変になります。入力値が 2 進数であるため、機能「http://www.xml.org/sax/features/validation」は DTD 検証の操作のみ行います。値 true は DTD 検証を TRUE に設定します。この機能で、XML スキーマ型検証を操作することはできません。次の例外が発生します。

- `org.xml.sax.SAXNotRecognizedException` - XMLReader が機能名を認識していない場合に発生します
- `org.xml.sax.SAXNotSupportedException` - XMLReader が機能名を認識しているがリクエストされた値を設定できない場合に発生します。

構文

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

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

setProperty()

プロパティの値を設定します。あらゆる完全修飾 URI がプロパティ名になります。XMLReader はプロパティ名を認識するがその値を設定できないという可能性があり、これが特に当てはまるのが SAX1 パーサーのアダプタの場合です。XMLReader があらゆるプロパティ名の設定を認識することは必須ではありませんが、コア・セットが SAX2 に提供されています。一部のプロパティ値は、解析前、解析中または解析後といった特定の状況でのみ、不変または可変になります。このメソッドは、拡張ハンドラを設定するため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に発生します。
- `org.xml.sax.SAXNotSupportedException` - XMLReader がプロパティ名を認識するがリクエストされた値を設定できない場合に発生します。

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 です。
state	プロパティのリクエストされた値。

XMLParseException クラス

XML 文書の処理中に解析例外が発生したことを示します。

構文

```
public class XMLParseException
   oracle.xml.parser.v2.XMLParseException
```

表 1-10 XMLParseException のフィールド

フィールド	構文	説明
ERROR	public static final int ERROR	非致命的エラーのコード
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

表 1-11 XMLParseException のメソッドの概要

メソッド	説明
XMLParseException()（1-53 ページ）	XMLParse 例外クラスのコンストラクタです。
formatErrorMessage()（1-53 ページ）	指定した索引でエラー・メッセージの書式を設定します。
getColumnNumber()（1-54 ページ）	指定した索引で発生したエラーの列番号を戻します。
getException()（1-54 ページ）	指定した索引のエラーで発生した例外を戻します。
getLineNumber()（1-54 ページ）	指定した索引で発生したエラーの行番号を戻します。
getMessage()（1-55 ページ）	指定した索引のエラー・メッセージを戻します。
getMessageType()（1-55 ページ）	指定した索引のメッセージ・タイプを戻します。
getNumMessages()（1-55 ページ）	解析中に検出されたエラー / 警告の合計数を戻します。
getPublicId()（1-56 ページ）	指定した索引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()（1-56 ページ）	特定の索引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()

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()

指定した索引のエラー・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	索引

getColumnNumber()

指定した索引で発生したエラーの列番号を戻します。

構文

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

パラメータ	説明
i	索引

getException()

指定した索引のエラーで発生した例外（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	索引

getLineNumber()

指定した索引で発生したエラーの行番号を戻します。

構文

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

パラメータ	説明
i	索引

getMessage()

指定した索引のエラー・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	索引

getMessageType()

指定した索引のエラー・メッセージのタイプを戻します。

構文

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

パラメータ	説明
i	索引

getNumMessages()

解析中に検出されたエラー / 警告の合計数を戻します。

構文

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

getPublicId()

指定した索引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	索引

getSystemId()

指定した索引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	索引

XMLParser クラス

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。このクラスには、W3C 勧告に従って XML1.0 文書を解析するメソッドが含まれています。このクラスは、インスタンス化できません（アプリケーションでは、用途に応じて、DOM パーサーまたは SAX パーサーをインスタンス化します）。

構文

```
public abstract class XMLParser
   oracle.xml.parser.v2.XMLParser
```

表 1-12 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() を置き換えます。URL オブジェクトです。
DTD_OBJECT	public static final java.lang.String DTD_OBJECT	検証で使用される DTD オブジェクト。XMLParser.setDoctype() を置き換えます。
SCHEMA_OBJECT	public static final java.lang.String SCHEMA_OBJECT	検証で使用されるスキーマ・オブジェクト。XMLParser.setXMLSchema() を置き換えます。
STANDALONE	public static final java.lang.String STANDALONE	入力ファイルのスタンドアロン・プロパティを設定します。true の場合、DTD は抽出されません。
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 を使用する場合のみです。

XMLParser のメソッド

表 1-13 XMLParser のメソッドの概要

メソッド	説明
getAttribute()（1-59 ページ）	実装の属性の値を取得します。
getBaseURL()（1-59 ページ）	ベース URL を戻します。
getEntityResolver()（1-60 ページ）	エンティティ・リゾルバを戻します
getErrorHandler()（1-60 ページ）	エラー・ハンドラを戻します。
getReleaseVersion()（1-60 ページ）	バージョン番号を戻します。
getValidationModeValue()（1-60 ページ）	検証モード値を戻します。
getXMLProperty()（1-61 ページ）	プロパティの値を戻します。
isXMLPropertyReadOnly()（1-61 ページ）	与えられたプロパティが読取り専用の場合に TRUE を戻します。
isXMLPropertySupported()（1-61 ページ）	与えられたプロパティがサポートされている場合に TRUE を戻します。
parse()（1-62 ページ）	XML を解析します。
reset()（1-62 ページ）	パーサーの状態をリセットします
setAttribute()（1-63 ページ）	基になる実装の特定の属性を設定します。
setBaseURL()（1-63 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()（1-63 ページ）	DTD を設定します。
setEntityResolver()（1-64 ページ）	エンティティ・リゾルバを設定します
setErrorHandler()（1-64 ページ）	エラー・イベント・ハンドラを登録します。
setLocale()（1-65 ページ）	エラー・レポート用のロケールを設定します。
setPreserveWhitespace()（1-65 ページ）	空白保持モードを設定します。

表 1-13 XMLParser のメソッドの概要（続き）

メソッド	説明
setValidationMode()（1-65 ページ）	検証モードを設定します。
setXMLProperty()（1-66 ページ）	XMLProperty を設定して戻します。
setXMLSchema()（1-66 ページ）	インスタンス・ドキュメントを検証するための XML スキーマを設定します。

getAttribute()

基になる実装の特定の属性の値を取得します。基になる実装が属性を認識しない場合に、`IllegalArgumentException` が発生します。

構文

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

パラメータ	説明
value	属性の値

getBaseURL()

ベース URL を戻します。

構文

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

getEntityResolver()

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（何も登録されていない場合は NULL）

getErrorHandler()

現行のエラー・ハンドラを戻します（何も登録されていない場合は NULL）。

構文

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

getReleaseVersion()

Oracle XML Parser のバージョン番号を戻します。

構文

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

getValidationModeValue()

次の検証モード値を戻します。

- 0 - XML パーサーが NONVALIDATING の場合
- 1 - XML パーサーが PARTIAL_VALIDATION の場合
- 2 - XML パーサーが DTD_VALIDATION の場合
- 3 - XML パーサーが SCHEMA_VALIDATION の場合

構文

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

getXMLProperty()

プロパティの値を戻します。プロパティは、存在しておりサポートされている場合に戻されます。そうでない場合は NULL が戻されます。

構文

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

パラメータ	説明
name	プロパティの名前

isXMLPropertyReadOnly()

与えられたプロパティが読取り専用の場合は 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	プロパティの名前

isXMLPropertySupported()

与えられたプロパティがサポートされている場合に TRUE を戻します。

構文

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

パラメータ	説明
name	プロパティの名前

parse()

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起きた場合に発生します。
- SAXException - SAX の例外であり、別の例外が隠されている可能性があります。
- IOException - I/O エラーの場合に発生します。

次の表に、オプションを示します。

構文	説明
`public void parse( org.xml.sax.InputSource in);`	与えられた入力ソースから XML を解析します。
`public final void parse( java.io.InputStream in);`	与えら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);`	与えられた Reader から XML を解析します。
`public void parse( String in);`	指定した URL から XML を解析します。
`public final void parse( java.net.URL url);`	与えられた 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	解析される XML 文書を指す URL

reset()

パーサーの状態をリセットします

構文

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

setAttribute()

基になる実装の特定の属性を設定します。基になる実装が属性を認識しない場合に、IllegalArgumentException が発生します。

構文

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

パラメータ	説明
name	属性の名前
value	属性の値

setBaseURL()

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。parse() が XML 文書の解析に使用される場合は、このメソッドを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()

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()

エンティティ・リゾルバを登録します。リゾルバ引数が NULL の場合に、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 は自分のデフォルト解決を実行します。アプリケーションは解析の途中で新規または異なるリゾルバを登録する場合があり、SAX パーサーはただちに新規リゾルバを使用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()

エラー・イベント・ハンドラを登録します。ハンドラ引数が NULL の場合に、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 パーサーが通知するすべてのエラー・イベントは自動的に無視されます。ただし正常な処理は継続しないことがあります。予期せぬ不具合を回避するために、すべての SAX アプリケーションがエラー・ハンドラを実装することを強くお薦めします。アプリケーションは解析の途中で新規または別のハンドラを登録する場合があり、SAX パーサーはただちに新規ハンドラを使用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()

エラー・レポート用のロケールを設定します。エラー発生時に 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()

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()

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 の 4 つのうち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	パーサーを設定する検証モードのタイプを決定します。

setXMLProperty()

XMLProperty を設定して戻します。プロパティ・セットの値は正常に設定された場合に戻されます。プロパティが読取り専用で設定できないか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	プロパティの名前
value	プロパティの値

setXMLSchema()

インスタンス・ドキュメントを検証するための XML スキーマを設定します。

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、XMLParser のメソッドである setTokenHandler() 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、ドキュメントを検証したり、内部 / 外部エンティティの追加 / 読取りを行いません。XMLToken ハンドラが null の場合は、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、解析中も（コールバック・メソッド内から）登録可能です。

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 構文変数に準拠しています。

構文

```
public interface XMLToken
```

表 1-14 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)* "'"
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 - '-')))* '-->'

表 1-14 XMLToken のフィールド（続き）

フィールド	構文	説明
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
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]+ ';'

表 1-14 XMLToken のフィールド（続き）

フィールド	構文	説明
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? '?' '>'

token()

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・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した XML トークン定数
value	解析したテキストからの対応する部分文字列

XMLTokenizer クラス

W3C 勧告に従って、XML 1.0 パーサーを実装します。

構文

```
public class XMLTokenizer
   oracle.xml.parser.v2.XMLTokenizer
```

表 1-15 XMLTokenizer のメソッドの概要

メソッド	説明
XMLTokenizer()（1-70 ページ）	新しい Tokenizer オブジェクトを作成します。
parseDocument()（1-71 ページ）	ドキュメントを解析します。
setErrorHandler()（1-71 ページ）	新規エラー・イベント・ハンドラを登録します。
setErrorStream()（1-71 ページ）	エラー用の出力ストリームを登録します。
setToken()（1-72 ページ）	XML トークン化機能のための新しいトークンを登録します。
setTokenHandler()（1-72 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()（1-72 ページ）	XML をトークン化します。

XMLTokenizer()

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);`	新しい Tokenizer オブジェクトをハンドラから作成します。

パラメータ	説明
handler	ハンドラ

parseDocument()

ドキュメントを解析します。

構文

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

setErrorHandler()

新しいエラー・イベント・ハンドラを登録します。以前のエラー処理設定は置き換えられます。

構文

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

パラメータ	説明
handler	登録中のハンドラ

setErrorStream()

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	登録中のエラー・ストリーム

setToken()

XML トークン化機能のため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	設定中の XMLToken

setTokenHandler()

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	登録中の XMLToken

tokenize()

XML をトークン化します。次の表に、オプションを示します。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起きた場合に発生します。
- SAXException - SAX の例外であり、別の例外が隠されている可能性があります。
- IOException - I/O エラーの場合に発生します。

構文	説明
`public final void tokenize( org.xml.sax.InputSource in);`	指定した入力ソースから XML をトークン化します。
`public final void tokenize( java.io.InputStream in);`	与えられた入力ストリームから XML をトークン化します。

構文	説明
`public final void tokenize( java.io.Reader r);`	与えられた Reader 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);`	与えられた 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`	解析のためのソース
`url`	解析する XML 文書を指す URL

NSName クラス

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

構文

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

表 1-16 NSName のメソッドの概要

メソッド	説明
getExpandedName()（1-74 ページ）	この名前の解決済完全名を戻します。
getLocalName()（1-74 ページ）	この名前のローカル名を戻します。
getNamespace()（1-75 ページ）	この名前の解決済名前空間を戻します。
getPrefix()（1-75 ページ）	この名前の接頭辞を戻します。
getQualifiedName()（1-75 ページ）	修飾名を戻します。

getExpandedName()

この名前の解決済完全名を戻します。

構文

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

getLocalName()

この名前のローカル名を戻します。

構文

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

getNamespace()

この名前の解決済名前空間を戻します。

構文

```
public String getNamespace()
```

getPrefix()

この名前の接頭辞を戻します。

構文

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

getQualifiedName()

修飾名を戻します。

構文

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

XMLError クラス

oracle.xml.util パッケージのこのクラスは、エラー・メッセージとこのエラーが発生した行番号を保持します。

構文

```
public class XMLError
   oracle.xml.util.XMLError
```

表 1-17 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

表 1-18 XMLError のメソッドの概要

メソッド	説明
XMLError()（1-78 ページ）	XMLError のデフォルトのコンストラクタです。
error()（1-78 ページ）	ベクトルに新しいエラーを追加します。
error0()（1-79 ページ）	ベクトルに新しいエラーを追加します。追加パラメータは含まれません。
error1()（1-79 ページ）	ベクトルに新しいエラーを追加します。追加パラメータが 1 つ含まれます。
error2()（1-80 ページ）	ベクトルに新しいエラーを追加します。追加パラメータが 2 つ含まれます。
error3()（1-80 ページ）	ベクトルに新しいエラーを追加します。追加パラメータが 3 つ含まれます。

表 1-18 XMLError のメソッドの概要（続き）

メソッド	説明
flushErrorStream()（1-81 ページ）	すべてのエラーを出力ストリーム（デフォルト）またはエラー・ハンドラへフラッシュします。
formatErrorMesg()（1-81 ページ）	エラー・メッセージの書式を設定します。
getColumnNumber()（1-81 ページ）	指定した索引で発生したエラーの列番号を戻します。
getException()（1-82 ページ）	指定した索引のエラーで発生した例外（存在する場合）を戻します。
getFirstError()（1-82 ページ）	最初のエラーを戻します。
getLineNumber()（1-82 ページ）	指定した索引で発生したエラーの行番号を戻します。
getLocator()（1-82 ページ）	登録したロケータを戻します。
getMessage()（1-83 ページ）	指定した索引のエラー・メッセージを戻します。
getMessage0()（1-83 ページ）	パラメータを含まないエラー・メッセージを戻します。
getMessage1()（1-84 ページ）	パラメータを 1 つ含むエラー・メッセージを戻します。
getMessage2()（1-84 ページ）	パラメータを 2 つ含むエラー・メッセージを戻します。
getMessage3()（1-85 ページ）	パラメータを 3 つ含むエラー・メッセージを戻します。
getMessage4()（1-85 ページ）	パラメータを 4 つ含むエラー・メッセージを戻します。
getMessage5()（1-86 ページ）	パラメータを 5 つ含むエラー・メッセージを戻します。
getMessageType()（1-86 ページ）	指定した索引のエラー・メッセージのタイプを戻します。
getNumMessages()（1-87 ページ）	解析中に検出されたエラー / 警告の合計数を戻します。
getPublicId()（1-87 ページ）	指定した索引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()（1-87 ページ）	指定した索引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()（1-88 ページ）	JAXP 1.1 エラーをすべて ErrorListener へフラッシュします。ErrorListener が設定されていない場合は、デフォルトで System.err になります。
reset()（1-88 ページ）	エラー・クラスをリセットします。

表 1-18 XMLError のメソッドの概要（続き）

メソッド	説明
setErrorStream()（1-88 ページ）	出力ストリームを登録します。
setException()（1-89 ページ）	例外を登録します。
setLocale()（1-89 ページ）	ロケールを登録します。
setLocator()（1-89 ページ）	ロケータを登録します。
showWarnings()（1-90 ページ）	警告通知のオン / オフを切り替えます。

XMLError()

XMLError のデフォルトのコンストラクタです。

構文

```
public  XMLError();
```

error()

ベクトルに新しいエラーを追加します。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);`	ベクトルに新しいエラーを追加します。メッセージが 1 つ含まれます。
`public void error3( int id, int type, String[] params);`	ベクトルに新しいエラーを追加します。パラメータの配列が含まれます。

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

パラメータ	説明
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()

ベクトルに新しいエラーを追加します。パラメータは含まれません。

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()

ベクトルに新しいエラーを追加します。パラメータが 1 つ含まれます。

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()

ベクトルに新しいエラーを追加します。パラメータが 2 つ含まれます。

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()

ベクトルに新しいエラーを追加します。パラメータが 3 つ含まれます。

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()

すべてのエラーを出力ストリーム（デフォルト）またはエラー・ハンドラへフラッシュします。

構文

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

formatErrorMesg()

エラー・メッセージの書式を設定します。

構文

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

パラメータ	説明
i	索引

getColumnNumber()

指定した索引で発生したエラーの列番号を戻します。

構文

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

パラメータ	説明
i	索引

getException()

指定した索引のエラーで発生した例外（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	索引

getFirstError()

最初のエラーを戻します。

構文

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

getLineNumber()

指定した索引で発生したエラーの行番号を戻します。

構文

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

パラメータ	説明
i	索引

getLocator()

登録したロケータを戻します。

構文

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

getMessage()

エラー・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);`	指定した索引のエラー・メッセージを戻します。
`public String getMessage( int errId, String[] params);`	パラメータ数が 5 を超え、1 つの配列にまとめられたエラー・メッセージを戻します。

パラメータ	説明
`i`	索引
`errId`	エラー ID
`param`	パラメータの配列

getMessage0()

パラメータを含まないエラー・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()

引数を 1 つ含むエラー・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()

パラメータを 2 つ含むエラー・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()

パラメータを 3 つ含むエラー・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()

パラメータを 4 つ含むエラー・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()

パラメータを 5 つ含むエラー・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()

指定した索引のエラー・メッセージのタイプを戻します。

構文

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

パラメータ	説明
i	索引

getNumMessages()

解析中に検出されたエラー / 警告の合計数を戻します。

構文

```
public int getNumMessages()
```

getPublicId()

指定した索引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	索引

getSystemId()

指定した索引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	索引

printErrorListener()

JAXP 1.1 エラーをすべて ErrorListener へフラッシュします。ErrorListener が設定されていない場合は、デフォルトで System.err になります。

構文

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

reset()

エラー・クラスをリセットします。

構文

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

setErrorStream()

エラー・レポート用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・レポート用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でエラーが起きた場合に、`IOException` が発生します。
`public void setErrorStream( java.io.OutputStream out, String enc);`	エラー・レポートとそのエンコーディング用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でエラーが起きた場合に、`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 / 警告の出力
`enc`	出力ストリームのエンコーディング

setException()

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()

エラー・レポート用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・レポート用のロケール

setLocator()

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 ID/ 公開識別子情報を取得するロケータ

showWarnings()

警告通知のオン / 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

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

構文

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

表 1-19 XMLException のフィールド

フィールド	構文	説明
ERROR	public static final int ERROR	非致命的エラーのコード
FATAL_ERRORS	public static final int FATAL_ERRORS	致命的エラーのコード
WARNING	public static final int WARNINGS	警告のコード

表 1-20 oracle.xml.util.XMLException のメソッドの概要

メソッド	説明
XMLException()（1-92 ページ）	XMLException を生成します。
formatErrorMessage()（1-93 ページ）	指定した索引のエラー・メッセージを戻します。
getColumnNumber()（1-93 ページ）	指定した索引で発生したエラーの列番号を戻します。
getException()（1-94 ページ）	指定した索引のエラーで発生した例外（存在する場合）を戻します。
getLineNumber()（1-94 ページ）	指定した索引で発生したエラーの行番号を戻します。
getMessage()（1-94 ページ）	指定した索引のエラー・メッセージを戻します。
getMessageType()（1-95 ページ）	指定した索引のエラー・メッセージのタイプを戻します。
getNumMessages()（1-95 ページ）	解析中に検出されたエラー / 警告の合計数を戻します。
getPublicId()（1-95 ページ）	指定した索引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()（1-96 ページ）	指定した索引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()（1-96 ページ）	XMLException 内の XMLError オブジェクトを戻します。

表 1-20 oracle.xml.util.XMLException のメソッドの概要（続き）

メソッド	説明
printStackTrace()（1-96 ページ）	この Throwable およびそのバックトレースを出力します。
setException()（1-97 ページ）	潜在的な例外（存在する場合）を設定します。
toString()（1-97 ページ）	埋め込まれた例外を戻します。

XMLException()

XML 例外を生成します。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);`	XMLException をメッセージ、公開識別子、システム識別子、行、列およびタイプから生成します。
`public XMLException( XMLError err, java.lang.Exception e);`	XMLException をエラーおよび例外から生成します。
`public XMLException( XMLError err, int firsterr);`	XMLException をエラーおよび第 1 エラーから生成します。
`public XMLException( XMLError err, int firsterr, java.lang.Exception e);`	XMLException をエラー、例外および第 1 エラーから生成します。

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

パラメータ	説明
type	型
err	エラー
e	例外
firsterr	最初のエラー

formatErrorMessage()

指定した索引のエラー・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	索引

getColumnNumber()

指定した索引で発生したエラーの列番号を戻します。

構文

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

パラメータ	説明
i	索引

getException()

指定した索引のエラーで発生した例外（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	索引

getLineNumber()

指定した索引で発生したエラーの行番号を戻します。

構文

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

パラメータ	説明
i	索引

getMessage()

指定した索引のエラー・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	索引

getMessageType()

指定した索引のエラー・メッセージのタイプを取得します。

構文

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

パラメータ	説明
i	索引

getNumMessages()

解析中に検出されたエラー / 警告の合計数を戻します。

構文

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

getPublicId()

指定した索引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	索引

getSystemId()

指定した索引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	索引

getXMLError()

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

構文

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

printStackTrace()

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();`	この 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);`	この Throwable およびそのバックトレースを、指定した出力ストリームに出力します。
`public void printStackTrace( java.io.PrintWriter s);`	この Throwable およびそのバックトレースを、指定したプリント・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()

潜在的な例外（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()

埋め込まれた例外を戻します。

構文

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

2

Document Object Model（DOM）

XML 文書はツリー表示が可能であり、そのノードは開始タグと終了タグの間にある情報から構成されています。このツリー表現、または Document Object Model（DOM）は、XML パーサーが文書を解析する場合にメモリー内に形成されます。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、このツリーへアクセスおよびナビゲートするための API について説明します。内容は次のとおりです。

- 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

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

NSResolver インタフェース

名前空間の解決をサポートします。

構文

```
public interface NSResolver
```

resolveNamespacePrefix()

与えられた名前空間の接頭辞の有効範囲で名前空間定義を検索し、それを戻します。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 インタフェースは、DOM ツリーで表示される XML 文書の印刷に使用されるメソッドを定義します。

構文

```
public interface PrintDriver
```

表 2-1 PintDriver 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()（2-5 ページ）	XMLElement の各属性のプリント・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()（2-5 ページ）	XMLNode の子のそれぞれにプリント・メソッドをコールします。
printComment()（2-6 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()（2-7 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。
printProcessingInstruction()（2-8 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）	プリント・ドライバのエンコーディングを設定します。

close()

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()

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()

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()

XMLElement の各属性のプリント・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	出力される属性を持つ要素

printCDATASection()

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()

XMLNode の子のそれぞれにプリント・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	出力される子を持つノード

printComment()

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()

DTD を出力します。

構文

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

パラメータ	説明
dtd	出力される DTD

printDocument()

XMLDocument を出力します。

構文

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

パラメータ	説明
doc	出力されるドキュメント

printDocumentFragment()

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	出力されるドキュメント・フラグメント

printElement()

XMLElement を出力します。

構文

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

パラメータ	説明
elem	出力される要素

printEntityReference()

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	出力される XMLEntityReference ノード

printProcessingInstruction()

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	出力される XMLPI ノード

printTextNode()

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	テキスト・ノード

setEncoding()

プリント・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	出力されるドキュメントのエンコーディング

AttrDecl クラス

Document Type Definition（DTD）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。

構文

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

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	属性の存在型 - 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	属性の存在型 - 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	属性の存在型 - 暗黙
NMTOKEN	public static final int NMTOKEN	属性の型 - トークン型 - 名前トークン
NMTOKENS	public static final int NMTOKENS	属性の型 - トークン型 - 複数の名前トークン

表 2-2 AttrDecl のフィールド（続き）

フィールド	構文	説明
NOTATION	public static final int NOTATION	属性の型 - 列挙型 - 表記法
REQUIRED	public static final int REQUIRED	属性の存在型 - 要求

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）	属性の存在型を戻します。
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()（2-12 ページ）	属性の値を列挙として戻します。
getNodeName()（2-13 ページ）	属性宣言の名前を戻します。
getNodeType()（2-13 ページ）	基礎となる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 メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
typeToString()（2-14 ページ）	属性の形の文字列表現を戻します。
writeExternal()（2-14 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

AttrDecl()

デフォルトのコンストラクタです。

構文

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

getAttrPresence()

属性の存在型を戻します。

構文

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

getAttrType()

属性の型を戻します。

構文

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

getDefaultValue()

属性のデフォルト値を戻します。

構文

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

getEnumerationValues()

属性の値を列挙として戻します。

構文

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

getNodeName()

属性宣言の名前を戻します。

構文

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

getNodeType()

基礎となる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()

圧縮ストリームよりオブジェクトを読み取り、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み取り中にエラーが起きた場合に発生します。
- `ClassNotFoundException` - クラスが見つからない場合に発生します。

構文

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

パラメータ	説明
in	圧縮ストリームの読取りに使用される ObjectInput ストリーム

typeToString()

属性の形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	属性の形の数値表現

writeExternal()

バイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。シリアライズ / 圧縮ストリームの場合、IOException が発生します。

構文

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

パラメータ	説明
out	ストリームの書込みに使用される ObjectOutput ストリーム

DTD クラス

DOM の DocumentType インタフェースを実装し、文書型を保持します。XML 文書の定義情報です。

構文

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

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）	指定する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 の名前付きエンティティを検索して戻します。検索されない場合は null を戻します。
findNotation()（2-17 ページ）	DTD から名前表記法を取得します。検索されない場合は null を戻します。
getChildNodes()（2-18 ページ）	このノードのすべての子ノードを含むノード・リストを戻します。
getElementDecls()（2-18 ページ）	DTD の要素宣言を含む NamedNodeMap を戻します。
getEntities()（2-18 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 NamedNodeMap を戻します。
getInternalSubset()（2-18 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、または DOCTYPE キーワードの直後の名前を戻します。
getNodeName()（2-19 ページ）	DTD の名前、または DOCTYPE キーワードの直後の名前を戻します。
getNodeType()（2-19 ページ）	基礎となるオブジェクトの型を表すコードを戻します。
getNotations()（2-19 ページ）	DTD で宣言されている表記法を含む NamedNodeMap を戻します。
getOwnerImplementation()（2-20 ページ）	DTD 実装の所有者を戻します。

表 2-4 DTD のメソッドの概要（続き）

メソッド	説明
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()（2-20 ページ）	DTD のルート・タグを戻します。
getSystemId()（2-20 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）	ノードに子ノードがあるかどうかを判別します。XMLNode では、DTD はオーバーライド・メソッドを持つ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()（2-21 ページ）	このドキュメントの内容を特定の出力に書き込みます。
readExternal()（2-22 ページ）	writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
setRootTag()（2-22 ページ）	DTD のルート・タグを設定します。
writeExternal()（2-22 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

DTD()

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、この DOM ノードがデシリアライズ / 解凍中の場合にのみ使用できることに注意してください。シリアライズ / 圧縮ストリームから DOM ノードを構成するためにこのノードをデシリアライズするには、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()

指定する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()

DTD の名前付きエンティティを検索して戻します。検索されない場合は 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リアン

findNotation()

DTD から名前表記法を取得します。検索されない場合は 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	表記法の名前

getChildNodes()

このノードのすべての子ノードを含む NodeList を戻します。子ノードが存在しない場合は、ノードを含まない NodeList が戻されます。戻された NodeList の内容は、たとえば、それが作成される基になったノード・オブジェクトの子ノードに変更があった場合、その変更が NodeList アクセッサが戻すノードにすぐに反映されるという意味で最新のものであり、ノード内容の静的スナップショットではありません。このことは、`getElementsByTagName` メソッドが戻す NodeList を含む、すべての NodeList についていえます。

構文

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

getElementDecls()

DTD の要素宣言を含む `NamedNodeMap` を戻します。このマップのすべてのノードは、`ElementDecl` オブジェクトです。DTD の要素宣言を含む NamedNodeMap

構文

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

getEntities()

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 `NamedNodeMap` を戻します。複製は廃棄されます。また、このマップのすべてのノードは、`Entity` インタフェースを実装します。

構文

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

getInternalSubset()

DTD の内部サブセットを戻します。

構文

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

getName()

DTD の名前、または DOCTYPE キーワードの直後の名前を戻します。

構文

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

getNodeName()

DTD の名前、または DOCTYPE キーワードの直後の名前を戻します。

構文

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

getNodeType()

基礎となる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()

DTD で宣言されている表記法を含む NamedNodeMap を戻します。複製は廃棄されます。このマップ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()

DTD 実装の所有者を戻します。

構文

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

getPublicId()

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()

DTD のルート・タグを戻します。

構文

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

getSystemId()

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()

ノードに子ノードがあるかどうかを判別します。XMLNode では、DTD はオーバーライド・メソッドを持つことができないため、常に false を戻します。

構文

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

normalize()

DTD を正規化します。

構文

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

printExternalDTD()

このドキュメントの内容を特定の出力に書き込みます。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起きた場合に IOException が発生します。

構文	説明
`public void printExternalDTD( java.io.OutputStream out);`	このドキュメントの内容を特定の出力ストリームに書き込みます。
`public void printExternalDTD( java.io.OutputStream out, String enc);`	このドキュメントの内容を特定のエンコードされた出力ストリームに書き込みます。
`public void printExternalDTD( java.io.PrintWriter out);`	このドキュメントの内容を特定のプリント・ライターに書き込みます。

パラメータ	説明
out	出力
enc	出力に使用されるエンコーディング

readExternal()

writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従っ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み取り中にエラーが起きた場合に発生します。
- `ClassNotFoundException` - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

setRootTag()

DTD のルート・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()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。シリアライズ / 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`	ObjectOutput ストリーム

ElementDecl クラス

DTD の要素宣言を表します。

構文

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

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	要素内容型 - 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・フラグです。
ELEMENTS	public static final byte ELEMENTS	要素内容型 - 子は要素になることができます。内容モデルを参照してください。
EMPTY	public static final byte EMPTY	要素内容型 - 子を持ちません。
ID_ATTR_DECL	public static final int ID_ATTR_DECL	DTD の ID 属性宣言を表すノード・フラグです。
MIXED	public static final byte MIXED	要素内容型 - 子は PCDATA および要素になることができます。内容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）。

表 2-6 ElementDecl のメソッドの概要

メソッド	説明
ElementDecl()（2-25 ページ）	デフォルトのコンストラクタです。
cloneNode()（2-25 ページ）	この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()（2-25 ページ）	要素に追加できる要素名のベクトルを戻します。
findAttrDecl()（2-26 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()（2-26 ページ）	列挙された属性宣言を戻します。
getContentElements()（2-26 ページ）	この要素に追加できる要素のベクトルを戻します。
getContentType()（2-26 ページ）	要素の内容モデルを戻します。
getNodeName()（2-27 ページ）	要素宣言の名前を戻します。
getNodeType()（2-27 ページ）	基礎となるオブジェクトの型を表すコードを戻します。
getParseTree()（2-27 ページ）	内容モデルを解析したツリーのルート・ノードを戻します。
readExternal()（2-28 ページ）	writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
validateContent()（2-28 ページ）	要素ノードの内容を検証します。
writeExternal()（2-28 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

ElementDecl()

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、この DOM ノードがデシリアライズ / 解凍中の場合にのみ使用できることに注意してください。シリアライズ / 圧縮ストリームから DOM ノードを構成するためにこのノードをデシリアライズするには、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()

この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、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、子であるテキスト・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。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。

構文

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

パラメータ	説明
deep	true の場合、指定し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、ノードのみが（ノードが要素の場合はその属性も）複製されます。

expectedElements()

要素に追加できる要素名のベクトルを戻します。

構文

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

パラメータ	説明
e	要素

findAttrDecl()

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()

列挙された属性宣言を戻します。

構文

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

getContentElements()

この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()

要素の内容モデルを戻します。

構文

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

getNodeName()

要素宣言の名前を戻します。

構文

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

getNodeType()

基礎となる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()

内容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・タイプおよび名前を戻します。

構文

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

readExternal()

writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従っ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み取り中にエラーが起きた場合に発生します。
- ClassNotFoundException - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

validateContent()

要素ノードの内容を検証します。有効な場合は true、無効な場合は 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()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。シリアライズ / 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	シリアライズ / 圧縮ストリームの書込みに使用される ObjectOutput ストリーム

XMLAttr クラス

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。

構文

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

表 2-7 XMLAttr のメソッドの概要

メソッド	説明
addText()（2-30 ページ）	XMLNode にテキストを追加します。
cloneNode()（2-30 ページ）	この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()（2-31 ページ）	この属性の解決済完全名を戻します。
getLocalName()（2-31 ページ）	この属性のローカル名を戻します。
getName()（2-31 ページ）	属性名を戻します。
getNamespaceURI()（2-31 ページ）	属性の名前空間を戻します。
getNextAttribute()（2-31 ページ）	次の属性がある場合は、それを戻します。
getNextSibling()（2-32 ページ）	次の兄弟関係がある場合は、それを戻します。
getNodeType()（2-32 ページ）	基礎となるオブジェクトの型を表すコードを戻します。
getNodeValue()（2-32 ページ）	ノード・タイプに応じて、ノードの値を戻します。
getOwnerElement()（2-32 ページ）	この属性を所有する要素を戻します。
getParentNode()（2-33 ページ）	ノードの親を戻します。
getPrefix()（2-33 ページ）	要素の名前空間接頭辞を戻します。
getPreviousSibling()（2-33 ページ）	以前の兄弟関係がある場合は、それを戻します。
getSpecified()（2-33 ページ）	属性が要素で明示的に指定されている場合は true を、それ以外の場合は false を戻します。
getValue()（2-34 ページ）	属性値を戻します。
readExternal()（2-34 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()（2-34 ページ）	このノードの値を、ノード・タイプに応じて設定します。

表 2-7 XMLAttr のメソッドの概要（続き）

メソッド	説明
setValue()（2-35 ページ）	値を設定します。
writeExternal()（2-35 ページ）	バイナリ圧縮ストリームでオブジェクトの状態を保存します。

addText()

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	追加されたテキスト

cloneNode()

この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、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、子であるテキスト・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。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。

構文

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

パラメータ	説明
deep	true の場合、指定し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、ノードのみが（ノードが要素の場合はその属性も）複製されます。

getExpandedName()

この属性の解決済完全名を戻します。

構文

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

getLocalName()

この属性のローカル名を戻します。

構文

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

getName()

属性名を戻します。

構文

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

getNamespaceURI()

属性の名前空間を戻します。

構文

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

getNextAttribute()

次の属性がある場合は、それを戻します。

構文

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

getNextSibling()

次の兄弟関係がある場合は、それを戻します。

構文

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

getNodeType()

基礎となる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()

ノード・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: ノードが読取り専用である場合に発生します。
- DOMSTRING_SIZE_ERR: ノードが実装プラットフォーム上で DOMString 変数より多い文字を戻す場合に発生します。

構文

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

getOwnerElement()

この属性を所有する要素を戻します。

構文

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

getParentNode()

ノードの親を戻します。ドキュメント、ドキュメント・フラグメントおよび属性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、またはツリーから削除された場合は、null が戻されます。

構文

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

getPrefix()

要素の名前空間接頭辞を戻します。

構文

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

getPreviousSibling()

以前の兄弟関係がある場合は、それを戻します。

構文

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

getSpecified()

属性が要素で明示的に指定されている場合は true を、それ以外の場合は false を戻します。

構文

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

getValue()

属性値を戻します。

構文

```
public String getValue();
```

readExternal()

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み取り中に例外が発生した場合に発生します。
- ClassNotFoundException - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

setNodeValue()

このノードの値を、ノード・タイプに応じて設定します。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR: ノードが読取り専用である場合に発生します。
- DOMSTRING_SIZE_ERR: ノードが実装プラットフォーム上で DOMString 変数より多い文字を戻す場合に発生します。

構文

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

パラメータ	説明
nodeValue	設定されるノード値

setValue()

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()

バイナリ圧縮ストリームで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される ObjectOutput ストリーム

XMLCDATA クラス

DOM の 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。

構文

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

表 2-8 XMLCDATA のメソッドの概要

メソッド	説明
XMLCDATA()（2-36 ページ）	デフォルトのコンストラクタです。
getNodeName()（2-36 ページ）	ノードの名前を戻します。
getNodeType()（2-37 ページ）	基礎となるオブジェクトの型を表すコードを戻します。
readExternal()（2-37 ページ）	writeExternal() メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
writeExternal()（2-37 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

XMLCDATA()

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、この DOM ノードがデシリアライズ / 解凍中の場合にのみ使用できることに注意してください。シリアライズ / 圧縮ストリームから DOM ノードを構成するためにこのノードをデシリアライズするには、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()

ノードの名前を戻します。

構文

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

getNodeType()

基礎となる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()

writeExternal() メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従っ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み取り中にエラーが起きた場合に発生します。
- ClassNotFoundException - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

writeExternal()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、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される ObjectOutput ストリーム

XMLComment クラス

DOM の 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。

構文

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

表 2-9 XMLComment の概要

メソッド	説明
XMLComment()（2-38 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、この DOM ノードがデシリアライズ / 解凍中の場合にのみ使用できることに注意してください。
addText()（2-39 ページ）	コメント・テキストを追加します。
getNodeName()（2-39 ページ）	ノードの名前を戻します。
getNodeType()（2-39 ページ）	基礎となる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）	圧縮ストリームの情報を読み取り、それに従ってオブジェクトをリストアします。
reportSAXEvents()（2-40 ページ）	DOM ツリーから SAX イベントを通知します。
writeExternal()（2-40 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

XMLComment()

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、この DOM ノードがデシリアライズ / 解凍中の場合にのみ使用できることに注意してください。シリアライズ / 圧縮ストリームから DOM ノードを構成するためにこのノードをデシリアライズするには、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()

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()

ノードの名前を戻します。

構文

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

getNodeType()

基礎となる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()

writeExternal() メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従っ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み取り中にエラーが起きた場合に発生します。
- ClassNotFoundException - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

reportSAXEvents()

DOM ツリーから SAX イベントを通知します。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()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、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される ObjectOutput ストリーム

XMLDeclPI クラス

XML 宣言の Processing Instruction を実装します。「XMLPI クラス」も参照してください。

構文

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

表 2-10 XMLDeclPI のメソッドの概要

メソッド	説明
XMLDeclPI()（2-42 ページ）	XMLDeclPI の新しいインスタンスを作成します。
cloneNode()（2-42 ページ）	この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()（2-43 ページ）	完全に構成された文字列 'version=1.0....' を戻します。
getEncoding()（2-43 ページ）	<?xml...?> タグに格納される文字コード情報やコード情報、またはユーザー定義の出力コード（コード情報より後で設定された場合）を戻します。
getNodeValue()（2-43 ページ）	ノードの値を戻します。
getStandalone()（2-43 ページ）	<?xml...?> タグに格納されるスタンドアロン情報、またはスタンドアロン属性を戻します。
getVersion()（2-44 ページ）	<?xml...?> タグに格納されるバージョン情報、またはバージョン番号を取得します。
readExternal()（2-44 ページ）	writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
setEncoding()（2-44 ページ）	出力用の文字コードを設定します。
setStandalone()（2-45 ページ）	<?xml...?> タグに格納されるスタンドアロン情報を設定します。
setVersion()（2-45 ページ）	<?xml...?> タグに格納されるバージョン番号を設定します。
writeExternal()（2-45 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

XMLDeclPI()

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します。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、スタンドアロン、textDecl 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 の作成で使用されたバージョンです。
`encoding`	新しい XMLDeclPI の作成で使用されたエンコーディングです。
`standaolone`	新しい XMLDeclPI がスタンドアロンの場合に指定します。
`textDecl`	新しい XMLDeclPI のテキスト宣言です。

cloneNode()

この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、指定し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、ノードのみが（ノードが要素の場合はその属性も）複製されます。

getData()

完全に構成された文字列 'version=1.0....' を戻します。次の DOMException が発生します。

- DOMSTRING_SIZE_ERR: このノードが、実装プラットフォーム上で DOMString 変数より多い文字を戻す場合に発生します。

構文

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

getEncoding()

<?xml...?> タグに格納される文字コード情報やコード情報、またはユーザー定義の出力コード（コード情報より後で設定された場合）を戻します。

構文

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

getNodeValue()

ノードの値を戻します。次の DOMException が発生します。

- DOMSTRING_SIZE_ERR: ノードが実装上で DOMString 変数より多い文字を戻す場合に発生します。

構文

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

getStandalone()

<?xml...?> タグに格納されるスタンドアロン情報、またはスタンドアロン属性を戻します。

構文

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

getVersion()

<?xml...?> タグに格納されるバージョン情報、またはバージョン番号を戻します。

構文

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

readExternal()

writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従っ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み取り中にエラーが起きた場合に発生します。
- `ClassNotFoundException` - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

setEncoding()

出力用の文字コードを設定します。また、最終的には <?xml...?> タグに格納される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、この設定を行いません。したがって、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`	設定する文字コード

setStandalone()

<?xml...?> タグに格納される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	新しい XMLDeclPI がスタンドアロンの場合に指定します。スタンドアロンの場合は true、それ以外の場合は false となります。

setVersion()

<?xml...?> タグに格納される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()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、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される ObjectOutput ストリーム

XMLDocument クラス

DOM の Document インタフェースを実装します。XML 文書全体を表し、DOM 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 の仕様に従い、ツリーのルートは、複数のコメントおよび処理命令の組合せで構成されますが、ルート要素は 1 つのみです。ルート要素を検出するショート・カットとして、補助メソッド `getDocumentElement` が提供されます。

構文

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

表 2-11 XMLDocument のメソッドの概要

メソッド	説明
XMLDocument()（2-49 ページ）	空のドキュメントを作成します。
addID()（2-49 ページ）	このドキュメントに対応する ID 要素を追加します。
adoptNode()（2-50 ページ）	他のドキュメントからこのドキュメントにノードを適用します。
appendChild()（2-50 ページ）	新しいノードをドキュメントに追加します。
cloneNode()（2-51 ページ）	この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()（2-51 ページ）	与えられた名前の属性を作成します。
createAttributeNS()（2-52 ページ）	与えら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()（2-52 ページ）	指定した文字列を値に持つ CDATA セクション・ノードを作成します。
createComment()（2-53 ページ）	指定した文字列を値に持つコメント・ノードを作成します。
createDocumentFragment()（2-53 ページ）	空のドキュメント・フラグメント・オブジェクトを作成します。
createElement()（2-53 ページ）	指定した型の要素を作成します。
createElementNS()（2-54 ページ）	与えられた修飾名および名前空間 URI を持つ要素を作成します。
createEntityReference()（2-54 ページ）	実体参照オブジェクトを作成します。

表 2-11 XMLDocument のメソッドの概要（続き）

メソッド	説明
createEvent()（2-55 ページ）	指定した型のイベント・オブジェクトを作成します。
createMutationEvent()（2-55 ページ）	指定した型の変異イベント・オブジェクトを作成します。
createNodeIterator()（2-55 ページ）	指定されたルート、論理ビューに含めるノードのタイプを決定するフラグ、ノード用のフィルタ、エンティティへの参照やその子孫を含むことができるかどうかを判別するフラグを持つノード・イテレータを作成します。
createProcessingInstruction()（2-56 ページ）	指定した名前およびデータ文字列を持つ処理命令ノードを作成します。DOMException が発生します。
createRange()（2-56 ページ）	ドキュメントの始まりに最初と最後の境界ポイントを持つドキュメント範囲オブジェクトを作成します。
createRangeEvent()（2-57 ページ）	指定した型の範囲イベント・オブジェクトを作成します。
createTextNode()（2-57 ページ）	指定した文字列を値に持つテキスト・ノードを作成します。
createTraversalEvent()（2-57 ページ）	指定した型の走査イベント・オブジェクトを作成します。
createTreeWalker()（2-58 ページ）	指定されたルート、論理ビューに含めるノードのタイプを決定するフラグ、ノード用のフィルタ、エンティティへの参照やその子孫を含むことができるかどうかを判別するフラグを持つノード・イテレータを作成します。
expectedElements()（2-58 ページ）	要素に追加できる要素名のベクトルを戻します。
getColumnNumber()（2-59 ページ）	列番号のデバッグ情報を戻します。
getDebugMode()（2-59 ページ）	デバッグ・フラグを戻します。
getDoctype()（2-59 ページ）	このドキュメントに対応する DTD を戻します。
getDocumentElement()（2-59 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()（2-60 ページ）	elementId によって ID を与えられた要素を戻します。
getElementsByTagName()（2-60 ページ）	与えられたタグ名を持つすべての要素のノード・リストを、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。
getElementsByTagNameNS()（2-60 ページ）	与えら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ノード・リストを、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。

表 2-11 XMLDocument のメソッドの概要（続き）

メソッド	説明
getEncoding()（2-61 ページ）	<?xml...?> タグに格納される文字コード情報、またはユーザー定義の出力コード（文字コード情報より後で設定された場合）を戻します。
getIDHashtable()（2-61 ページ）	XML DOM ツリーの ID 要素ハッシュテーブルを戻します。
getImplementation()（2-61 ページ）	このドキュメントを処理する DOM インプリメンテーション・オブジェクトを戻します。
getLineNumber()（2-61 ページ）	行番号のデバッグ情報を戻します。
getNodeType()（2-62 ページ）	基礎となるオブジェクトの型を表すコードを戻します。
getOwnerDocument()（2-62 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。
getStandalone()（2-62 ページ）	スタンドアロン情報を取得します。これは <?xml...?> タグに格納されているスタンドアロン属性です。
getSystemId()（2-62 ページ）	この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()（2-63 ページ）	この要素に含まれている、マークアップされていないテキストを戻します。
getVersion()（2-63 ページ）	<?xml...?> タグに格納されるバージョン情報を設定します。
importNode()（2-63 ページ）	他のドキュメントからこのドキュメントにノードをインポートします。
insertBefore()（2-64 ページ）	newChild ノードを既存の子ノード refChild の前に挿入します。
print()（2-64 ページ）	このドキュメントの内容を特定の出力に書き込みます。
printExternalDTD()（2-65 ページ）	このドキュメントの内容を特定の出力ストリームに書き込みます。
readExternal()（2-66 ページ）	writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
removeChild()（2-66 ページ）	この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()（2-67 ページ）	子のノード・リスト内の子ノード oldChild を newChild と置換し、oldChild ノードを戻します。
reportSAXEvents()（2-67 ページ）	DOM ツリーから SAX イベントを通知します。
setDoctype()（2-68 ページ）	ドキュメントの doctype URI を設定します。

表 2-11 XMLDocument のメソッドの概要（続き）

メソッド	説明
setEncoding()（2-68 ページ）	出力用の文字コードを設定します。
setLocale()（2-68 ページ）	エラー・レポート用のロケールを設定します。
setNodeContext()（2-69 ページ）	ノード・コンテキストを設定します。
setParsedDoctype()（2-69 ページ）	sysid の解析によって doctype オブジェクトを設定します。
setStandalone()（2-69 ページ）	<?xml...?> タグに格納されるスタンドアロン情報を設定します。
setVersion()（2-70 ページ）	<?xml...?> タグに格納されるバージョン番号を設定します。
validateElementContent()（2-70 ページ）	要素ノードの内容を検証します。
writeExternal()（2-70 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

XMLDocument()

空のドキュメントを作成します。

構文

```
public  XMLDocument();
```

addID()

この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
id	ID 値
e	ID に対応する XMLElement

adoptNode()

他のドキュメントからこのドキュメントにノードを適用します。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削除されています。次の DOMException が発生します。

- NOT_SUPPORTED_ERR: ノードのタイプが適用されていてもサポートされていない場合に発生します。

構文

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

パラメータ	説明
srcNode	適用されるノード

appendChild()

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: このノードが要素ノードのタイプの子を許可しないタイプであった場合に発生します。
- WRONG_DOCUMENT_ERR: これとは異なるドキュメントから要素が作成された場合に発生します。

構文

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

パラメータ	説明
newNode	追加される新しいノード

cloneNode()

この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、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、子であるテキスト・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。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。

構文

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

パラメータ	説明
deep	true の場合、指定し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、ノードのみが（ノードが要素の場合はその属性も）複製されます。

createAttribute()

与えられた名前の属性を作成します。作成された属性インスタンスは、setAttribute メソッドを使用して、要素に対して設定できます。次の DOMException が発生します。

- INVALID_CHARACTER_ERR: 指定した名前が無効な文字を含む場合に発生します。

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()

与えら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: 指定した修飾名が無効な文字を含む場合に発生します。
- NAMESPACE_ERR: 修飾名が間違った形式であるか、修飾名が接頭辞を持っていて名前空間 URI が null または空の文字列であるか、または修飾名が「xml」の接頭辞を持っていて名前空間 URI が「http://www.w3.org/2000/xmlns/」以外のものである場合に発生します。

構文

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

パラメータ	説明
namespaceURI	作成される属性 / 要素の名前空間
qualifiedName	作成される属性 / 要素の修飾名

createCDATASection()

指定した文字列を値に持つ CDATA セクション・ノードを作成します。次の 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	CDATA セクションの内容のデータ

createComment()

指定した文字列を値に持つコメント・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()

空のドキュメント・フラグメント・オブジェクトを作成します。

構文

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

createElement()

指定し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対して、属性を直接設定できます。次の DOMException が発生します。

- INVALID_CHARACTER_ERR: 指定した名前が無効な文字を含む場合に発生します。

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()

与えられた修飾名および名前空間 URI を持つ要素を作成します。次の DOMException が発生します。

- INVALID_CHARACTER_ERR: 指定した修飾名が無効な文字を含む場合に発生します。
- NAMESPACE_ERR: 修飾名が間違った形式であるか、修飾名が接頭辞を持っていて名前空間 URI が null または空の文字列であるか、または修飾名が「xml」の接頭辞を持っていて名前空間 URI が「http://www.w3.org/XML/1998/namespace」以外のものである場合に発生します。

構文

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

パラメータ	説明
namespaceURI	作成される属性 / 要素の名前空間
qualifiedName	作成される属性 / 要素の修飾名

createEntityReference()

実体参照オブジェクトを作成します。次の DOMException が発生します。

- INVALID_CHARACTER_ERR: 指定した名前が無効な文字を含む場合に発生します。

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名

createEvent()

指定した型のイベント・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()

指定した型の変異イベント・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	変異イベントの型

createNodeIterator()

指定されたルート、論理ビューに含めるノードのタイプを決定するフラグ、ノード用のフィルタ、エンティティへの参照やその子孫を含むことができるかどうかを判別するフラグを持つノード・イテレータを作成します。次の DOMException が発生します。

- NOT_SUPPORTED_ERR: ノード・イテレータが指定されたルートで作成できない場合に発生します。

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 / ツリー・ウォーカーにどのようなタイプのノードが含まれるのかを示すフラグ
filter	イテレータ / ツリー・ウォーカーから不要のノードをフィルタにかけるノード・フィルタ
expandEntityReference	実体参照の走査を示すフラグ

createProcessingInstruction()

指定した名前およびデータ文字列を値に持つ処理命令ノードを作成します。次の DOMException が発生します。

- INVALID_CHARACTER_ERR: 無効な文字が指定された場合に発生します。

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()

ドキュメントの始まりに最初と最後の境界ポイントを持つドキュメント範囲オブジェクトを作成します。

構文

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

createRangeEvent()

指定し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	イベントの型

createTextNode()

指定した文字列を値に持つテキスト・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()

指定した型の走査イベント・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	走査イベントの型

createTreeWalker()

指定されたルート、論理ビューに含めるノードのタイプを決定するフラグ、ノード用のフィルタ、エンティティへの参照やその子孫を含むことができるかどうかを判別するフラグを持つノード・イテレータを作成します。次の DOMException が発生します。

- NOT_SUPPORTED_ERR: ノード・イテレータが指定されたルートで作成できない場合に発生します。

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 / ツリー・ウォーカーにどのようなタイプのノードが含まれるのかを示すフラグ
filter	イテレータ / ツリー・ウォーカーから不要のノードをフィルタにかけるノード・フィルタ
expandEntityReference	実体参照の走査を示すフラグ

expectedElements()

要素に追加できる要素名のベクトルを戻します。

構文

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

パラメータ	説明
e	要素

getColumnNumber()

列番号のデバッグ情報を戻します。

構文

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

getDebugMode()

デバッグ・フラグを戻します。

構文

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

getDoctype()

このドキュメントに対応する DTD を戻します。DTD のない XML 文書の場合は、null を戻します。DOM レベル 1 の仕様では、DTD の編集はサポートされないことに注意してください。

構文

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

getDocumentElement()

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()

elementId によって ID を与えられた要素を戻します。そのような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、動作は定義されません。

構文

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

パラメータ	説明
elementId	elementId は一致する ID 要素を取得するために使用されます。

getElementsByTagName()

与えられたタグ名を持つすべての要素を含んだ 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。

構文

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

パラメータ	説明
tagname	一致するタグ名。特殊値「*」を指定すると、すべてのタグに一致します。

getElementsByTagNameNS()

与えら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ノード・リストを、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。

構文

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

パラメータ	説明
namespaceURI	要求された要素の名前空間
localName	要求された要素のローカル名

getEncoding()

<?xml...?> タグに格納される文字コード情報、またはユーザー定義の出力コード（文字コード情報より後で設定された場合）を戻します。

構文

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

getIDHashtable()

XML DOM ツリーの ID 要素ハッシュテーブルを戻します。

構文

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

getImplementation()

このドキュメントを処理する DOM インプリメンテーション・オブジェクトを戻します。DOM アプリケーションは、複数実装からオブジェクトを使用できます。

構文

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

getLineNumber()

行番号のデバッグ情報を戻します。

構文

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

getNodeType()

基礎となる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()

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。このノードがドキュメントである場合は、null が戻されます。

構文

```
public org.w3c.dom.Document getOwnerDocument();
```

getStandalone()

スタンドアロン情報を取得します。これは <?xml...?> タグに格納されているスタンドアロン属性です。

構文

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

getSystemId()

この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()

この要素に含まれている、マークアップされていないテキストを戻します。テキスト要素に関しては、これは生データです。子ノードを持つ要素に関して、このメソッドはサブツリー全体を走査し、各ターミナル・テキスト要素にテキストを追加して、サブツリーに対しXML マークアップを効率的に削除します。たとえば、XML 文書に「William Shakespeare」が含まれている場合、XMLDocument.getText とすると、「William Shakespeare」が戻されます。

構文

```
public String getText();
```

getVersion()

<?xml...?> タグに格納されるバージョン情報を設定します。

構文

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

importNode()

他のドキュメントからこのドキュメントに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ていません。すべてのノードに関して、ノードをインポートすると、インポートした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。そのオブジェクトは、ソース・ノードの nodeName および nodeType と同一の属性値に加え、名前空間（接頭辞、ローカル名、名前空間 URI）に関連する属性を持っています。次の DOMException が発生します。

- NOT_SUPPORTED_ERR: ノードのタイプがインポートされていてもサポートされていない場合に発生します。

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です。
deep	インポートするノードの子孫があるかどうかを示します。

insertBefore()

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。これは、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。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR: このノードが newChild ノード・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このノードの祖先クラスの 1 つである場合に発生します。
- WRONG_DOCUMENT_ERR: この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 newChild が作成された場合に発生します。
- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
- NOT_FOUND_ERR: refChild がこのノードの子でない場合に発生します。

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	参照ノードであり、このノードの前に新しいノードが挿入されます。

print()

このドキュメントの内容を特定の出力に書き込みます。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);	このドキュメントの内容を特定の出力ストリームに書き込みます。
public void print(java.io.OutputStream out, String enc);	このドキュメントの内容を特定のエンコードされた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);`	このドキュメントの内容を特定のプリント・ライターに書き込みます。
`public void print(` `  PrintDriver pd);`	このドキュメントの内容を特定のプリント・ドライバに書き込みます。

パラメータ	説明
out	書込み先となる出力
enc	出力に使用されるエンコーディング
pd	各ノードの書込みに使用されるプリント・ドライバ

printExternalDTD()

このドキュメントの内容を特定の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);`	このドキュメントの内容を特定の出力ストリームに書き込みます。

`public void printExternalDTD(`
`  java.io.OutputStream out,`
`  String enc);`
	このドキュメントの内容を特定のエンコードされた出力ストリームに書き込みます。

`public void printExternalDTD(` `  java.io.PrintWriter out);`	このドキュメントの内容を特定のプリント・ライターに書き込みます。

パラメータ	説明
out	書込み先となる出力
enc	出力に使用されるエンコーディング

readExternal()

writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従っ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み取り中にエラーが起きた場合に発生します。
- ClassNotFoundException - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

removeChild()

この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: このドキュメントが読取り専用である場合に発生します。
- NOT_FOUND_ERR: oldChild がこのノードの子でない場合に発生します。

構文

```
public org.w3c.dom.Node removeChild( org.w3c.dom.Node elem);
```

パラメータ	説明
elem	削除される要素

replaceChild()

子のノード・リスト内の子ノード oldChild を newChild と置換し、oldChild ノードを戻します。これは、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。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR: このノードが newChild ノードのタイプの子を許可しないタイプであった場合に発生します。
- WRONG_DOCUMENT_ERR: これとは異なるドキュメントから newChild が作成された場合に発生します。
- NOT_FOUND_ERR: oldChild がこのノードの子でない場合に発生します。

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()

DOM ツリーから SAX イベントを通知します。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()

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
root	ルート要素名
sysid	doctype のシステム識別子
pubid	doctype の公開識別子（null の場合もあります）

setEncoding()

出力用の文字コードを設定します。また、最終的には <?xml...?> タグに格納される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、この設定を行いません。したがって、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	設定する文字コード

setLocale()

エラー・レポート用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・レポート用のロケール

setNodeContext()

ノード・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()

sysid の解析によって 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
root	ルート要素名
sysid	doctype のシステム識別子
pubid	doctype の公開識別子（null の場合もあります）

setStandalone()

<?xml...?> タグに格納される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()

<?xml...?> タグに格納される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()

要素ノードの内容を検証します。有効な場合は true、無効な場合は false を戻します。

構文

```
public boolean validateElementContent( org.w3c.dom.Element elem);
```

パラメータ	説明
elem	検証される要素

writeExternal()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。シリアライズ / 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	シリアライズ / 圧縮ストリームの書込みに使用される ObjectOutput ストリーム

XMLDocumentFragment クラス

DOM の DocumentFragment インタフェースを実装します。XMLNode ではなく XMLElement を拡張することで、要素として扱えるようになります。「DocumentFragment」、「NodeFactory」および「DOMParser.setNodeFactory()」も参照してください。

構文

```
public class XMLDocumentFragment implements java.io.Serializable
```

表 2-12 XMLDocumentFragment のメソッドの概要

メソッド	説明
getAttributes()（2-71 ページ）	XMLDocumentFragment の属性を戻します。
getNodeType()（2-71 ページ）	基礎となるオブジェクトの型を表すコードを戻します。
getParentNode()（2-72 ページ）	ノードの親を戻します。

getAttributes()

XMLDocumentFragment（空の NamedNodeMap）の属性を戻します。

構文

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

getNodeType()

基礎となる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()

ノードの親を戻します。

構文

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

XMLDOMException クラス

DOM 例外を発生するために使用されます。

構文

```
public class XMLDOMException
```

XMLDOMException()

XMLDOMException 例外を構成します。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持つ XMLDOMException 例外を構成します。
`public XMLDOMException( short code, String mess);`	指定されたメッセージおよびエラー・コードを持つ XMLDOMException 例外を構成します。

パラメータ	説明
code	DOM インタフェースで示されるコードで、デフォルト・メッセージを使用します。
mess	XMLDOMException の構成で使用されるメッセージです。

XMLDOMImplementation

DOMImplementation を実装します。

構文

```
public class XMLDOMImplementation implements java.io.Serializable
```

表 2-13 XMLDOMImplementation のメソッドの概要

メソッド	説明
XMLDOMImplementation() (2-74 ページ)	XMLDOMImplementation の新しいインスタンスを作成します。
createDocument() (2-75 ページ)	指定した DocumentType ノードと指定した名前を持つルート要素および空の DocumentType ノードを含む XMLDocument オブジェクトを作成します。
createDocumentType() (2-75 ページ)	ルート要素名およびシステム / 公開識別子を持つ DocumentType ノードを作成します。
hasFeature()（2-76 ページ）	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。
setFeature()（2-76 ページ）	指定された機能を設定します。

XMLDOMImplementation()

XMLDOMImplementation の新しいインスタンスを作成します。

構文

```
public  XMLDOMImplementation();
```

createDocument()

指定した DocumentType ノードと指定した名前を持つルート要素および空の DocumentType ノードを含む 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: 指定した修飾名が無効な文字を含む場合に発生します。
- NAMESPACE_ERR: 修飾名が間違った形式であるか、修飾名が接頭辞を持っていて名前空間 URI が null または空の文字列であるか、または修飾名が「xml」の接頭辞を持っていて名前空間 URI が「http://www.w3.org/XML/1998/namespace」以外のものである場合に発生します。
- WRONG_DOCUMENT_ERR: doctype がすでに異なるドキュメントに使用されているか、または異なる実装から作成されている場合に発生します。

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	ドキュメントのルート要素の名前空間
qualifiedName	ドキュメント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()

ルート要素名およびシステム / 公開識別子を持つ 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: 指定した修飾名が無効な文字を含む場合に発生します。
- NAMESPACE_ERR: 修飾名が間違った形式の場合に発生します。

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()

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。機能が実装された場合は true、それ以外の場合は false を戻します。

構文

```
public boolean hasFeature( String feature,
                           String version);
```

パラメータ	説明
feature	テストされている機能
version	テストされている機能のバージョン

setFeature()

指定された機能を設定します。機能が設定できない場合には DOMException が発生します。

構文

```
public void setFeature( String feature);
```

パラメータ	説明
feature	DOM 機能

XMLElement クラス

DOM Element インタフェースを実装します。

構文

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

表 2-14 XMLElement のメソッドの概要

メソッド	説明
XMLElement()（2-79 ページ）	デフォルトのコンストラクタです。
cloneNode()（2-79 ページ）	この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()（2-80 ページ）	名前で属性値を戻します。その属性が設定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()（2-80 ページ）	名前で属性ノードを戻します。または該当する属性が存在しない場合は null を戻します。
getAttributeNodeNS()（2-80 ページ）	指定の名前空間 URI とローカル名を持つ属性が存在している場合は、それを戻します。それ以外の場合は null を戻します。
getAttributeNS()（2-81 ページ）	名前空間 URI およびローカル名を持つ属性が存在している場合はその値を、それ以外の場合は null を戻します。
getAttributes()（2-81 ページ）	このノードが要素の場合はその属性を含む NamedNodeMap を、それ以外の場合は null を戻します。
getChildrenByTagName()（2-81 ページ）	与えられたタグ名を持つすべての子ノードのノード・リストを戻します。次の表に、オプションを示します。
getElementsByTagName()（2-82 ページ）	与えられたタグ名を持つすべての要素のノード・リストを、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。
getElementsByTagNameNS()（2-82 ページ）	与えら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ノード・リストを、この要素ツリーの先行順走査で検出した順に戻します。
getExpandedName()（2-83 ページ）	要素の解決済完全名を戻します。
getFirstAttribute()（2-80 ページ）	最初の属性ノードを取得します。属性が存在しない場合は null になります。

表 2-14 XMLElement のメソッドの概要（続き）

メソッド	説明
getLocalName()（2-83 ページ）	要素のローカル名を戻します。
getNamespaceURI()（2-83 ページ）	要素の名前空間 URI を戻します。
getNodeType()（2-84 ページ）	基礎となるオブジェクトの型を表すコードを戻します。
getPrefix()（2-84 ページ）	要素の名前空間の接頭辞を戻します。
getQualifiedName()（2-84 ページ）	要素の修飾名を戻します。
getTagName()（2-84 ページ）	要素の名前を戻します。
hasAttribute()（2-85 ページ）	与えられた名前を持つ属性が、この要素で指定されたか、またはデフォルト値を持つ場合に true を戻します。
hasAttributeNS()（2-85 ページ）	与えられたローカル名および名前空間 URI を持つ属性が、この要素で指定されたか、またはデフォルト値を持つ場合に true を戻します。
hasAttributes()（2-85 ページ）	このノードが属性を持つ場合に true を戻します。
readExternal()（2-86 ページ）	入力ストリームを読み取り、入力ストリームの情報に従ってオブジェクトを再生成することで、`writeExternal()` によって書き込まれた情報をリストアします。
removeAttribute()（2-86 ページ）	指定した属性名の属性を削除します。
removeAttributeNode()（2-87 ページ）	指定する属性を削除し、戻します。
removeAttributeNS()（2-87 ページ）	指定するローカル名と名前空間 URI の属性を削除します。
reportSAXEvents()（2-88 ページ）	DOM ツリーから SAX イベントを通知します。
resolveNamespacePrefix()（2-88 ページ）	与えられた名前空間の接頭辞の有効範囲で、この要素の名前空間定義を検索します。
setAttribute()（2-88 ページ）	新しい属性を追加します。
setAttributeNode()（2-89 ページ）	新しい属性ノードを追加します。
setAttributeNodeNS()（2-90 ページ）	新しい名前空間を認識する属性ノードを追加します。

表 2-14 XMLElement のメソッドの概要（続き）

メソッド	説明
validateContent()（2-91 ページ）	要素ノードの内容を検証します。
writeExternal()（2-91 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

XMLElement()

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、この DOM ノードがデシリアライズ / 解凍中の場合にのみ使用できることに注意してください。シリアライズ / 圧縮ストリームから DOM ノードを構成するためにこのノードをデシリアライズするには、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 XMLElement 作成はすべて、XMLDocument の `createElement()` を使用できます。

構文

```
public  XMLElement();
```

cloneNode()

この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、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、子であるテキスト・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。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。

構文

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

パラメータ	説明
`true`	`true` の場合、指定し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、ノードのみが（ノードが要素の場合はその属性も）複製されます。

getAttribute()

名前で属性値を戻します。その属性が設定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()

名前で属性ノードを戻します。または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()

指定の名前空間 URI とローカル名を持つ属性が存在する場合は、その属性を戻します。それ以外の場合は 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	要求された属性ノードの名前空間
localName	要求された属性ノードのローカル名

getAttributeNS()

存在している場合は、名前空間 URI およびローカル名を持つ属性が存在している場合は、その値を戻します。それ以外の場合は null を戻します。

構文

```
public String getAttributeNS( String namespaceURI,
                              String localName);
```

パラメータ	説明
namespaceURI	要求された属性の名前空間
localName	要求された属性のローカル名

getAttributes()

ノードが要素の場合はその属性を含む NamedNodeMap を、それ以外の場合は null を戻します。

構文

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

getChildrenByTagName()

与えられたタグ名を持つすべての子ノードのノード・リストを戻します。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);	与えられたタグ名を持つすべての子ノードのノード・リストを戻します。
public org.w3c.dom.NodeList getChildrenByTagName(String name, String ns);	与えられたタグ名および名前空間を持つすべての子ノードのノード・リストを戻します。

パラメータ	説明
name	一致するタグ名（ローカル名）
ns	名前空間

getElementsByTagName()

与えられたタグ名を持つすべての要素を含んだ 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。

構文

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

パラメータ	説明
tagname	一致するタグ名。特殊値「*」を指定すると、すべてのタグに一致します。

getElementsByTagNameNS()

与えら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ノード・リストを、この要素ツリーの先行順走査で検出した順に戻します。

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()

要素の解決済完全名を戻します。

構文

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

getFirstAttribute()

最初の属性ノードを取得します。属性が存在しない場合は null になります。

構文

```
public XMLNode getFirstAttribute();
```

getLocalName()

要素のローカル名を戻します。

構文

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

getNamespaceURI()

要素の名前空間 URI を戻します。

構文

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

getNodeType()

基礎となる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()

要素の名前空間の接頭辞を戻します。

構文

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

getQualifiedName()

要素の修飾名を戻します。

構文

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

getTagName()

要素の名前を戻します。たとえば、<elementExample id="demo">... </elementExample> とした場合、`tagName` は `elementExample` の値を持っています。XML では、DOM のすべての操作と同様、大 / 小文字が区別されることに注意してください。HTML DOM は、ソースである HTML ドキュメントでの大 / 小文字にかかわらず、HTML 要素のタグ名を大文字の正規形で戻します。

構文

```
public String getTagName();
```

hasAttribute()

与えられた名前を持つ属性が、この要素で指定されたか、またはデフォルト値を持つ場合に 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	存在型がチェックされている属性の名前

hasAttributeNS()

与えられたローカル名および名前空間 URI を持つ属性が、この要素で指定されたか、またはデフォルト値を持つ場合に 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	存在型がチェックされている属性の名前空間
localName	存在型がチェックされている属性のローカル名

hasAttributes()

このノードが属性を持つ場合に true を戻します。それ以外の場合は false を戻します。

構文

```
public boolean hasAttributes();
```

readExternal()

入力ストリームを読み取り、入力ストリームの情報に従ってオブジェクトを再生成することで、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み取り中に例外が発生した場合に発生します。
- ClassNotFoundException - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

removeAttribute()

指定した属性名の属性を削除します。削除し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。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。

構文

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

パラメータ	説明
name	削除する属性の名前

removeAttributeNode()

指定された属性を削除し、戻します。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
- NOT_FOUND_ERR: oldAttr が要素の属性でない場合に発生します。

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属性ノード。削除し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。

removeAttributeNS()

指定するローカル名と名前空間 URI の属性を削除します。次の DOMException が発生します。

- NO_MODIFICATIONS_ALLOWED_ERR: この要素が読取り専用である場合に発生します。

構文

```
public void removeAttributeNS( String namespaceURI,
                               String localName);
```

パラメータ	説明
namespaceURI	削除される属性の名前空間
localName	削除される属性のローカル名

reportSAXEvents()

DOM ツリーから SAX イベントを通知します。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()

与えられた名前空間の接頭辞の有効範囲で、この要素の名前空間定義を検索します。

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となります。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()

新しい属性を追加します。指定する属性名を持つ属性がすでに要素に存在する場合は、その値が、該当する値パラメータの値に変更されます。この値は、単純な文字列であり、設定されている場合は解析されません。このため、すべてのマークアップ（実体参照として認識される構文など）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値を割り当てるには、1 つの属性ノードに加えて、任意のテキスト・ノードおよび実体参照ノードを作成し、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。このメソッドは認識されていない名前空間であるため、新しい属性がこのメソッドを通じて追加された場合、名前空間表は更新されません。次の DOMException が発生します。

- INVALID_CHARACTER_ERR: 指定した名前が無効な文字を含む場合に発生します。
- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。

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()

新しい属性を追加します。指定する属性名を持つ属性がすでに要素に存在する場合は、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、前に存在した属性ノードが戻されます。それ以外の場合は null が戻されます。次の DOMException が発生します。

- WRONG_DOCUMENT_ERR: この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 newAttr が作成された場合に発生します。
- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
- INUSE_ATTRIBUTE_ERR: newAttr がすでに他の要素オブジェクトの属性である場合に発生します。DOM ユーザーは、属性ノードを明示的に複製して、それを他の要素で再度使用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	属性リストに追加される属性

setAttributeNodeNS()

新しい属性を追加します。次の DOMException が発生します。

- INVALID_CHARACTER_ERR: 指定した名前が無効な文字を含む場合に発生します。
- NAMESPACE_ERR: 修飾名が間違った形式であるか、修飾名が接頭辞を持っていて名前空間 URI が null または空の文字列であるか、修飾名が「xmlns」で名前空間 URI が「http://www.w3.org/2000/xmlns/」以外のものであるか、または修飾名が「xml」の接頭辞を持っていて名前空間 URI が「http://www.w3.org/XML/1998/namespaces」以外のものである場合に発生します。
- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
- WRONG_DOCUMENT_ERR: このドキュメントを作成したものとは異なるドキュメントから newAttr が作成された場合に発生します。
- INUSE_ATTRIBUTE_ERR: newAttr がすでに他の要素オブジェクトの属性である場合に発生します。

次の表に、オプションを示します。

構文	説明
`public org.w3c.dom.Attr setAttributeNodeNS( org.w3c.dom.Attr newAttr);`	新しい属性ノードを追加し、戻します。指定する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構成し、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変更され、その値は値パラメータに変更されます。この値は、単純な文字列であり、設定されている場合は解析されません。このため、すべてのマークアップ（実体参照として認識される構文など）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	属性リストに追加される属性
namespaceURI	追加される属性の名前空間
localName	追加される属性のローカル名
value	追加される属性の値

validateContent()

要素ノードの内容を検証します。有効な場合は true、無効な場合は 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 を使用している要素ノードの内容を検証します。
`public boolean validateContent( oracle.xml.parser.schema.XMLSchema schema);`	指定した XML スキーマのパラメータ・スキーマに対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, String mode);`	指定したモードの指定した XML スキーマに対して要素の内容を検証します。

パラメータ	説明
dtd	要素の検証に使用される DTD オブジェクト
schema	要素の検証に使用される XML スキーマ・オブジェクト
mode	検証モード

writeExternal()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、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	シリアライズ / 圧縮の書込みに使用される ObjectOutput ストリーム

XMLEntity クラス

DOM Entity インタフェースを実装し、XML の内部 / 外部エンティティを XML DTD で定義されるとおりに表します。

構文

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

表 2-15 XMLEntity のメソッドの概要

メソッド	説明
XMLEntity()（2-93 ページ）	デフォルトのコンストラクタです。
cloneNode()（2-93 ページ）	この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()（2-93 ページ）	基礎となるオブジェクトの型を表すコードを戻します。
getNodeValue()（2-94 ページ）	ノード・タイプに応じて、ノードの値を戻します。
getNotationName()（2-94 ページ）	解析対象外エンティティの場合、そのエンティティの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()（2-94 ページ）	公開識別子を戻します。
getSystemId()（2-94 ページ）	システム識別子を戻します。
readExternal()（2-95 ページ）	writeExternal() メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
setNodeValue()（2-95 ページ）	エンティティの値を設定します。
writeExternal()（2-95 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

XMLEntity()

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、この DOM ノードがデシリアライズ / 解凍中の場合にのみ使用できることに注意してください。シリアライズ / 圧縮ストリームから DOM ノードを構成するためにこのノードをデシリアライズするには、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()

この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、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、子であるテキスト・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。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。

構文

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

パラメータ	説明
deep	true の場合、指定し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、ノードのみが（ノードが要素の場合はその属性も）複製されます。

getNodeType()

基礎となる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()

ノード・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: ノードが読取り専用である場合に発生します。
- DOMSTRING_SIZE_ERR: ノードが、実装プラットフォーム上で DOMString 変数より多い文字を戻す場合に発生します。

構文

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

getNotationName()

解析対象外エンティティの場合、そのエンティティの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()

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()

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()

writeExternal() メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従っ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み取り中にエラーが起きた場合に発生します。
- ClassNotFoundException - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

setNodeValue()

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()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、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	シリアライズ / 圧縮ストリームの書込みに使用される ObjectOutput ストリーム

XMLEntityReference クラス

DOM の EntityReference インタフェースを実装します。

構文

```
public class XMLEntityReference implements java.lang.Cloneable,
java.io.Externalizable
```

表 2-16 XMLEntityReference のメソッドの概要

メソッド	説明
XMLEntityReference()（2-96 ページ）	デフォルトのコンストラクタ
getNodeType()（2-96 ページ）	基礎となるオブジェクトの型を表すコードを戻します。
readExternal()（2-97 ページ）	writeExternal() メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
writeExternal()（2-97 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

XMLEntityReference()

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、この DOM ノードがデシリアライズ / 解凍中の場合にのみ使用できることに注意してください。シリアライズ / 圧縮ストリームから DOM ノードを構成するためにこのノードをデシリアライズするには、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()

基礎となる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()

writeExternal() メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従っ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み取り中にエラーが起きた場合に発生します。
- `ClassNotFoundException` - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

writeExternal()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、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される ObjectOutput ストリーム

XMLNode クラス

DOM の Node インタフェースを実装し、DOM 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ためのメカニズムとして含まれます。特定の nodeType のこれらの属性（要素の nodeValue、コメントの attributes など）が明確にマッピングされていない場合は、null が戻されます。派生クラスには、関連情報を取得および設定するためにより有効な、別のメカニズムが含まれる場合があることに注意してください。XMLNSNode のかわりに XMLNode を拡張するこの DOM ノードは、DOM 仕様で定義されたノード名を修正しました。また、このクラスを拡張する子ノードを持てない唯一のノードでもあります。

構文

```
public abstract class XMLNode implements java.lang.Cloneable, java.io.Externalizable
```

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です。
capturing	public static final String capturing	イベント・ターゲットに処理される前に、イベント・ターゲットの祖先のひとつによって処理できます。
DOMAttrModified	public static final String DOMAttrModified	ノードで変更された属性です。
DOMCharacterDataModified	public static final String DOMCharacterDataModified	ノード内の文字データが変更されています。
DOMNodeInserted	public static final String DOMNodeInserted	別のノードの子がノードに追加されています。
DOMNodeInsertedIntoDocument	public static final String DOMNodeInsertedIntoDocument	ノードの直接挿入、またはそれを含むサブツリーへの挿入のいずれかにより、ノードがドキュメントに挿入されています。
DOMNodeRemoved	public static final String DOMNodeRemoved	ノードはその親ノードから削除されました。

表 2-17 XMLNode のフィールド（続き）

フィールド	構文	説明
DOMNodeRemovedFromDocument	public static final String DOMNodeRemovedFromDocument	ノードの直接削除、またはそれを含むサブツリーの削除のいずれかにより、ノードがドキュメントから削除されています。
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一般イベントです。より特定したイベントのかわりに使用できます。
ELEMENTDECL	public static final short ELEMENTDECL	要素宣言です。
noncapturing	public static final String noncapturing	イベント・ターゲットの子孫に先に処理されることなく、イベント・ターゲットにより処理されます。
RANGE_DELETE_EVENT	public static final String RANGE_DELETE_EVENT	範囲イベントを削除するフラグです。
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です。
RANGE_INSERT_EVENT	public static final String RANGE_INSERT_EVENT	範囲イベントを設定するフラグです。
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です。
RANGE_REPLACE_EVENT	public static final String RANGE_REPLACE_EVENT	範囲イベントを置換するフラグです。
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です。
TRAVERSAL_DELETE_EVENT	public static final String TRAVERSAL_DELETE_EVENT	走査削除イベントを設定するフラグです。
TRAVERSAL_REPLACE_EVENT	public static final String TRAVERSAL_REPLACE_EVENT	走査置換イベントを設定するフラグです。
XMLDECL_NODE	public static final short XMLDECL_NODE	属性宣言ノードです。

表 2-18 XMLNode のメソッドの概要

メソッド	説明
XMLNode()（2-102 ページ）	新しい XMLNode を構成します。
addEventListener()（2-103 ページ）	イベント・ターゲット（ノード）のイベント・リスナーを登録します。
appendChild()（2-103 ページ）	ノードの子リストの最後に newChild ノードを追加し、新しいノードを戻します。
cloneNode()（2-104 ページ）	この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()（2-104 ページ）	実装イベント・モデルにイベントを送ります。
getAttributes()（2-104 ページ）	このノードの属性を含む NamedNodeMap を戻します。
getChildNodes()（2-105 ページ）	ノード・リストにある、このノードのすべての子を戻します。
getColumnNumber()（2-105 ページ）	列番号のデバッグ情報を戻します。
getDebugMode()（2-105 ページ）	デバッグ情報モードを戻します。
getFirstChild()（2-105 ページ）	ノードの最初の子ノードを戻します。
getLastChild()（2-106 ページ）	ノードの最後の子ノードを戻します。
getLineNumber()（2-106 ページ）	行番号のデバッグ情報を戻します。
getLocalName()（2-106 ページ）	名前空間が意味を持つよう、ノード・タイプに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()（2-106 ページ）	名前空間が意味を持つよう、ノード・タイプに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()（2-107 ページ）	このノードの直前のノードを戻します。
getNodeName()（2-107 ページ）	ノードの名前を戻します。
getNodeType()（2-107 ページ）	ノードのタイプを戻します。
getNodeValue()（2-107 ページ）	ノード・タイプに応じて、ノードの値を戻します。
getOwnerDocument()（2-108 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()（2-108 ページ）	ノードの親を戻します。

表 2-18 XMLNode のメソッドの概要（続き）

メソッド	説明
getPrefix()（2-108 ページ）	名前空間が意味を持つよう、ノード・タイプにオーバーライドされたノードの接頭辞を戻します。
getPreviousSibling()（2-108 ページ）	ノードの直前のノードを戻します。
getProperty()（2-109 ページ）	ノードのプロパティの値を戻します。
getSystemId()（2-109 ページ）	この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()（2-109 ページ）	この要素に含まれている、マークアップされていないテキストを戻します。
hasAttributes()（2-109 ページ）	このノードが要素である場合、属性があるかどうかを決定します。
hasChildNodes()（2-110 ページ）	ノードに子ノードがあるかどうかを判別します。
insertBefore()（2-110 ページ）	newChild ノードを既存の子ノード refChild の前に挿入します。
isNodeFlag()（2-111 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()（2-111 ページ）	DOM インプリメンテーションが固有の機能を実装し、その機能がノードにサポートされているかどうかをテストします。
print()（2-111 ページ）	このノードの内容を出力に書き込みます。
readExternal()（2-112 ページ）	writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
removeChild()（2-112 ページ）	oldChild が示す子ノードを子リストから削除し、それを戻します。
removeEventListener()（2-113 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()（2-113 ページ）	子のノード・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()（2-114 ページ）	DOM ツリーから SAX イベントを通知します。SAXException が発生します。
resetNodeFlag()（2-114 ページ）	ノード・フラグ情報をリセットします。
selectNodes()（2-114 ページ）	与えられたパターンに一致するツリー内のノードを、ノード・リストとして戻します。

表 2-18 XMLNode のメソッドの概要（続き）

メソッド	説明
selectSingleNode()（2-115 ページ）	与えられたパターンに一致するツリー内の最初のノードを戻します。
setDebugInfo()（2-115 ページ）	ノードのデバッグ情報を設定します。
setNodeFlag()（2-116 ページ）	ノード・フラグ情報を設定します。
setNodeValue()（2-116 ページ）	このノードの値を、ノード・タイプに応じて設定します。
setPrefix()（2-116 ページ）	名前空間が意味を持つよう、ノード・タイプにオーバーライドされたノードの接頭辞を設定します。
setProperty()（2-117 ページ）	ノードのプロパティを設定します。
transformNode()（2-117 ページ）	与えられたスタイルシートを使用して、ツリーにあるノードを変換し、結果のドキュメント・フラグメントを戻します。
valueOf()（2-117 ページ）	パターンに一致する、ツリー内の最初のノードの値を選択します。
writeExternal()（2-118 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

XMLNode()

新しい XMLNode を構成します。

構文

```
protected  XMLNode();
```

パラメータ	説明
tag	ノードの名前

addEventListener()

イベント・ターゲット（ノード）のイベント・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	リスナーが登録されているイベントの型
listener	リスナー・オブジェクト
useCapture	リスナーが取得を開始する場合に、それを示すフラグ

appendChild()

ノードの子リストの最後に newChild ノードを追加し、新しいノードを戻します。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。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR: このノードが newChild ノード・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クラスの 1 つである場合に発生します。
- WRONG_DOCUMENT_ERR: この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 newChild が作成された場合に発生します。
- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。

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。これがドキュメント・フラグメント・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、このノードの子リストに移動されます。

cloneNode()

この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、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、子であるテキスト・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。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。

構文

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

パラメータ	説明
deep	true の場合、指定し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、ノードのみが（ノードが要素の場合はその属性も）複製されます。

dispatchEvent()

実装イベント・モデルにイベントを送ります。dispatchEvent がコールされる前にイベントが初期化されたことでイベント・タイプが指定されていない場合、値 UNSPECIFIED_EVENT_TYPE 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()

ノードが要素の場合はその属性を含む NamedNodeMap を、それ以外の場合は null を戻します。

構文

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

getChildNodes()

ノード・リストにある、このノードのすべての子を戻します。子ノードが存在しない場合は、ノードを含まない NodeList が戻されます。戻された NodeList の内容は、たとえば、それが作成される基になったノード・オブジェクトの子ノードに変更があった場合、その変更が NodeList アクセッサが戻すノードにすぐに反映されるという意味で最新のものであり、ノード内容の静的スナップショットではありません。このことは、`getElementsByTagName` メソッドが戻す NodeList を含む、すべての NodeList についていえます。

構文

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

getColumnNumber()

列番号のデバッグ情報を戻します。

構文

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

getDebugMode()

デバッグ情報モードを戻します。

構文

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

getFirstChild()

ノードの最初の子ノードを戻します。該当するノードがない場合は、`null` を戻します。

構文

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

getLastChild()

ノードの最後の子ノードを戻します。該当するノードがない場合は、null を戻します。

構文

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

getLineNumber()

行番号のデバッグ情報を戻します。

構文

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

getLocalName()

名前空間が意味を持つよう、ノード・タイプに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()

名前空間が意味を持つよう、ノード・タイプに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()

このノードの直前のノードを戻します。該当するノードがない場合は、null を戻します。

構文

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

getNodeName()

ノードの名前を戻します。

構文

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

getNodeType()

ノードのタイプを戻します。

構文

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

getNodeValue()

ノード・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: ノードが読取り専用である場合に発生します。
- DOMSTRING_SIZE_ERR: ノードが実装プラットフォーム上で DOMString 変数より多い文字を戻す場合に発生します。

構文

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

getOwnerDocument()

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。これは、新しいノードを作成するために使用するドキュメント・オブジェクトでもあります。このノードがドキュメントである場合は、null が戻されます。

構文

```
public org.w3c.dom.Document getOwnerDocument();
```

getParentNode()

ノードの親を戻します。ドキュメント、ドキュメント・フラグメントおよび属性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、またはツリーから削除された場合は、null が戻されます。

構文

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

getPrefix()

名前空間が意味を持つよう、ノード・タイプに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()

ノードの直前のノードを戻します。該当するノードがない場合は、null を戻します。

構文

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

getProperty()

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	プロパティの名前

getSystemId()

この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()

この要素に含まれている、マークアップされていないテキストを戻します。テキスト要素に関しては、これは生データです。子ノードを持つ要素に関して、このメソッドはサブツリー全体を走査し、各ターミナル・テキスト要素にテキストを追加して、サブツリーに対し XML マークアップを効率的に削除します。

構文

```
public String getText();
```

hasAttributes()

このノードが要素である場合、属性があるかどうかを判別します。このノードが属性を持つ場合に true を戻します。それ以外の場合は false を戻します。

構文

```
public boolean hasAttributes();
```

hasChildNodes()

ノードに子ノードがあるかどうかを判別します。ノードが子ノードを持つ場合は true、子ノードを持たない場合は false を戻します。

構文

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

insertBefore()

newChild ノードを既存の子ノード refChild の前に挿入し、このノードを戻します。refChild が null の場合は、子リストの最後に 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。これは、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。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR: このノードが newChild ノード・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このノードの祖先クラスの 1 つである場合に発生します。
- WRONG_DOCUMENT_ERR: この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 newChild が作成された場合に発生します。
- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
- NOT_FOUND_ERR: refChild がこのノードの子でない場合に発生します。

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	参照ノードであり、このノードの前に新しいノードが挿入されます。

isNodeFlag()

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()

DOM インプリメンテーションが固有の機能を実装し、その機能がノードに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、それ以外の場合は false を戻します。

構文

```
public boolean isSupported( String feature, String version);
```

パラメータ	説明
feature	テストされている機能
version	テストされている機能のバージョン

print()

このノードの内容を出力に書き込みます。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);`	ノードの内容を特定の出力ストリームに書き込みます。
`public void print( java.io.OutputStream out, String enc);`	ノードの内容を特定のエンコードされた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);`	ノードの内容を、特定のプリント・ライターを使用して書き込みます。

パラメータ	説明
out	出力
enc	出力に使用されるエンコーディング

readExternal()

writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従っ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み取り中にエラーが起きた場合に発生します。
- `ClassNotFoundException` - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

removeChild()

`oldChild` が示す子ノードを子リストから削除し、それを戻します。次の `DOMException` が発生します。

- `NO_MODIFICATION_ALLOWED_ERR`: このノードが読取り専用である場合に発生します。
- `NOT_FOUND_ERR`: `oldChild` がこのノードの子でない場合に発生します。

構文

```
public org.w3c.dom.Node removeChild( org.w3c.dom.Node oldChild);
```

パラメータ	説明
oldChild	削除されたノード

removeEventListener()

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	リスナーが登録されているイベントの型
listener	リスナー・オブジェクト
useCapture	リスナーが取得を開始する場合に、それを示すフラグ

replaceChild()

子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。これは、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。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR: このノードが newChild ノード・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このノードの祖先クラスの 1 つである場合に発生します。
- WRONG_DOCUMENT_ERR: この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 newChild が作成された場合に発生します。
- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
- NOT_FOUND_ERR: oldChild がこのノードの子でない場合に発生します。

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()

DOM ツリーから SAX イベントを通知します。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()

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()

与えられたパターンに一致するツリー内のノードを、ノード・リストとして戻します。このメソッドでは、パターンは名前空間の接頭辞を含まないことを想定しています。XSLException は、一致検索中にエラーが起きた場合に発生します。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);	パターンを使用した一致
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た一致

パラメータ	説明
pattern	一致させる XSL パターン
nsr	与えられたパターンに一致する接頭辞を変換する NSResolver

selectSingleNode()

与えられたパターンに一致するツリー内の最初のノードを戻します。XSLException は、一致検索中にエラーが起きた場合に発生します。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);`	パターンを使用した一致
`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た一致

パラメータ	説明
pattern	一致させる XSL パターン
nsr	与えられたパターンに一致する接頭辞を変換する NSResolver

setDebugInfo()

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()

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()

このノードの値を、ノード・タイプに応じて設定します。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR: ノードが読取り専用である場合に発生します。
- DOMSTRING_SIZE_ERR: ノードが実装プラットフォーム上で DOMString 変数より多い文字を戻す場合に発生します。

構文

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

パラメータ	説明
nodeValue	設定するノード値

setPrefix()

名前空間が意味を持つよう、ノード・タイプにオーバーライドされたノードの接頭辞を設定します。次の 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()

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	プロパティの名前
propValue	プロパティの値

transformNode()

与えられたスタイルシートを使用して、ツリーにあるノードを変換し、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 XSLStylesheet

valueOf()

パターンに一致する、ツリー内の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。XSLException は、一致検索中にエラーが起きた場合に発生します。

構文	説明
public String valueOf(String pattern);	パターンを使用した一致
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た一致

パラメータ	説明
pattern	一致させる XSL パターン
nsr	与えられたパターンに一致する接頭辞を変換する NSResolver

writeExternal()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、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	シリアライズ / 圧縮ストリームの書込みに使用される ObjectOutput ストリーム

XMLNotation クラス

DOM Notation インタフェースを実装し、DTD で宣言される表記法を表します。

構文

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

表 2-19 XMLNotation のメソッドの概要

メソッド	説明
XMLNotation()（2-120 ページ）	デフォルトのコンストラクタです。
cloneNode()（2-120 ページ）	この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()（2-120 ページ）	表記法の名前を戻します。
getNodeType()（2-121 ページ）	基礎となるオブジェクトの型を表すコードを戻します。
getPublicId()（2-121 ページ）	公開識別子を戻します。指定されていない場合は null を戻します。
getSystemId()（2-121 ページ）	システム識別子を戻します。指定されていない場合は null を戻します。
readExternal()（2-121 ページ）	writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
setPublicId()（2-122 ページ）	公開識別子を設定します。
setSystemId()（2-122 ページ）	システム識別子を設定します。
writeExternal()（2-122 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

XMLNotation()

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、この DOM ノードがデシリアライズ / 解凍中の場合にのみ使用できることに注意してください。シリアライズ / 圧縮ストリームから DOM ノードを構成するためにこのノードをデシリアライズするには、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 XMLElement 作成はすべて `XMLNotation()` を使用します。

構文

```
public  XMLNotation();
```

cloneNode()

この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、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、子であるテキスト・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。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。

構文

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

パラメータ	説明
deep	`true` の場合、指定し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、ノードのみが複製されます。

getNodeName()

表記法の名前を戻します。

構文

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

getNodeType()

基礎となる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()

公開識別子を戻します。指定されていない場合は null を戻します。

構文

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

getSystemId()

システム識別子を戻します。指定されていない場合は null を戻します。

構文

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

readExternal()

writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従っ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み取り中にエラーが起きた場合に発生します。
- `ClassNotFoundException` - クラスが見つからない場合に発生します。

構文

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

パラメータ	説明
`in`	圧縮ストリームの読取りに使用される ObjectInput ストリーム

setPublicId()

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()

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()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、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	シリアライズ / 圧縮ストリームの書込みに使用される ObjectOutput ストリーム

XMLNSNode クラス

名前空間名および子にサポートを追加するため、XMLNode を拡張します。

構文

```
public class XMLNSNode extends oracle.xml.parser.v2.XMLNode implements
java.lang.Cloneable java.io.Externalizable,
```

表 2-20 XMLNSNode のメソッドの概要

メソッド	説明
XMLNSNode()（2-124 ページ）	新しい XMLNSNode を構成します。
addText()（2-124 ページ）	このノードにテキストを追加します。最後の子ノードがテキスト・ノードの場合は、最後の子ノードに追加します。
appendChild()（2-125 ページ）	このノードの子リストの最後に newChild ノードを追加します。
getChildNodes()（2-126 ページ）	ノード・リストにある、このノードのすべての子を戻します。
getFirstChild()（2-126 ページ）	ノードの最初の子ノードを戻します。
getLastChild()（2-126 ページ）	ノードの最後の子ノードを戻します。
getLocalName()（2-126 ページ）	名前空間が意味を持つよう、ノード・タイプに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()（2-127 ページ）	名前空間が意味を持つよう、ノード・タイプに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()（2-127 ページ）	ノード・タイプに応じて、ノードの名前を戻します。
getPrefix()（2-127 ページ）	名前空間が意味を持つよう、ノード・タイプにオーバーライドされたノードの接頭辞を戻します。
getText()（2-127 ページ）	この要素に含まれている、マークアップされていないテキストを戻します。
hasChildNodes()（2-128 ページ）	ノードに子ノードがあるかどうかを判別します。
insertBefore()（2-128 ページ）	既存の子ノードの前に、子ノードを挿入します。
normalize()（2-129 ページ）	このノードの下位サブツリー全体にあるテキスト・ノードを、属性ノードも含めて、すべて正規形に変換します。この正規形では、構造のみがテキスト・ノードを分割します。

表 2-20 XMLNSNode のメソッドの概要（続き）

メソッド	説明
removeChild()（2-129 ページ）	oldChild が示す子ノードを子リストから削除し、それを戻します。
replaceChild()（2-129 ページ）	子リスト内の子ノード oldChild を newChild と置換します。
setPrefix()（2-130 ページ）	名前空間が意味を持つよう、ノード・タイプにオーバーライドされたノードの接頭辞を設定します。

XMLNSNode()

新しい XMLNSNode を構成します。

構文

```
protected  XMLNSNode( String tag);
```

パラメータ	説明
tag	ノードの名前

addText()

このノードにテキストを追加します。最後の子ノードがテキスト・ノードの場合は、最後の子ノードに追加します。テキストがこのノードに追加できない場合、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	文字配列の開始索引
length	追加する配列数
str	追加するテキスト

appendChild()

ノードの子リストの最後に newChild ノードを追加し、戻します。これは、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。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR: このノードが newChild ノード・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クラスの 1 つである場合に発生します。
- WRONG_DOCUMENT_ERR: この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 newChild が作成された場合に発生します。
- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。

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。これがドキュメント・フラグメント・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、このノードの子リストに移動されます。

getChildNodes()

ノード・リストにある、このノードのすべての子を戻します。子ノードが存在しない場合は、ノードを含まない NodeList が戻されます。戻された NodeList の内容は、たとえば、それが作成される基になったノード・オブジェクトの子ノードに変更があった場合、その変更が NodeList アクセッサが戻すノードにすぐに反映されるという意味で最新のものであり、ノード内容の静的スナップショットではありません。このことは、getElementsByTagName メソッドが戻す NodeList を含む、すべての NodeList についていえます。

構文

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

getFirstChild()

ノードの最初の子ノードを戻します。該当するノードがない場合は、null を戻します。

構文

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

getLastChild()

ノードの最後の子ノードを戻します。該当するノードがない場合は、null を戻します。

構文

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

getLocalName()

名前空間が意味を持つよう、ノード・タイプに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()

名前空間が意味を持つよう、ノード・タイプに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()

ノード・タイプに応じて、ノードの名前を戻します。

構文

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

getPrefix()

名前空間が意味を持つよう、ノード・タイプに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()

この要素に含まれている、マークアップされていないテキストを戻します。テキスト要素に関しては、これは生データです。子ノードを持つ要素に関して、このメソッドはサブツリー全体を走査し、各ターミナル・テキスト要素にテキストを追加して、サブツリーに対し XML マークアップを効率的に削除します。たとえば、XML 文書に「William Shakespeare」が含まれている場合、`XMLDocument.getText` とすると、「William Shakespeare」が戻されます。

構文

```
public String getText();
```

hasChildNodes()

ノードに子ノードがあるかどうかを判別します。このノードが子を持つ場合に true を戻します。それ以外の場合は false を戻します。

構文

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

insertBefore()

newChild ノードを既存の子ノード refChild の前に挿入し、このノードを挿入された状態で戻します。refChild が null の場合は、子リストの最後に 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。これは、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。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR: このノードが newChild ノード・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このノードの祖先クラスの 1 つである場合に発生します。
- WRONG_DOCUMENT_ERR: この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 newChild が作成された場合に発生します。
- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
- NOT_FOUND_ERR: refChild がこのノードの子でない場合に発生します。

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	子リストに挿入する新しいノード
refChild	参照ノードであり、このノードの前に新しいノードが挿入されます。

normalize()

このノードの下位サブツリー全体にテキスト・ノードを、属性ノードも含めて、すべて正規形に変換します。この正規形では、構造（要素、コメント、処理命令、CDATA セクション、実体参照など）のみがテキスト・ノードを分割し、隣接するテキスト・ノードまたは空のテキスト・ノードがない状態になります。これを使用して、ドキュメントの DOM 表示が、保存および再ロードされた場合と同様であることを確認できます。また、特定のドキュメント・ツリー構造に依存する操作（X ポインタ検索など）を使用する場合に有効です。

構文

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

removeChild()

oldChild が示す子ノードを子リストから削除し、それを戻します。次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
- NOT_FOUND_ERR: oldChild がこのノードの子でない場合に発生します。

構文

```
public org.w3c.dom.Node removeChild( org.w3c.dom.Node oldChild);
```

パラメータ	説明
oldChild	削除されるノード

replaceChild()

子のノード・リスト内の子ノード oldChild を newChild と置換し、oldChild ノードを戻します。これは、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。次の DOMException が発生します。

- HIERARCHY_REQUEST_ERR: このノードが newChild ノード・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このノードの祖先クラスの 1 つである場合に発生します。
- WRONG_DOCUMENT_ERR: この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 newChild が作成された場合に発生します。
- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
- NOT_FOUND_ERR: oldChild がこのノードの子でない場合に発生します。

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()

名前空間が意味を持つよう、ノード・タイプにオーバーライドされた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

出力ストリームを書込み、XML エンコーディングを処理します。

構文

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

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はありません。
DEFAULT	public static int DEFAULT	追加のインデントおよび新しい行はありません。
PRETTY	public static int PRETTY	読みやすくするため、インデントと新しい行を追加します。

表 2-22 XMLOutputStream のメソッドの概要

メソッド	説明
XMLOutputStream()（2-132 ページ）	ASCII 出力を作成します。
addIndent()（2-132 ページ）	出力のためインデント・レベルを設定します。
close()（2-132 ページ）	出力ストリームをクローズします。
flush()（2-133 ページ）	出力ストリームをフラッシュします。
getOutputStyle()（2-133 ページ）	現行の出力スタイルを戻します。
setEncoding()（2-133 ページ）	出力用の文字コードを設定します。
setOutputStyle()（2-134 ページ）	出力スタイルを設定します。
write()（2-134 ページ）	出力ストリームの型に従って、文字を出力します。
writeChars()（2-134 ページ）	出力する文字列を書き込みます。
writeIndent()（2-135 ページ）	インデントを書き込みます。
writeNewLine()（2-135 ページ）	新しい行を書き込みます。
writeQuotedString()（2-135 ページ）	引用句で囲まれた文字列を書き込みます。

XMLOutputStream()

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);`	出力ストリームを使用して出力を作成します。
`public XMLOutputStream( java.io.PrintWriter out);`	プリント・ライターを使用して出力を作成します。

パラメータ	説明
`out`	出力

addIndent()

出力のためインデント・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()

出力ストリームをクローズします。エラーが起きた場合、`IOException` が発生します。

構文

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

flush()

出力ストリームをフラッシュします。エラーが起きた場合、IOException が発生します。

構文

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

getOutputStyle()

現行の出力スタイルを戻します。

構文

```
public int getOutputStyle();
```

setEncoding()

出力用の文字コードを設定します。エンコーディング・タイプの設定でエラーが起きた場合、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	エンコーディングがリトルエンディアン・タイプである場合に、それを示すフラグ
byteOrderMark	バイト・オーダー・マークが設定されている場合に、それを示すフラグ

setOutputStyle()

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()

出力ストリームの型に従って、文字を出力します。文字の書き込み中にエラーが起きた場合、IOException が発生します。

構文

```
public void write( int c);
```

パラメータ	説明
c	書き込まれた文字

writeChars()

出力する文字列を書き込みます。文字列の書き込み中にエラーが起きた場合、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まれた文字列

writeIndent()

インデントを書き込みます。文字列の書き込み中にエラーが起きた場合、IOException が発生します。

構文

```
public void writeIndent();
```

writeNewLine()

新しい行を書き込みます。文字列の書き込み中にエラーが起きた場合、IOException が発生します。

構文

```
public void writeNewLine();
```

writeQuotedString()

引用句で囲まれた文字列を書き込みます。文字列の書き込み中にエラーが起きた場合、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まれた文字列

XMLPI クラス

DOM 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。

構文

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

表 2-23 XMLPI のメソッドの概要

メソッド	説明
XMLPI()（2-136 ページ）	XMLPI の新しいインスタンスを作成します。
addText()（2-137 ページ）	ノードにテキスト文字列を追加し、更新されたノードを戻します。
getNodeName()（2-137 ページ）	PI ノードの名前を戻します。
getNodeType()（2-137 ページ）	基礎となるオブジェクトのノードの型を戻します。
getTarget()（2-137 ページ）	この処理命令のターゲットを戻します。
readExternal()（2-138 ページ）	writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
reportSAXEvents()（2-138 ページ）	DOM ツリーから SAX イベントを通知します。
writeExternal()（2-138 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

XMLPI()

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、この DOM ノードがデシリアライズ / 解凍中の場合にのみ使用できることに注意してください。シリアライズ / 圧縮ストリームから DOM ノードを構成するためにこのノードをデシリアライズするには、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()

ノードにテキスト文字列を追加し、更新されたノードを戻します。

構文

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

パラメータ	説明
str	追加されるテキスト文字列

getNodeName()

PI ノードの名前を戻します。

構文

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

getNodeType()

基礎となるオブジェクトのノードの型を戻します。

構文

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

getTarget()

この処理命令のターゲットを戻します。XML は、これを処理命令を開始するマークアップの後の最初のトークンとして定義します。

構文

```
public String getTarget();
```

readExternal()

writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従っ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み取り中にエラーが起きた場合に発生します。
- ClassNotFoundException - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

reportSAXEvents()

DOM ツリーから SAX イベントを通知します。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()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、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される ObjectOutput ストリーム

XMLPrintDriver クラス

PrintDriver インタフェースを実装します。

構文

```
public class XMLPrintDriver extends Object implements
oracle.xml.parser.v2.PrintDriver
```

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

表 2-25 XMLPrintDriver のメソッドの概要

メソッド	説明
XMLPrintDriver()（2-140 ページ）	XMLPrintDriver のインスタンスを作成します。
close()（2-140 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()（2-141 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()（2-141 ページ）	XMLAttr ノードを出力します。
printAttributeNodes()（2-141 ページ）	XMLElement の各属性のプリント・メソッドをコールします。
printCDATASection()（2-142 ページ）	XMLCDATA ノードを出力します。
printChildNodes()（2-142 ページ）	XMLNode の子のそれぞれにプリント・メソッドをコールします。
printComment()（2-142 ページ）	XML コメント・ノードを出力します。
printDoctype()（2-143 ページ）	DTD を出力します。
printDocument()（2-143 ページ）	XMLDocument を出力します。
printDocumentFragment()（2-143 ページ）	空の XMLDocumentFragment オブジェクトを出力します。
printElement()（2-144 ページ）	XMLElement を出力します。
printEntityReference()（2-144 ページ）	XMLEntityReference ノードを出力します。

表 2-25 XMLPrintDriver のメソッドの概要（続き）

メソッド	説明
printProcessingInstruction()（2-144 ページ）	XMLPI ノードを出力します。
printTextNode()（2-145 ページ）	XMLText ノードを出力します。
setEncoding()（2-145 ページ）	プリント・ドライバのエンコーディングを設定します。

XMLPrintDriver()

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	出力ストリーム
pw	プリント・ライター

close()

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()

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()

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()

XMLElement の各属性のプリント・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	出力される属性を持つ要素

printCDATASection()

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()

XMLNode の子のそれぞれにプリント・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	出力される子を持つノード

printComment()

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()

DTD を出力します。

構文

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

パラメータ	説明
dtd	出力される DTD

printDocument()

XMLDocument を出力します。

構文

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

パラメータ	説明
doc	出力されるドキュメント

printDocumentFragment()

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	出力されるドキュメント・フラグメント

printElement()

XMLElement を出力します。

構文

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

パラメータ	説明
elem	出力される要素

printEntityReference()

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()

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()

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	テキスト・ノード

setEncoding()

プリント・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	出力されるドキュメントのエンコーディング

XMLRangeException クラス

RangeException をカスタマイズします。

構文

```
public class XMLRangeException
```

XMLRangeException()

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	例外のコード

XMLText クラス

DOM 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。

構文

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

表 2-26 XMLText のメソッドの概要

メソッド	説明
XMLText()（2-148 ページ）	XMLText のインスタンスを作成します。
addText()（2-148 ページ）	テキストをテキスト・ノードのデータに追加します。
getData()（2-149 ページ）	この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()（2-149 ページ）	XMLText ノードの名前を戻します。
getNodeType()（2-149 ページ）	基礎となるオブジェクトの型を表すノードの型を戻します。
getNodeValue()（2-150 ページ）	このテキスト・ノードの文字列値を戻します。
isWhiteSpaceNode()（2-150 ページ）	テキスト・ノードが空白ノードかどうかをチェックします。
readExternal()（2-150 ページ）	writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。
reportSAXEvents()（2-151 ページ）	DOM ツリーから SAX イベントを通知します。
splitText()（2-151 ページ）	テキスト・ノードを、指定したオフセットで 2 つのテキスト・ノードに分割します。
writeExternal()（2-152 ページ）	このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、オブジェクトの状態を保存します。

XMLText()

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、この DOM ノードがデシリアライズ / 解凍中の場合にのみ使用できることに注意してください。シリアライズ / 圧縮ストリームから DOM ノードを構成するためにこのノードをデシリアライズするには、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()

appendData と同様に、テキストをテキスト・ノードのデータに追加します。

構文

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

パラメータ	説明
ch	追加される文字配列
start	開始索引
length	文字配列の長さ

getData()

この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、テキスト・ノードに格納できるデータ量を制限することはできません。ただし、実装制限によって、ノードのデータ全体を単一の DOM 文字列に格納できない場合があります。このような場合、substringData をコールすると、データを適切なピース・サイズで取得できます。

次の DOMException が発生します。

- NO_MODIFICATION_ALLOWED_ERR: ノードが読取り専用である場合に発生します。
- DOMSTRING_SIZE_ERR: ノードが実装プラットフォーム上で DOMString 変数より多い文字を戻す場合に発生します。

構文

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

getNodeName()

XMLText ノードの名前を戻します。

構文

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

getNodeType()

基礎となるオブジェクトの型を表すノードの型を戻します。

構文

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

getNodeValue()

このテキスト・ノードの文字列値を戻します。値の取得中にエラーが起きた場合、DOMException が発生します。

構文

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

isWhiteSpaceNode()

テキスト・ノードが空白ノードかどうかをチェックします。

構文

```
public boolean isWhiteSpaceNode();
```

readExternal()

writeExternal メソッドにより圧縮ストリームで書き込まれた情報を読み取り、それに従ってオブジェクトをリストアします。XMLText オブジェクトが独立ノードとしてデシリアライズ（または読み取り）され、他の DOM ノードからコールされていない場合、このメソッドがコールされます。次の例外が発生します。

- IOException - 入力ストリームの読み取り中にエラーが起きた場合に発生します。
- ClassNotFoundException - クラスが見つからない場合に発生します。

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取りに使用される ObjectInput ストリーム

reportSAXEvents()

DOM ツリーから SAX イベントを通知します。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()

テキスト・ノードを、指定したオフセットで 2 つのテキスト・ノードに分割します。このため、これら 2 つのノードは兄弟関係となり、1 つ目のノードにはオフセットまでの内容のみが含まれます。新しいテキスト・ノードを戻します。新しく挿入される 2 つ目のノードには、オフセット点の後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: 指定したオフセットが負数の場合、または data 内の文字数より多い場合に発生します。
- NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。

構文

```
public org.w3c.dom.Text splitText( int offset);
```

パラメータ	説明
offset	分割点となるオフセット（0 から開始）

writeExternal()

このオブジェクトに関する情報を持つバイナリ圧縮ストリームを作成することで、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	圧縮ストリームの書込みに使用される ObjectOutput ストリーム

3

Java 用の XML 処理（JAXP）

この章では、oracle.xml.parser.v2 パッケージの JAXP API について説明します。

- JXDocumentBuilder クラス
- JXDocumentBuilderFactory クラス
- JXSAXParser クラス
- JXSAXParserFactory クラス
- JXSAXTransformerFactory クラス
- JXTransformer クラス

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

JXDocumentBuilder クラス

JXDocumentBuilder は、XML 文書から DOM ドキュメント・インスタンスを取得するための API を定義します。アプリケーション・プログラマは、このクラスを使用して `org.w3c.dom.Document` を XML から取得できます。

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、XML を様々な入力ソースから解析できます。入力ソースとは、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 から様々なクラスを再利用します。このため、基になる 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` に解析する必要がありません。実装がこれらの既存 API を使用してアプリケーションと通信するだけですみます。

構文

```
public class JXDocumentBuilder
```

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）	このパーサーが名前空間を理解するかどうかを示します。
isValidating()（3-3 ページ）	このパーサーが XML 文書を解析するかどうかを示します。
newDocument()（3-3 ページ）	DOM ツリーの構築に使用する DOM ドキュメント・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）	与えられた入力ソースの内容を XML 文書として解析し、新しい DOM ドキュメント・オブジェクトを戻します。
setEntityResolver()（3-4 ページ）	解析する XML 文書中に存在するエンティティを解決する EntityResolver を指定します。
setErrorHandler()（3-5 ページ）	解析する XML 文書中に存在するエンティティの解決に使用する `ErrorHandler` を指定します。

getDOMImplementation()

この文書を処理する DOM インプリメンテーション・オブジェクトを戻します。DOM アプリケーションは、複数実装からオブジェクトを使用できます。

構文

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

isNamespaceAware()

このパーサーが名前空間を理解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()

このパーサーが XML 文書を解析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 isValidating();
```

newDocument()

DOM ツリーの構築に使用する DOM ドキュメント・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()

与えられた入力ソースの内容を XML 文書として解析し、新しい DOM ドキュメント・オブジェクトを戻します。次の例外が発生します。

- IOException - I/O エラーが起きた場合に発生します。
- SAXException - 解析エラーが起きた場合に発生します。
- IllegalArgumentException - 入力ソースが NULL の場合に発生します。

構文

```
public org.w3c.dom.Document parse( org.xml.sax.InputSource is);
```

パラメータ	説明
is	解析する内容を含む入力ソース

setEntityResolver()

解析する XML 文書中に存在するエンティティを解決する EntityResolver を指定します。これを NULL に設定すると、基になる実装はデフォルトの実装と動作を使用します。

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()

解析する XML 文書中に存在するエンティティの解決に使用する ErrorHandler を指定します。これを null に設定すると、基になる実装はデフォルトの実装と動作を使用します。

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

ファクトリ API を定義します。この API によりアプリケーションは、XML 文書から DOM オブジェクト・ツリーを作成するパーサーを得ることができます。

構文

```
public class JXDocumentBuilderFactory
```

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で使用するベース URL。
DEBUG_MODE	public static final java.lang.String DEBUG_MODE	デバッグ・モード : Boolean.TRUE または FALSE。
DTD_OBJECT	public static final java.lang.String DTD_OBJECT	検証に使用する DTD オブジェクト。
ERROR_ENCODING	public static final java.lang.String ERROR_ENCODING	エラー・ストリーム用のエラー・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・ストリーム : OutputStream または PrintWriter。ErrorHandler が設定されている場合は無視されます。
NODE_FACTORY	public static final java.lang.String NODE_FACTORY	NodeFactory はカスタム Node を作成します。
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。
SHOW_WARNINGS	public static final java.lang.String SHOW_WARNINGS	警告の無視 : Boolean.TRUE または FALSE。
USE_DTD_ONLY_FOR_VALIDATION	public static final java.lang.String USE_DTD_ONLY_FOR_VALIDATION	検証に使用する DTD オブジェクト。パーサー文書には追加されません。

表 3-3 JXDocumentBuilderFactory のメソッドの概要

メソッド	説明
JXDocumentBuilderFactory() (3-7 ページ)	デフォルトのコンストラクタです。
getAttribute() (3-7 ページ)	ユーザーは基になる実装の特定の属性を取得できます。
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作成するよう構成されているかどうかを示します。
isIgnoringComments() (3-8 ページ)	ファクトリが、コメントを無視するパーサーを作成するよう構成されているかどうかを示します。
isNamespaceAware() (3-8 ページ)	ファクトリが、名前空間を認識するパーサーを作成するよう構成されているかどうかを示します。
newDocumentBuilder() (3-8 ページ)	DocumentBuilder の新しいインスタンスを、現在設定済のパラメータを使用して作成します。
setAttribute() (3-9 ページ)	実装に対して特定の属性を設定します。

JXDocumentBuilderFactory()

デフォルトのコンストラクタです。

構文

```
public  JXDocumentBuilderFactory();
```

getAttribute()

属性値を実装から取得します。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	属性の名前

isExpandEntityReferences()

ファクトリが、実体参照ノードを拡張するパーサーを作成するよう構成されているかどうかを示します。常に TRUE を戻します。

構文

```
public boolean isExpandEntityReferences();
```

isIgnoringComments()

ファクトリが、コメントを無視するパーサーを作成するよう構成されているかどうかを示します。常に FALSE を戻します。

構文

```
public boolean isIgnoringComments();
```

isNamespaceAware()

ファクトリが、名前空間を認識するパーサーを作成するよう構成されているかどうかを示します。常に TRUE を戻します。

構文

```
public boolean isNamespaceAware();
```

newDocumentBuilder()

現在設定済のパラメータから DocumentBuilder の新しいインスタンスを作成します。失敗すると、ParserConfigurationException が発生します。

構文

```
public DocumentBuilder newDocumentBuilder();
```

setAttribute()

実装に対して特定の属性を設定します。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`	属性の名前
`value`	属性の値

JXSAXParser クラス

org.xml.sax.XMLReader 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 に置き換わりました。

移行を容易にするため、このクラスは同じ名前とインタフェースを引き続きサポートしながら新しいメソッドに対応しています。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、XML を様々な入力ソースから解析できます。入力ソースとは、入力ストリーム、ファイル、URL および SAX 入力ソースです。

この静的メソッドは、新しいファクトリ・インスタンスをシステム・プロパティ設定に基づいて作成し、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。

どのファクトリ実装を作成するかを制御するシステム・プロパティの名前は、「javax.xml.style.TransformFactory」です。このプロパティ名が示すクラスは、この抽象クラスの具象サブクラスです。プロパティが定義されていない場合、プラットフォームのデフォルトが使用されます。内容が基になるパーサーによって解析されると、与えられた HandlerBase のメソッドがコールされます。

構文

```
public class JXSAXParser
```

表 3-4 JXSAXParser のメソッドの概要

メソッド	説明
getProperty()（3-11 ページ）	XMLReader の基になる実装内でリクエストされたプロパティの値を戻します。
getXMLReader()（3-11 ページ）	このクラスの実装によってカプセル化されている XMLReader を戻します。
isNamespaceAware()（3-11 ページ）	このパーサーが名前空間を理解するように構成されているかどうかを示します。
isValidating()（3-12 ページ）	このパーサーが XML 文書を検証するように構成されているかどうかを示します。
setProperty()（3-12 ページ）	XMLReader の基になる実装で特定のプロパティを設定します。

getProperty()

org.xml.sax.XMLReader の基になる実装内で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 - 基になる XMLReader がプロパティ名を認識しない場合に発生します。
- SAXNotSupportedException - 基になる XMLReader がプロパティ名を認識するが、このプロパティをサポートしない場合に発生します。

構文

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

パラメータ	説明
name	取得されるプロパティの名前

getXMLReader()

このクラスの実装によってカプセル化されている XMLReader を戻します。

構文

```
public XMLReader getXMLReader();
```

isNamespaceAware()

このパーサーが名前空間を理解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理解する場合は TRUE を戻し、理解しない場合は FALSE を戻します。

構文

```
public boolean isNamespaceAware();
```

isValidating()

このパーサーが XML 文書を解析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 isValidating();
```

setProperty()

XMLReader の基になる実装で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 - 基になる XMLReader がプロパティ名を認識しない場合に発生します。
- SAXNotSupportedException - 基になる XMLReader がプロパティ名を認識するが、このプロパティをサポートしない場合に発生します。

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

ファクトリ API を定義します。この API によりアプリケーションは、SAX ベース・パーサーを構成および取得して XML 文書を解析可能になります。

構文

```
public class JXSAXParserFactory
```

表 3-5 JXSAXParserFactory のメソッドの概要

メソッド	説明
JXSAXParserFactory()（3-13 ページ）	デフォルトのコンストラクタです。
getFeature()（3-14 ページ）	XMLReader の基になる実装内でリクエストされたプロパティの値を戻します。
isNamespaceAware()（3-14 ページ）	ファクトリが、名前空間を認識するパーサーを作成するよう構成されているかどうかを示します。
newSAXParser()（3-14 ページ）	SAXParser の新しいインスタンスを、現在設定済のファクトリ・パラメータを使用して作成します。
setFeature()（3-15 ページ）	XMLReader の基になる実装で特定の機能を設定します。

JXSAXParserFactory()

デフォルトのコンストラクタです。

構文

```
public  JXSAXParserFactory();
```

getFeature()

XMLReader の基になる実装内でリクエストされたプロパティの値を戻します。次の例外が発生します。

- `SAXNotRecognizedException` - 基になる XMLReader がプロパティ名を認識しない場合に発生します。
- `SAXNotSupportedException` - 基になる XMLReader がプロパティ名を認識するが、このプロパティをサポートしない場合に発生します。

構文

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

パラメータ	説明
name	取り出されるプロパティの名前

isNamespaceAware()

ファクトリが、名前空間を認識するパーサーを作成するよう構成されているかどうかを示します。

構文

```
public boolean isNamespaceAware();
```

newSAXParser()

SAXParser の新しいインスタンスを、現在設定済のファクトリ・パラメータを使用して作成します。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に、`ParserConfigurationException` が発生します。

構文

```
public SAXParser newSAXParser();
```

setFeature()

XMLReader の基になる実装で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 - 基になる XMLReader がプロパティ名を認識しない場合に発生します。
- SAXNotSupportedException - 基になる XMLReader がプロパティ名を認識するが、このプロパティをサポートしない場合に発生します。

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 インスタンスを使用して、Transformer オブジェクトと Templates オブジェクトを作成できます。

どのファクトリ実装を作成するかを決定するシステム・プロパティの名前は、javax.xml.transform.TransformerFactory です。このプロパティ名は、JXSAXTransformerFactory のような TransformerFactory の具象サブクラスを示します。プロパティが定義されていない場合、プラットフォームのデフォルトが使用されます。

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。2 種類の ContentHandler があり、1 つは Transformers オブジェクトの作成、もう 1 つは Templates オブジェクトの作成に使用します。

アプリケーションが変換時に使用する XMLReader のために ErrorHandler または EntityResolver の設定を必要とする場合は、URIResolver を使用して、（getXMLReader により）XMLReader への参照を提供する SAXSource を戻す必要があります。

構文

```
public class JXSAXTransformerFactory
```

表 3-6 JXSAXTransformerFactory のメソッドの概要

メソッド	説明
JXSAXTransformerFactory()（3-17 ページ）	デフォルトのコンストラクタです。
getAssociatedStylesheet()（3-17 ページ）	対応するスタイルシート仕様を、XML スタイルシート処理命令により取得します。
getAttribute()（3-18 ページ）	ユーザーは基になる実装の特定の属性を取得できます。
getErrorListener()（3-18 ページ）	TransformerFactory の現行のエラー・イベント・ハンドラ（NULL にはならない）を取得します。
getFeature()（3-19 ページ）	機能の値を検索します。
getURIResolver()（3-19 ページ）	URI を解決する変換時にデフォルトで使用するオブジェクトを取得します。
newTemplates()（3-19 ページ）	Source を処理して、ソースのコンパイル表現である Templates オブジェクトにします。
newTemplatesHandler()（3-20 ページ）	SAX ContentHandler イベントを Templates オブジェクトに処理できる TemplatesHandler オブジェクトを取得します。
newTransformer()（3-20 ページ）	新しい Transformer オブジェクトを生成します。

表 3-6 JXSAXTransformerFactory のメソッドの概要（続き）

メソッド	説明
newTransformerHandler()（3-21 ページ）	TransformerHandler オブジェクトを生成します。
newXMLFilter()（3-22 ページ）	XMLFilter を作成します。
setAttribute()（3-22 ページ）	基になる実装上に特定の属性を設定します。
setErrorListener()（3-23 ページ）	TransformerFactory のイベント・リスナーを設定します。これは変換命令の処理に使用します。変換そのものには使用しません。
setURIResolver()（3-23 ページ）	xsl:import や xsl:include で使用する URI を解決する変換時にデフォルトで使用するオブジェクトを設定します。

JXSAXTransformerFactory()

デフォルトのコンストラクタです。

構文

```
public  JXSAXTransformerFactory();
```

getAssociatedStylesheet()

対応するスタイルシート仕様を、ソース・パラメータおよびその他パラメータで指定されるドキュメントに一致する XML スタイルシート処理命令（http://www.w3.org/TR/xml-stylesheet/ を参照）により取得します。解析に適した Source オブジェクトを TransformerFactory へ戻します。複数のスタイルシートを戻すことが可能です。この場合、これらのスタイルシートは、インポートのリストまたは 1 つのスタイルシート内のカスケードであるかのように適用されます。

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	一致するメディア属性。NULL にでき、その場合は優先テンプレートが使用されます。(代替 = No)。
title	一致するタイトル属性の値。NULL にできます。
charset	一致する文字セット属性の値。NULL にできます。

getAttribute()

ユーザーは基になる実装の特定の属性を取得できます。属性の値を戻します。基になる実装が属性を認識しない場合に、IllegalArgumentException が発生します。

構文

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

パラメータ	説明
name	属性の名前

getErrorListener()

TransformerFactory の現行のエラー・イベント・ハンドラ（NULL にはならない）を取得します。

構文

```
public ErrorListener getErrorListener();
```

getFeature()

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。あらゆる絶対 URI が機能名になります。

構文

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

パラメータ	説明
name	機能名。絶対 URI です。

getURIResolver()

document()、xsl:import()、または xsl:include() で使用される URI を解決する変換時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 で設定された URIResolver を戻します。

構文

```
public URIResolver getURIResolver();
```

newTemplates()

Source を処理して、ソースのコンパイル表現である Templates オブジェクトにします。変換に使用可能な Templates オブジェクトを戻します。NULL にはできません。この Templates オブジェクトは、複数スレッドで同時に使用可能です。Templates オブジェクトを作成することにより、TransformerFactory は、ランタイム変換に影響を及ぼすことなく変換命令のパフォーマンス最適化を詳細に行うことができます。このメソッドが解析中に Templates オブジェクトの作成に失敗すると、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()

SAX ContentHandler イベントを Templates オブジェクトに処理できる TemplatesHandler オブジェクトを取得します。SAX 解析イベントの ContentHandler として使用される TransformerHandler への NULL でない参照を戻します。TemplatesHandler を作成できない場合は、TransformerConfigurationException を発生します。

構文

```
public TemplatesHandler newTemplatesHandler();
```

newTransformer()

新しい Transformer オブジェクトを生成します。単一スレッドで変換実行に使用可能な Transformer オブジェクトを戻します。NULL にはできません。このオブジェクトを複数スレッドで同時に実行しないよう注意し、かわりに異なる TransformerFactory をそれぞれのスレッドで同時に使用するようにしてください。解析中に Templates オブジェクトの作成に失敗すると、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();`	ソースを結果へコピーする Transformer オブジェクトを新規作成します。
`public Transformer newTransformer( Source source);`	Source を処理して Transformer オブジェクトにします。

パラメータ	説明
source	URI、入力ストリームなどを保持するオブジェクト

newTransformerHandler()

TransformerHandler オブジェクトを生成します。SAX 解析イベントを変換できる TransformerHandler への NULL でない参照を戻します。変換は、たとえば一連の SAX 解析イベントを DOM ツリーにコピーする変換の本体（またはコピー）として定義されています。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 イベントを結果へ処理できる TransformerHandler オブジェクトを生成します。
`public TransformerHandler newTransformerHandler( Source source);`	source 引数が指定する変換命令を使用します。
`public TransformerHandler newTransformerHandler( Templates templates);`	templates 引数が指定する変換命令を使用します。

パラメータ	説明
source	変換命令のソース
templates	コンパイルされた変換命令

newXMLFilter()

XMLFilter を作成します。XMLFilter オブジェクトを戻します（この機能がサポートされていない場合は NULL）。TemplatesHandler を作成できない場合は、TransformerConfigurationException が発生します。

構文	説明
`public XMLFilter newXMLFilter( Source source);`	与えら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);`	与えら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	コンパイルされた変換命令

setAttribute()

基になる実装上に特定の属性を設定します。このコンテキストにおける属性は、この実装が提供するオプションになるよう定義されています。

構文

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

パラメータ	説明
name	属性の名前
value	属性の値

setErrorListener()

TransformerFactory のエラー・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しますが、変換そのものには使用しません。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()

xsl:import または xsl:include で使用する URI を解決する変換時に、デフォルトで使用す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

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 メソッドで取得できます。このインスタンスを使用して、様々なソースの XML を処理したり変換出力を様々なシンクに書き込みできます。

このクラスのオブジェクトは、複数スレッドでの同時実行には使用できません。異なる Transformer を、それぞれのスレッドで同時に使用することはできます。Transformer は何度も使用できます。パラメータと出力プロパティは、変換が変わっても維持されます。

構文

```
public class JXTransformer
```

表 3-7 JXTransformer のメソッドの概要

メソッド	説明
JXTransformer()（3-25 ページ）	JXTransformer オブジェクトを構成します。
clearParameters()（3-25 ページ）	setParameter を使用して設定したすべてのパラメータを消去します。
getErrorListener()（3-25 ページ）	変換に有効なエラー・イベント・ハンドラを取得します。
getOutputProperties()（3-26 ページ）	変換の出力プロパティのコピーを取得します。
getOutputProperty()（3-26 ページ）	変換に有効な出力プロパティの文字列値を戻します。
getParameter()（3-27 ページ）	明示的に設定されたパラメータを戻します。
getURIResolver()（3-27 ページ）	URI の解決に使用するオブジェクトを戻します。
setErrorListener()（3-27 ページ）	変換に有効なエラー・イベント・リスナーを設定します。
setOutputProperties()（3-28 ページ）	変換の出力プロパティを設定します。
setOutputProperty()（3-28 ページ）	変換に有効となる出力プロパティを設定します。
setParameter()（3-29 ページ）	変換のパラメータを追加します。
setURIResolver()（3-30 ページ）	document() の URI 解決に使用するオブジェクトを設定します。
transform()（3-30 ページ）	ソース・ツリーを処理して出力結果にします。

JXTransformer()

JXTransformer オブジェクトを構成します。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);`	XSLStylesheet を使用してソースを変換する JXTransformer オブジェクトを構成します。

パラメータ	説明
`templates`	XSLStylesheet またはテンプレート

clearParameters()

setParameter を使用して設定したすべてのパラメータを消去します。

構文

```
public void clearParameters();
```

getErrorListener()

変換に有効なエラー・イベント・ハンドラを取得します。エラー・ハンドラは NULL にできません。

構文

```
public javax.xml.transform.ErrorListener getErrorListener();
```

getOutputProperties()

変換の出力プロパティのコピーを取得します。戻されたプロパティには、ユーザーが設定したプロパティとスタイルシートで設定したプロパティが入っています。これらのプロパティは、XSL Transformations（XSLT）W3C 勧告の第 16 項で指定されるデフォルト・プロパティにより「デフォルト化」されています。ユーザーやスタイルシートにより明確に設定されたプロパティはベース・プロパティ・リスト内にあり、明確に設定されていない XSLT デフォルト・プロパティはデフォルト・プロパティ・リストになります。したがって、`getOutputProperties().getProperty()` は、`setOutputProperty()`、`setOutputProperties()` で設定されたプロパティ、スタイルシート内のプロパティまたはデフォルト・プロパティを取得しますが、`getOutputProperties().get()` は、`setOutputProperty()`、`setOutputProperties()` で明示的に設定されたプロパティかスタイルシート内のプロパティのみを取得します。戻された Properties オブジェクトの変形は、この変換に含まれるプロパティに影響を与えることはありません。認識される引数キーがなく名前空間が修飾されていない場合、プロパティは無視されます。動作は `setOutputProperties()` と直行しません。

構文

```
public java.util.Properties getOutputProperties();
```

getOutputProperty()

変換に有効な出力プロパティの文字列値を戻します（プロパティが見つからなかった場合は NULL）。指定されたプロパティは、setOutputProperty で設定されたプロパティの場合もあれば、スタイルシートで指定されたプロパティの場合もあります。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（修飾名前空間の場合もあります）を指定する null でない文字列

getParameter()

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。与えられた名前のパラメータが見つからなかった場合は NULL を戻します。このメソッドは、デフォルト・パラメータを戻しません。デフォルト・パラメータは、変換処理中にノード・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()

document() などで使われている URI の解決に使用す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を取得します（または null）。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()

変換に有効なエラー・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()

変換の出力プロパティを設定します。これらのプロパティは、テンプレート内に xsl:output で設定したプロパティをオーバーライドします。認識される引数キーがなく名前空間が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、それまでに設定されたプロパティが削除され、この値はテンプレート・オブジェクトで定義した値に戻ります。

修飾されたプロパティ・キー名を 2 つの部分に分かれた文字列で渡します。名前空間 URI は中括弧（{}）で囲まれ、その後にローカル名が続きます。名前の URL が NULL の場合、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「{」文字かどうかを調べれば、null でない URI の有無を安全にチェックできます。

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"/> を使用して定義した要素から取得した場合、修飾名は「{http://xyz.foo.com/yada/baz.html}foo」になります。接頭辞は使用し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	変換に有効なあらゆる同一プロパティのオーバーライドに使用される出力プロパティのセット

setOutputProperty()

変換に有効となる出力プロパティを設定します。プロパティ修飾名を 2 つの部分に分かれた文字列で渡します。名前空間 URI は中括弧（{}）で囲まれ、その後にローカル名が続きます。名前の URL が NULL の場合、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「{」文字かどうかを調べれば、null でない URI の有無を安全にチェックできます。

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"/> を使用して定義した要素から取得した場合、修飾名は「{http://xyz.foo.com/yada/baz.html}foo」になります。接頭辞は使用し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で有効になりません。

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（修飾名前空間の場合もあります）を指定する null でない文字列
value	出力プロパティの null でない文字列値

setParameter()

変換のパラメータを追加します。修飾名を 2 つの部分に分かれた文字列で渡します。名前空間 URI は中括弧（{}）で囲まれ、その後にローカル名が続きます。名前の URL が NULL の場合、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「{」文字かどうかを調べれば、null でない URI の有無を安全にチェックできます。

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"/>` を使用して定義した要素から取得した場合、修飾名は「`{http://xyz.foo.com/yada/baz.html}foo`」になります。接頭辞は使用し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	パラメータの名前。中括弧（{}）で囲んだ名前空間 URI で始められます。
value	値オブジェクト。有効な Java オブジェクトであればいずれも使用できます。適切なオブジェクト強制を提供するか、オブジェクトを単に拡張で使用するために渡すかは、プロセッサに委ねられます。

setURIResolver()

document() で使われる URI の解決に使用するオブジェクトを設定します。リゾルバ引数が NULL の場合、URIResolver 値は消去されてデフォルトの動作が使用されます。現在、document() ファンクション中の 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()

ソース・ツリーを処理して出力結果にします。変換の過程で回復不能なエラーが起きた場合に、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 用の XSLT 処理

XSLT Processor は、XSLT スタイルシートで指定された変換を使用して、XML 文書を他のテキスト形式に変換できます。

この章の内容は、次のとおりです。

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

oraxsl クラス

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。このクラスでは、動作を指定する様々なコマンドライン・オプションを使用できます。

構文

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

表 4-1 oraxsl のコマンドライン・オプション

コマンド	説明
-w	注意メッセージを表示
-e <error log>	エラー・メッセージを出力するファイル
-l <xml file list>	変換する XML ファイルのリスト
-d <directory>	変換する XML ファイルのディレクトリ
-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>	結果ファイルのディレクトリ
-p <param list>	引数リスト
-t <# of threads>	使用するスレッド数
-v	メッセージを表示する

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()

クラス・コンストラクタです。

構文

```
public  oraxsl();
```

main()

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

XPath 処理中に例外が発生したことを示します。

構文

```
public class XPathException extends oracle.xml.parser.v2.XSLException
```

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()（4-4 ページ）	エラー・メッセージを取得します。

getErrorID()

エラー ID を取得します。

構文

```
public int getErrorID();
```

getMessage()

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・メッセージを構成します。クラス `java.lang.Throwable` の `getMessage()` に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た 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・メッセージの取得に使用される XMLError クラス

XSLException クラス

XSL 変換中に例外が発生したことを示します。

構文

```
public class XSLException extends oracle.xml.util.XMLException
```

XSLException()

新しい XSL 例外を生成します。

構文

```
public  XSLException( String mesg);
```

パラメータ	説明
mesg	元のメッセージ

XSLExtensionElement クラス

拡張要素のベース要素です。

構文

```
public class XSLExtensionElement
```

表 4-4 XSLExtensionElement のメソッドの概要

メソッド	説明
XSLExtensionElement()（4-6 ページ）	デフォルトのコンストラクタです。
getAttributeTemplateValue()（4-7 ページ）	テンプレートとしての属性値を取得します。
getAttributeValue()（4-7 ページ）	属性値を取得します。
getChildNodes()（4-7 ページ）	拡張要素の childNodes を取得します。
processAction()（4-8 ページ）	拡張要素の本体を実行します。
processContent()（4-8 ページ）	拡張要素の内容を処理します。

XSLExtensionElement()

デフォルトのコンストラクタです。

構文

```
public  XSLExtensionElement();
```

getAttributeTemplateValue()

テンプレートとしての属性値を取得します。

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	属性の名前

getAttributeValue()

属性の値を取得します。

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	属性の名前

getChildNodes()

ノード・リストとしての拡張要素の childNodes を取得します。

構文

```
protected final java.util.Vector getChildNodes();
```

processAction()

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()

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

事前に構成された 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。変換は、eXtensible Stylesheet Language Transformation（XSLT）1.0 仕様に従って行われます。

構文

```
public class XSLProcessor
```

表 4-5 XSLProcessor のメソッドの概要

メソッド	説明
XSLProcessor()（4-10 ページ）	デフォルトのコンストラクタです。
getParam()（4-10 ページ）	トップレベルのスタイルシート・パラメータの値を取得します。
newXSLStylesheet()（4-10 ページ）	XSLStylesheet を構成します。
processXSL()（4-11 ページ）	入力された XML 文書を変換します。
removeParam()（4-14 ページ）	トップレベルのスタイルシート・パラメータの値を削除します。
resetParams()（4-14 ページ）	すべてのパラメータ・セットをリセットします。
setBaseURL()（4-14 ページ）	href の追加 / インポートを変換するベース URL を設定します。
setEntityResolver()（4-15 ページ）	href の追加 / インポートを変換するエンティティ・リゾルバを設定します。
setErrorStream()（4-15 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()（4-15 ページ）	エラー・レポート用のロケールを設定します。
setParam()（4-16 ページ）	トップレベルのスタイルシート・パラメータの値を設定します。
showWarnings()（4-16 ページ）	XSLOutput オブジェクトのオーバーライドを設定します。

XSLProcessor()

デフォルトのコンストラクタです。

構文

```
public  XSLProcessor();
```

getParam()

トップレベルのスタイルシート・パラメータの値を取得します。

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()

新しい XSLStylesheet を構成して戻します。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);	与えられた InputStream XSL を使用して XSLStylesheet を構成します。
public XSLStylesheet newXSLStylesheet(Reader xsl);	与えられた Reader を使用して XSLStylesheet を構成します。
public XSLStylesheet newXSLStylesheet(java.net.URL xsl);	与えられた URL を使用して XSLStylesheet を構成します。
public XSLStylesheet newXSLStylesheet(XMLDocument xsl);	与えられた XMLDocument を使用して XSLStylesheet を構成します。

パラメータ	説明
xsl	XSL ドキュメント

processXSL()

入力された XML 文書を変換します。エラーが起きた場合に 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);	与えられた InputStream 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);	与えられた Reader 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);	与えられた 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。
public XMLDocumentFragment processXSL(XSLStylesheet xsl, XMLDocument xml);	与えられた 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);	与えられた 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);	与えられた 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 os);	与えられた 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。

構文	説明

`public void processXSL(`
`  XSLStylesheet xsl,`
`  XMLDocumentFragment xml,`
`  printWriter pw);`
	与えられた 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);`
	与えられた XMLDocumentFragment、スタイルシートおよび XMLDocument 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・フラグメントのため、XMLDocumentHandler の次のファンクションはコールされません。- setDocumentLocator, startDocument, endDocument, - setDoctype, endDoctype, setXMLDecl, setTextDecl

`public void processXSL(`
`  XSLStylesheet xsl,`
`  XMLDocument xml,`
`  OutputStream out);`
	与えられた 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。

`public void processXSL(`
`  XSLStylesheet xsl,`
`  XMLDocument xml,`
`  java.io.PrintWriter pw);`
	与えられた 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);`
	与えられた 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 を介して通知されます。XSLT の結果がドキュメント・フラグメントのため、XMLDocumentHandler の次のファンクションはコールされません。- setDocumentLocator, startDocument, endDocument, - setDoctype, endDoctype, setXMLDecl, setTextDecl

`public XMLDocumentFragment processXSL(`
`  XSLStylesheet xsl,`
`  XMLElement inp);`
	与えられた 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。XMLDocumentFragment を戻します。

`public void processXSL(`
`  XSLStylesheet xsl,`
`  XMLElement inp,`
`  org.xml.sax.ContentHandler handler);`
	与えられた 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 ContentHandler を介して通知されます。XSLT の結果がドキュメント・フラグメントのため、ContentHandler の次のファンクションはコールされません。- setDocumentLocator, startDocument, endDocument

構文	説明

`public void processXSL(`
`  XSLStylesheet xsl,`
`  XMLElement xml,`
`  OutputStream out);`
	与えられた 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);`
	与えられた 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。

`public void processXSL(`
`  XSLStylesheet xsl,`
`  XMLElement xml,`
`  XMLDocumentHandler handlerXML);`
	与えられた 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・フラグメントのため、XMLDocumentHandler の次のファンクションはコールされません。- setDocumentLocator, startDocument, endDocument, - setDoctype, endDoctype, setXMLDecl, setTextDecl

パラメータ	説明
`xsl`	変換に使用する XSLStylesheet
`xml`	変換する XML 文書
`ref`	入力 XML ファイル内の外部エンティティを変換する参照 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()

トップレベルのスタイルシート・パラメータの値を削除します。エラーが起きた場合に 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()

すべてのパラメータ・セットをリセットします。エラーが起きた場合に XSLException が発生します。

構文

```
public void resetParams();
```

setBaseURL()

href の追加 / インポートを変換するベース URL を設定します。 ベース URL を使用する前にセットが使用されていた場合は、エンティティ・リゾルバです。「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()

href の追加 / インポートを変換するエンティティ・リゾルバを設定します。設定されない場合、ベース URL が（設定されていれば）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()

警告を出力するための出力ストリームを設定します。警告用の出力ストリームが指定されていない場合は、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	エラーおよび警告用の出力ストリーム

setLocale()

アプリケーションは、これを使用することで、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()

トップレベルのスタイルシート・パラメータの値を設定します。パラメータの値は、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引用符で囲む必要があります)。このパラメータ・ファンクションは XSLStylesheet のパラメータ・ファンクションとともに使用できません。XSLProcessor のパラメータ・ファンクションが使用された場合、XSLStylesheet ファンクションを使用しているパラメータは無視されます。エラーが起きた場合に 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。下位互換性のため、文字列は XPath 式としてみなされます。

showWarnings()

警告を出力するかどうかを決定します。

構文

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

パラメータ	説明
flag	警告を表示するかどうかを決定します。デフォルトでは、警告は出力されません。

XSLStylesheet クラス

テンプレート、キー、変数、属性セットなどの XSL スタイルシート情報を保持します。構成済の同じスタイルシートを使用して、複数の XML 文書を変換できます。

構文

```
public class XSLStylesheet
```

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

表 4-7 XSLStylesheet のメソッドの概要

メソッド	説明
getDecimalFormat()（4-18 ページ）	スタイルシートで指定されている小数点書式記号を戻します。
getOutputEncoding()（4-18 ページ）	xsl:output で指定されているエンコーディングの値を戻します。
getOutputMediaType()（4-18 ページ）	xsl:output で指定されているメディア・タイプの値を戻します。
getOutputProperties()（4-18 ページ）	java.util.Properties としての xsl:output に指定されている出力プロパティを戻します。
newTransformer()（4-19 ページ）	変換にこのスタイルシートを使用する JAXP 変換オブジェクトを戻します。
getContextNode()（4-20 ページ）	トップレベルのスタイルシート・パラメータの値を削除します。
getContextPosition()（4-20 ページ）	すべてのパラメータ・セットをリセットします。
getContextPosition()（4-21 ページ）	トップレベルのスタイルシート・パラメータの値を設定します。

getDecimalFormat()

スタイルシートで指定されている小数点書式記号を戻します。

構文

```
public java.text.DecimalFormatSymbols getDecimalFormat( NSName nsname);
```

パラメータ	説明
nsname	XSL 小数点書式からの修飾名

getOutputEncoding()

xsl:output で指定されているエンコーディングの値を戻します。

構文

```
public String getOutputEncoding();
```

getOutputMediaType()

xsl:output で指定されているメディア・タイプの値を戻します。

構文

```
public jString getOutputMediaType();
```

getOutputProperties()

java.util.Properties としての xsl:output に指定されている出力プロパティを戻します。

構文

```
public java.util.Properties getOutputProperties();
```

newTransformer()

変換にこのスタイルシートを使用する JAXP 変換オブジェクトを戻します。

構文

```
public javax.xml.transform.Transformer newTransformer();
```

XSLTContext クラス

Xpath 処理コンテキストのクラスです。

構文

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

表 4-8 XSLTContext のメソッドの概要

メソッド	説明
getContextNode()（4-20 ページ）	現行のコンテキスト・ノードを戻します。
getContextPosition()（4-20 ページ）	現行のコンテキスト・ノード位置を戻します。
getContextSize()（4-21 ページ）	現行のコンテキスト・サイズを戻します。
getError()（4-21 ページ）	エラー・レポート用の XMLError インスタンスを戻します。
getVariable()（4-21 ページ）	与えられたスタック・オフセットでの変数を戻します。
reportCharacters()（4-22 ページ）	現行の出力ハンドラに文字を通知します。
reportNode()（4-22 ページ）	現行の出力ハンドラに XMLNode を通知します。
setError()（4-22 ページ）	XMLError を設定します。

getContextNode()

現行のコンテキスト・ノードを戻します。

構文

```
public XMLNode getContextNode();
```

getContextPosition()

現行のコンテキスト・ノード位置を戻します。

構文

```
public int getContextPosition();
```

getContextSize()

現行のコンテキスト・サイズを戻します。

構文

```
public int getContextSize();
```

getError()

エラー・レポート用の XMLError インスタンスを戻します。

構文

```
public XMLError getError();
```

getVariable()

与えられたスタック・オフセットでの変数を戻します。

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()

現行の出力ハンドラに文字を通知します。

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 XML 1.0 仕様で定義されているキャラクタのエスケープ操作を使用可能または使用不可にするブーリアン

reportNode()

現行の出力ハンドラに XMLNode を通知します。

構文

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

パラメータ	説明
node	出力されるノード

setError()

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

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。

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

XMLSchema クラス

このクラスには、異なるターゲット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。

構文

```
public class XMLSchemaNode extends oracle.xml.parser.schema.XSDNode
```

表 5-1 XML Schema のメソッドの概要

コンストラクタ	説明
XMLSchema()（5-2 ページ）	XMLSchema コンストラクタです。
getAllTargetNS()（5-3 ページ）	スキーマに定義されたすべてのターゲット名前空間を戻します。
getSchemaByTargetNS()（5-3 ページ）	指定された名前空間に対する schemaNode を戻します。
getSchemaTargetNS()（5-3 ページ）	最上位のスキーマのターゲット名前空間を戻します。
getXMLSchemaNodeTable()（5-3 ページ）	XMLSchemaNode 表を戻します。
getXMLSchemaURLS()（5-4 ページ）	XMLSchema の URL を戻します。
printSchema()（5-4 ページ）	スキーマの情報を出力します。

XMLSchema()

XMLSchema 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構成します。

パラメータ	説明
n	schemaNode セットの初期サイズ

getAllTargetNS()

スキーマに定義されたすべてのターゲット名前空間を戻します。

構文

```
public java.lang.String[] getAllTargetNS();
```

getSchemaByTargetNS()

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名前空間

getSchemaTargetNS()

最上位のスキーマのターゲット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()

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()

XMLSchema の URL を配列として戻します。

構文

```
public java.lang.String[] getXMLSchemaURLS();
```

printSchema()

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

このクラスには、ターゲット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。

構文

```
public class XMLSchemaNode extends oracle.xml.parser.schema.XSDNode
```

表 5-2 XMLSchemaNode のメソッドの概要

メソッド	説明
XMLSchemaNode() (5-5 ページ)	XMLSchema コンストラクタです。
getAttributeDeclarations() (5-6 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() (5-6 ページ)	スキーマに存在するすべての最上位の複合型要素を戻します。
getComplexTypeTable() (5-6 ページ)	複合型の定義を戻します。
getElementSet() (5-6 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() (5-6 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() (5-7 ページ)	単純型の定義を戻します。
getTargetNS() (5-7 ページ)	スキーマの targetNS を戻します。
getTypeDefinitionTable() (5-7 ページ)	型定義を戻します。

XMLSchemaNode()

XMLSchema のコンストラクタです。

構文

```
public  XMLSchemaNode();
```

getAttributeDeclarations()

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()

スキーマに存在するすべての最上位の複合型要素を配列として戻します。

構文

```
public XSDNode getComplexTypeSet();
```

getComplexTypeTable()

複合型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()

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()

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()

単純型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()

スキーマの targetNS を戻します。XSDNode.getTargetNS() をオーバーライドします。

構文

```
public String getTargetNS();
```

getTypeDefinitionTable()

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

このクラスは、スキーマ属性の宣言を表します。

構文

```
public class XSDAttribute extends oracle.xml.parser.schema.XSDNode
```

表 5-3 XSDAttribute のメソッドの概要

メソッド	説明
getDefaultVal()（5-8 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()（5-9 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()（5-9 ページ）	ノード名を戻します。
getRefLocalname()（5-9 ページ）	解決済の「ref」属性のローカル名を戻します。
getRefNamespace()（5-9 ページ）	解決済の「ref」属性の名前空間を戻します。
getRefState()（5-10 ページ）	refState を戻します。
getTargetNS()（5-10 ページ）	ターゲット名前空間を戻します。
getType()（5-10 ページ）	ノード型を戻します。
isRequired()（5-10 ページ）	属性が必要であるかどうかを確認します。

getDefaultVal()

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()

要素の場合は、「fixed」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()

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()

解決済の「ref」属性の refLocal 名を戻します。

構文

```
public String getRefLocalname();
```

getRefNamespace()

解決済の「ref」属性の RefNamespace を戻します。

構文

```
public String getRefNamespace();
```

getRefState()

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()

ターゲット名前空間を戻します。

構文

```
public String getTargetNS();
```

getType()

simpleType または complexType のいずれかのノード型を戻します。

構文

```
public XSDNode getType();
```

isRequired()

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

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、最上位スキーマの宣言と定義に対応するオブジェクト・セット（情報セット項目）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。（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。（存在する場合）対応する DOM ツリーに置換します。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。ツリーの最上位要素（情報セット項目）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、カーディナリティ情報（min/maxOccurs）を含む、SNode 型のノード / オブジェクトの後に、下位レベルの要素、グループ宣言および参照に対応するノードを持つツリーです。

構文

```
public class XSDBuilder
```

表 5-4 XSDBuilder のメソッドの概要

メソッド	説明
XSDBuilder()（5-11 ページ）	クラス・コンストラクタです。
build()（5-12 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()（5-13 ページ）	XML Schema オブジェクトを戻します。
setEntityResolver()（5-13 ページ）	インポート / インクルードを解決するための EntityResolver を設定します。
setError()（5-14 ページ）	XMLError オブジェクトを設定します。
setLocale()（5-14 ページ）	エラー・レポート用のロケールを設定します。

XSDBuilder()

XSDBuilder コンストラクタです。

構文

```
public XSDBuilder() throws XSDException;
```

build()

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 が正常に XMLSchema オブジェクトを作成しなかった場合に発生します。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);`	Reader および URL から XMLSchema オブジェクトを作成します。
`public Object build( String sysId);`	システム識別子から 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識別子から 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。
`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識別子から 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	文書内のインポート / インクルードにおいて、相対参照の解決に使用される URL
doc	スキーマ要素を含む XMLdocument
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名前空間
r	スキーマの Reader
schemaDoc	XMLDocument
schemaDocs	XMLDocument の配列
sysId	スキーマの場所
url	スキーマの URL

getObject()

XML Schema オブジェクトを戻します。

構文

```
public Object getObject();
```

setEntityResolver()

インポート / インクルードを解決するための EntityResolver を設定します。
「org.xml.sax.EntityResolver」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	EntityResolver

setError()

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()

エラー・レポート用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	ロケール・オブジェクト

XSDComplexType クラス

このクラスは、スキーマの ComplexType の定義を表します。

構文

```
public class XSDComplexType extends oracle.xml.parser.schema.XSDNode
```

表 5-5 XSDComplexType のメソッドの概要

メソッド	説明
getAttributeDeclarations()（5-15 ページ）	この複合型の属性宣言を戻します。
getAttributeWildcard()（5-16 ページ）	この複合型の属性ワイルドカードを戻します。
getBaseType()（5-16 ページ）	この complextype のベース型を戻します。
getContent()（5-16 ページ）	XSDComplexType の内容を戻します。
getDerivationMethod()（5-16 ページ）	親の型からこの型を派生した方法に関する情報を戻します。
getElementSet()（5-17 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()（5-17 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()（5-17 ページ）	解決済「base」属性のローカル名を戻します。
getTypeGroup()（5-17 ページ）	複合型の型グループを戻します。
init()（5-17 ページ）	複合型を初期化します。
isAbstract()（5-18 ページ）	複合型が抽象型である場合、TRUE を戻します。

getAttributeDeclarations()

この複合型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()

この複合型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()

この complextype のベース型を戻します。

構文

```
public XSDNode getBaseType();
```

getContent()

XSDComplexType の内容を戻します。

構文

```
public int getContent();
```

getDerivationMethod()

親の型からこの型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()

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()

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()

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()

複合型の型グループを戻します。

構文

```
public XSDGroup getTypeGroup();
```

init()

複合型を初期化します。

構文

```
public static void init();
```

isAbstract()

複合型が抽象型である場合、TRUE を戻します。

構文

```
public boolean isAbstract();
```

XSDConstrainingFacet クラス

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。

構文

```
public class XSDConstrainingFacet extends java.lang.Object implements
oracle.xml.parser.schema.XSDTypeConstants
```

表 5-6 XSDConstrainingFacet のメソッドの概要

メソッド	説明
getFacetId()（5-19 ページ）	ファセット ID を戻します。
getLexicalEnumeration()（5-20 ページ）	制約ファセットの LEXICAL 列挙を戻します。
getLexicalValue()（5-20 ページ）	制約ファセットの LEXICAL 値を戻します。
getName()（5-20 ページ）	制約ファセット名を戻します。
isFixed()（5-20 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()（5-21 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()（5-21 ページ）	制約ファセットに対する値を検証します。

getFacetId()

ファセット ID を戻します。

構文

```
public int getFacetId();
```

getLexicalEnumeration()

制約ファセットの LEXICAL 列挙を戻します。

構文

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

getLexicalValue()

制約ファセットの LEXICAL 値を戻します。

構文

```
public String getLexicalValue();
```

getName()

制約ファセット名を戻します。

構文

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

isFixed()

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()

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()

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

このクラスは、スキーマの単純型に対するデータ値を表します。

構文

```
public class XSDDataValue extends java.lang.Object implements
oracle.xml.parser.schema.XSDTypeConstants
```

表 5-7 XSDDataValue のメソッドの概要

メソッド	説明
compareTo()（5-22 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()（5-22 ページ）	STRING/BINARY 値の長さを戻します。
getLexicalValue()（5-23 ページ）	XSDDataValue クラスの LEXICAL 値を戻します。
getPrecision()（5-23 ページ）	小数値の精度を戻します。
getScale()（5-23 ページ）	小数値のスケールを戻します。

compareTo()

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()

STRING/BINARY 値の長さを取得します。データ値がこれらの型でない場合は、XSDException が発生します。

構文

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

getLexicalValue()

XSDDataValue クラスの LEXICAL 値を戻します。

構文

```
public String getLexicalValue();
```

getPrecision()

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()

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 クラスは、スキーマの要素宣言を表します。

構文

```
public class XSDElement
```

表 5-8 XSDElement のメソッドの概要

メソッド	説明
findEquivClass()（5-25 ページ）	このクラスと同等のクラスを検索します。
getDefaultVal()（5-25 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()（5-25 ページ）	解決済同等クラスのローカル名を戻します。
getFixedVal()（5-25 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()（5-26 ページ）	一連の ID を戻します。
getMaxOccurs()（5-26 ページ）	maxOccurs を戻します。
getMinOccurs()（5-26 ページ）	minOccurs を戻します。
getName()（5-26 ページ）	名前を戻します。
getRefLocalname()（5-26 ページ）	解決済の「ref」属性のローカル名を戻します。
getRefNamespace()（5-27 ページ）	解決済の「ref」属性の名前空間を戻します。
getRefState()（5-27 ページ）	refState を戻します。
getSubstitutionGroup()（5-27 ページ）	substitutionGroup を戻します。
getTargetNS()（5-27 ページ）	ターゲット名前空間を設定します。
getType()（5-28 ページ）	ノード型を設定します。
isAbstract()（5-28 ページ）	要素が抽象型の場合は、TRUE を戻します。
isNullable()（5-28 ページ）	要素が NULL 値可能である場合は、TRUE を戻します。
setMaxOccurs()（5-28 ページ）	maxOccurs を設定します。
setMinOccurs()（5-29 ページ）	minOccurs を設定します。

findEquivClass()

この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
rm	名前

getDefaultVal()

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()

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()

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()

一連の ID を配列として戻します。

構文

```
public XSDIdentity getIdentities();
```

getMaxOccurs()

maxOccurs を戻します。

構文

```
public int getMaxOccurs();
```

getMinOccurs()

minOccurs を戻します。

構文

```
public int getMinOccurs();
```

getName()

ノード名を戻します。

構文

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

getRefLocalname()

解決済の「ref」属性のローカル名を戻します。

構文

```
public String getRefLocalname();
```

getRefNamespace()

解決済の「ref」属性の名前空間を戻します。

構文

```
public String getRefNamespace();
```

getRefState()

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()

substitutionGroup を戻します。

構文

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

getTargetNS()

ターゲット名前空間を戻します。

構文

```
public java.lang.String getTargetNS();
```

getType()

simpleType または complexType のいずれかのノード型を戻します。

構文

```
public XSDNode getType();
```

isAbstract()

この要素が抽象型である場合、TRUE を戻します。

構文

```
public boolean isAbstract();
```

isNullable()

この要素が NULL 値可能である場合、TRUE を戻します。

構文

```
public boolean isNullable();
```

setMaxOccurs()

maxOccurs を設定します。

構文

```
public void setMaxOccurs( int max);
```

パラメータ	説明
maxOccurs	値

setMinOccurs()

minOccurs を設定します。

構文

```
public void setMinOccurs( int min);
```

パラメータ	説明
minOccurs	値

XSDException

XMLSchema 検証中に例外が発生したことを示します。

構文

```
public class XSDException extends Exception
```

getMessage()

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た 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 クラスは、スキーマ・グループの定義を表します。

構文

```
public class XSDGroup
```

表 5-9 XSDIdentity のメソッドの概要

メソッド	説明
getMaxOccurs()（5-31 ページ）	maxOccurs を戻します。
getMinOccurs()（5-31 ページ）	minOccurs を戻します。
getNodeVector()（5-32 ページ）	nodeVector に格納されたグループの小要素を戻します。
getOrder()（5-32 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()（5-32 ページ）	maxOccurs を設定します。
setMinOccurs()（5-32 ページ）	minOccurs を設定します。

getMaxOccurs()

maxOccurs を戻します。

構文

```
public int getMaxOccurs();
```

getMinOccurs()

minOccurs を戻します。

構文

```
public int getMinOccurs();
```

getNodeVector()

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()

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()

maxOccurs を設定します。

構文

```
public void setMaxOccurs( int max);
```

パラメータ	説明
maxOccurs	値

setMinOccurs()

minOccurs を設定します。

構文

```
public void setMinOccurs( int min);
```

パラメータ	説明
minOccurs	値

XSDIdentity クラス

スキーマの ID 制約定義 Unique、Key および Keyref を表します。

構文

```
public class XSDIdentity extends oracle.xml.parser.schema.XSDNode
```

表 5-10 XSDIdentity のメソッドの概要

メソッド	説明
getFields()（5-33 ページ）	フィールドを戻します。
getNodeType()（5-33 ページ）	ノード型を戻します。
getRefer()（5-34 ページ）	参照キーを戻します。
getSelector()（5-34 ページ）	セレクタを戻します。

getFields()

フィールドを戻します。

構文

```
public java.lang.String[] getFields();
```

getNodeType()

ノード型を戻します。XSDNode クラスの XSDNode.getNodeType() をオーバーライドします。

構文

```
public int getNodeType();
```

getRefer()

参照キーを戻します。

構文

```
public String getRefer();
```

getSelector()

セレクタを戻します。

構文

```
public String getSelector();
```

XSDNode クラス

大半の 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。

構文

```
public class XSDNode
```

表 5-11 XSDNode のメソッドの概要

メソッド	説明
getName()（5-35 ページ）	ノード名を戻します。
getNamespaceURI()（5-35 ページ）	psv の名前空間 URI を戻します。
getNodeType()（5-36 ページ）	XSDNode の型を戻します。
getTargetNS()（5-36 ページ）	ターゲット名前空間を戻します。
isNodeType()（5-36 ページ）	ノードが指定された型かどうかを確認します。

getName()

ノード名を戻します。

構文

```
public java.lang.String getName();
```

getNamespaceURI()

psv の名前空間 URI を戻します。

構文

```
public String getNamespaceURI()
```

getNodeType()

XSDNode の型を戻します。

構文

```
public int getNodeType();
```

getTargetNS()

ターゲット名前空間を戻します。

構文

```
public java.lang.String getTargetNS();
```

isNodeType()

ノードが指定された型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	確認するノードの型

XSDSimpleType クラス

このクラスは、スキーマの単純型の定義を表します。

構文

```
public class XSDSimpleType implements oracle.xml.parser.schema.XSDTypeConstants
```

表 5-12 XSDSimpleType のメソッドの概要

メソッド	説明
XSDSimpleType()（5-38 ページ）	クラス・コンストラクタです。
derivedFrom()（5-38 ページ）	指定されたベース型から型を派生させます。
getBase()（5-39 ページ）	この型のベース型を戻します。
getBasicType()（5-39 ページ）	この型を派生したベース型を取得します。
getBuiltInDatatypes()（5-39 ページ）	組込みデータ型を取得します。
getFacets()（5-39 ページ）	ファセットを取得します。
getMaxOccurs()（5-40 ページ）	maxOccurs の値を取得します。
getMinOccurs()（5-40 ページ）	minOccurs の値を取得します。
getVariety()（5-40 ページ）	型の種類を取得します。
isAbstract()（5-40 ページ）	この型が抽象型である場合、TRUE を戻します。
setFacet()（5-41 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()（5-41 ページ）	maxOccurs の値を設定します。
setMinOccurs()（5-41 ページ）	minOccurs の値を設定します。
setSource()（5-42 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()（5-42 ページ）	この型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()

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`	この型が派生したプリミティブ型の ID
`tnm`	この型の名前

derivedFrom()

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()

ベース型を取得します。

構文

```
public XSDSimpleType getBase();
```

getBasicType()

この型を派生したベース型を取得します。

構文

```
public int getBasicType();
```

getBuiltInDatatypes()

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();
```

パラメータ	説明
type	組込み型の名前

getFacets()

ファセットを取得します。

構文

```
public XSDConstrainingFacet getFacets();
```

getMaxOccurs()

maxOccurs の値を取得します。

構文

```
public int getMaxOccurs();
```

getMinOccurs()

minOccurs の値を取得します。

構文

```
public int getMinOccurs();
```

getVariety()

型の種類を取得します。

構文

```
public java.lang.String getVariety();
```

isAbstract()

この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()

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()

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()

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()

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()

この型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

このインタフェースは、W3C Schema Processor の定数を定義します。

構文

```
public interface XSDTypeConstants
```

表 5-13 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
_derivedBy	public static final String _derivedBy
_element	public static final String _element

表 5-13 XSDConstantValues で定義された定数（続き）

定数	定義
_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
_namespace	public static final String _namespace

表 5-13 XSDConstantValues で定義された定数（続き）

定数	定義
_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
_simpleContent	public static final String _simpleContent

表 5-13 XSDConstantValues で定義された定数（続き）

定数	定義
_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 5-13 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 5-13 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

表 5-13 XSDConstantValues で定義された定数（続き）

定数	定義
REF_RESOLVED	public static final int REF_RESOLVED
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 は、XMLSchema に対してインスタンス 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 としてイベントの内容に追加され、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に対して一致検索されます。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` のベクトルがパラレル・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` に対して一致検索されます。`endElement()` イベントを介して検出された終了要素は、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。

構文

```
public class XSDValidator
```

表 5-14 XSDValidator のメソッドの概要

メソッド	説明
XSDValidator() （5-51 ページ）	クラス・コンストラクタです。
characters()（5-52 ページ）	要素内の文字データの通知を伝播します。
endElement()（5-52 ページ）	要素の終わりの通知を受け取ります。
setDocumentLocator()（5-53 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()（5-53 ページ）	XMLError オブジェクトを現在のエラーに設定します。
setXMLProperties()（5-53 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()（5-54 ページ）	プロパティを設定します。
startElement()（5-54 ページ）	要素の始まりの通知を受け取ります。

XSDValidator()

XSDValidator のコンストラクタです。

構文

```
public  XSDValidator();
```

characters()

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。「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()

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。

構文

```
public void endElement( String namespaceURI,
                        String localName,
                        String qName);
```

パラメータ	説明
namespaceURI	名前空間 URI（使用不可、または名前空間が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（使用不可の場合は空の文字列）

setDocumentLocator()

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。「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()

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。

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()

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()

プロパティ値を設定し、このプロパティを戻します。プロパティが読取り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()

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。「org.xml.sax.Attributes」および「endElement()」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 URI（要素に名前空間 URI が設定されていないか、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（使用不可の場合は空の文字列）
atts	要素に追加された属性（属性が存在しない場合は空のオブジェクト）

6

Java 用の XML クラス生成

XML Class Generator for Java は、oracle.xml.classgen パッケージに含まれています。

この章では、次のクラスの API について説明します。

- CGDocument クラス
- CGNode クラス
- CGXSDElement クラス
- DTDClassGenerator クラス
- InvalidContentException クラス
- oracg クラス
- SchemaClassGenerator クラス

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

CGDocument クラス

DTD Class Generator によって生成されたクラスのベース・ドキュメント・クラスとして機能します。

構文

```
public abstract class CGDocument extends oracle.xml.classgen.CGNode implements
java.io.Externalizable
```

表 6-1 CGDocument のメソッドの概要

メソッド	説明
CGDocument()（6-2 ページ）	DTD のルート要素のコンストラクタです。
print()（6-3 ページ）	構成された XML 文書を出力します。
readExternal()（6-3 ページ）	圧縮ストリームを読み取って、ルート要素に対応するオブジェクトを作成します。

CGDocument()

DTD のルート要素の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	クラスを生成するために使用される DTD

print()

構成された XML 文書を出力します。ドキュメントの内容が DTD で指定される文法と一致しない場合、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。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);`	構成された 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);`	ユーザー定義エンコーディングを使用して、構成された XML 文書を出力ストリームに出力します。

パラメータ	説明
out	ドキュメントの出力ストリーム
enc	出力ストリームのエンコーディング

readExternal()

圧縮ストリームを読み取って、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・ドキュメントを含む生成されたクラスのインスタンス化に使用します。

構文

```
protected void readExternal( java.io.ObjectInput inArg,
                             oracle.xml.comp.CXMLContext cxmlContext);
```

パラメータ	説明
in	圧縮ストリームを読み取るために渡される 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

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。

構文

```
public abstract class CGNode
```

表 6-2 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リアン

CGNode のメソッド

表 6-3 CGNode のメソッドの概要

メソッド	説明
CGNode()（6-5 ページ）	DOM ツリーの要素のコンストラクタです。
addCDATASection()（6-6 ページ）	要素に CDATA セクションを追加します。
addData()（6-6 ページ）	要素ノードに PCDATA を追加します。
addNode()（6-6 ページ）	要素にノードを子として追加します。
deleteData()（6-7 ページ）	要素ノードから PCDATA を削除します。
getAttribute()（6-7 ページ）	属性値を取得します。
getDocument()（6-7 ページ）	ベース・ドキュメントを取得します。
getData()（6-8 ページ）	要素の PCDATA を取得します。
getDTDNode()（6-8 ページ）	ベース・ドキュメントから静的 DTD を取得します。
getElementNode()（6-8 ページ）	この CGNode に対応する XMLElement ノードを取得します。
getNode()（6-8 ページ）	名前が入力文字列と一致するこのノードに対応する要素の子の 1 つである CGNode を取得します。
readExternal()（6-9 ページ）	圧縮ストリームを読み取り、対応するノードをインスタンス化します。

表 6-3 CGNode のメソッドの概要（続き）

メソッド	説明
setAttribute()（6-9 ページ）	属性値を設定します。
setDocument()（6-10 ページ）	ベース・ドキュメントを設定します。
setElementNode()（6-10 ページ）	この CGNode に対応する XMLElement ノードを設定します。
storeID()（6-10 ページ）	ID 識別子のこの値を格納します。
storeIDREF()（6-11 ページ）	IDREF 識別子のこの値を格納します。
validateContent()（6-11 ページ）	DTD で指定された内容モデルに従って、要素の内容が有効かどうかを検証します。
validEntity()（6-11 ページ）	ENTITY 識別子が有効かどうかを検証します。
validID()（6-12 ページ）	ID 識別子が有効かどうかを検証します。
validNMTOKEN()（6-12 ページ）	NMTOKEN 識別子が有効かどうかを検証します。
writeExternal()（6-12 ページ）	このノードに対応する圧縮ストリームを書き込みます。

CGNode()

DOM ツリーの要素の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()

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、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()

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、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	要素に追加されるテキスト

addNode()

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、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()

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
theNode	要素から削除されるテキスト

getAttribute()

属性値を戻します。

構文

```
protected String getAttribute( String attName);
```

パラメータ	説明
attName	属性の名前

getDocument()

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()

要素の PCDATA を取得します。データが存在しない場合、InvalidContentException が発生します。

構文

```
protected String getData();
```

getDTDNode()

ベース CGDocument から静的 DTD を取得します。

構文

```
protected abstract oracle.xml.parser.v2.DTD getDTDNode();
```

getElementNode()

この 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()

名前が入力文字列と一致するノードに対応する要素の子の 1 つである CGNode を取得します。

構文

```
protected java.lang.Object getNode(String theNode);
```

パラメータ	説明
theNode	戻される CGNode に対応する文字列の名前

readExternal()

圧縮ストリームを読み取り、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 エラーの場合に発生します。
- ClassNotFoundException - クラスをインスタンス化できなかった場合に発生します。

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取りに使用される XMLObjectInput ストリーム
cxmlContext	圧縮ストリームのコンテキスト

setAttribute()

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	属性の名前
value	属性値

setDocument()

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 CGDocument

setElementNode()

この 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()

ID 識別子の値を格納します。IDREF 値で検証できます。

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()

IDREF 識別子の値を格納します。対応する ID で検証できます。

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()

DTD で指定された内容モデルに従って、要素の内容が有効かどうかを検証します。

構文

```
protected void validateContent();
```

validEntity()

ENTITY 識別子が有効かどうかを検証します。 ENTITY が有効な場合は true、無効な場合は false を戻します。

構文

```
protected boolean validEntity( String entity);
```

パラメータ	説明
name	ENTITY 属性の値

validID()

ID 識別子が有効かどうかを検証します。ID が有効な場合は true、無効な場合は 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()

NMTOKEN 識別子が有効かどうかを検証します。NMTOKEN が有効な場合は true、無効な場合は 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()

このノード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

Schema Class Generator によって生成された XML Schema に対応するすべてのクラスのベース・クラスとして機能します。

構文

```
public abstract class CGXSDElement extends java.lang.Object
```

表 6-4 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

表 6-5 CGXSDElement のメソッドの概要

メソッド	説明
CGXSDElement()（6-14 ページ）	デフォルトのコンストラクタです。
addAttribute()（6-14 ページ）	与えられたノードの属性をハッシュテーブルに追加します。
addElement()（6-14 ページ）	要素ノードのローカル要素をその要素に対応するベクトルに追加します。
getAttributes()（6-15 ページ）	属性を属性名および値のハッシュテーブルとして戻します。
getChildElements()（6-15 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()（6-15 ページ）	ノード値を戻します。
print()（6-15 ページ）	要素ノードを出力します。
printAttributes()（6-16 ページ）	属性ノードを出力します。
setNodeValue()（6-16 ページ）	要素のノード値を設定します。

CGXSDElement()

デフォルトのコンストラクタです。

構文

```
public  CGXSDElement();
```

addAttribute()

与えら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	属性名

addElement()

要素ノードのローカル要素をその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	追加するオブジェクト

getAttributes()

属性を属性名および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()

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()

ノード値を戻します。

構文

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

print()

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、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	出力が出力される XMLObjectOutput ストリーム

printAttributes()

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、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	出力が出力される XMLObjectOutput ストリーム
name	属性名
namespace	名前空間

setNodeValue()

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	ノード値

DTDClassGenerator クラス

DTD または DTD に基づく XML ファイルに対応するデータ・バインド・クラスを生成します。

構文

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

表 6-6 DTDClassGenerator のメソッドの概要

メソッド	説明
DTDClassGenerator()（6-17 ページ）	DTDClassGenerator のデフォルトのコンストラクタです。
generate()（6-18 ページ）	doctype 要素を持つ DTD をルートとして全検索し、Java クラスを生成します。
setGenerateComments()（6-18 ページ）	生成されたクラスの Java ドキュメントのコメントを生成するかどうかを決定します。
setJavaPackage()（6-18 ページ）	生成されたクラスのパッケージを設定します。
setOutputDirectory()（6-19 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()（6-19 ページ）	DTD をシリアライズ・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決定します。
setValidationMode()（6-19 ページ）	生成されたクラスが、XML 文書を検証する必要があるかどうかを決定します。

DTDClassGenerator()

DTDClassGenerator のデフォルトのコンストラクタです。

構文

```
public  DTDClassGenerator();
```

generate()

doctype 要素を持つ DTD をルートとして全検索し、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	クラスを生成するために使用される DTD
doctype	ルート要素名

setGenerateComments()

生成されたクラスの Java ドキュメントのコメントを生成するかどうかを決定します。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	Java ドキュメントのコメント生成をオン / オフにするブーリアン・フラグ

setJavaPackage()

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・セットなしです。

構文

```
public void setJavaPackage( java.util.Vector packageName);
```

パラメータ	説明
packageName	パッケージの名前

setOutputDirectory()

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。デフォルト値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()

DTD をシリアライズ・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決定します。生成されたクラスを使用して XML ファイルを作成した場合、DTD をシリアライズ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 のシリアライズ・オブジェクトとしての保存をオン / オフにするブーリアン・フラグ（true）デフォルトは、テキスト・ファイルとして保存します（false）。

setValidationMode()

生成されたクラスが、構成中の XML 文書を検証する必要があるかどうかを決定します。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 文書の検証をオン / オフにするブーリアン・フラグ デフォルトは true です。

InvalidContentException クラス

DTD ClassGenerator および Schema Class Generator によって発生する例外を定義します。

構文

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

InvalidContentException()

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は、例外に関する情報の入力文字列を受け取ります。

パラメータ	説明
s	例外に関する情報を含む文字列

oracg クラス

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。

構文

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

表 6-7 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 ベースの 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・ベースの 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名です。
-package <Package Name>	生成される Java クラスのパッケージ名です。
-comment	生成される Java ソース・コード用のコメントを生成します。

SchemaClassGenerator クラス

XML Schema に対応するクラスを生成します。

構文

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

表 6-8 SchemaClassGenerator のメソッドの概要

メソッド	説明
SchemaClassGenerator()（6-22 ページ）	コンストラクタです。
generate()（6-23 ページ）	最上位の要素、単純型要素、複合型要素に対応する Schema クラスを生成します。
setGenerateComments()（6-23 ページ）	Java ドキュメントのコメントを生成するかどうかを決定します。
setJavaPackage()（6-24 ページ）	1 つの名前空間に対して 1 つのユーザー定義の Java パッケージ名を割り当てます。
setOutputDirectory()（6-24 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()

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は、XML スキーマの説明を含む入力文字列を受け取ります。

パラメータ	説明
`fileName`	入力 XML スキーマ

generate()

各ノードで createSchemaClass() をコールして、最上位の要素、単純型要素、複合型要素に対応する 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
XML	スキーマ・オブジェクト

setGenerateComments()

Java ドキュメントのコメントを生成するかどうかを決定します。デフォルトは true です。

構文

```
public void setGenerateComments( boolean comments)
```

パラメータ	説明
comments	Java ドキュメントのコメント生成をオン / オフにします。デフォルトは true です。

setJavaPackage()

1 つの名前空間に対して 1 つのユーザー定義の Java パッケージ名を割り当てます。スキーマ内に定義される名前空間が問い合され、問合せの数は、ユーザーが指定する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 スキーマ
pkgName	コマンドラインを介して指定されるユーザー定義のパッケージ名を含むベクトル

setOutputDirectory()

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。現在のディレクトリがデフォルトです。

構文

```
public void setOutputDirectory( String dir);
```

パラメータ	説明
dir	出力ディレクトリ

7

XML SQL Utility for Java

XML SQL Utility for Java（XSU）は、SQL 問合せから XML を生成および格納します。

この章では、次の XSU クラスについて説明します。

- OracleXMLQuery クラス
- OracleXMLSave クラス
- OracleXMLSQLException クラス
- OracleXMLSQLNoRowsException クラス

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

OracleXMLQuery クラス

OracleXMLQuery クラスは、SQL 問合せを指定して XML を生成します。

構文

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

表 7-1 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 を生成することを指定します。
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 スキーマを生成することを指定します。
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。

表 7-2 OracleXMLQuery のメソッドの概要

メソッド	説明
OracleXMLQuery()（7-4 ページ）	クラス・コンストラクタです。
close()（7-5 ページ）	Oracle XML エンジンによって作成されたオープン・リソースをクローズします。
getNumRowsProcessed()（7-5 ページ）	処理された行の数を戻します。

表 7-2 OracleXMLQuery のメソッドの概要（続き）

メソッド	説明
getXMLDOM()（7-5 ページ）	データを XML 文書に変換します。
getXMLMetaData()（7-6 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。
getXMLSAX()（7-7 ページ）	XML 文書の DTD または XMLSchema を戻します。
getXMLSchema()（7-7 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。
getXMLString()（7-7 ページ）	指定した問合せに対応する XMLSchema を生成します。
keepObjectOpen()（7-8 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。
removeXSLTParam()（7-8 ページ）	XML データの取得元になるオブジェクトの永続性をオンまたはオフにします。
setCollIdAttrName()（7-9 ページ）	トップレベルのスタイルシート・パラメータの値を削除します。
setDataHeader()（7-9 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。
setDateFormat()（7-10 ページ）	XML データ・ヘッダーを設定します。
setEncoding()（7-10 ページ）	XML 文書に生成日付の書式を設定します。
setErrorTag()（7-10 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。
setException()（7-11 ページ）	XML エラー・ドキュメントを囲むタグを設定します。
setMaxRows()（7-11 ページ）	ユーザーに、XSU で対処される例外の指定を許可します。
setMetaHeader()（7-11 ページ）	XML に変換される行の最大数を設定します。
setRaiseException()（7-12 ページ）	XML メタ・ヘッダーを設定します。
setRaiseNoRowsException()（7-12 ページ）	呼び出された例外を発生させるかどうかを XSU に通知します。
setRowIdAttrName()（7-12 ページ）	生成された XML 文書が空の場合に、`OracleXMLNoRowsException` が発生するかどうかを XSU に通知します。
setRowIdAttrValue()（7-13 ページ）	行の囲みタグの ID 属性名を設定します。
setRowsetTag()（7-13 ページ）	行の囲みタグの ID 属性に値が割り当てられるスカラー列を指定します。
setRowTag()（7-13 ページ）	XML データセットを囲むタグを設定します。

表 7-2 OracleXMLQuery のメソッドの概要（続き）

メソッド	説明
setSkipRows()（7-14 ページ）	スキップする行数を設定します。
setSQLToXMLNameEscaping()（7-14 ページ）	マップされた SQL オブジェクト名が有効な XML 識別子ではない場合、XML タグのエスケープをオンまたはオフにします。
setStylesheetHeader()（7-14 ページ）	スタイルシート・ヘッダーを設定します。
setXSLT()（7-15 ページ）	生成された XML に適用する XSL 変換を登録します。
setXSLTParam()（7-16 ページ）	トップレベルのスタイルシート・パラメータの値を設定します。
useLowerCaseTagNames()（7-16 ページ）	タグ名を小文字に設定します。
useNullAttributeIndicator()（7-16 ページ）	特別な XML 属性を使用するか、または XML 文書からのエンティティを省略することで、null であることを示すかどうかを指定します。
useTypeForCollElemTag()（7-17 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かどうかを XSU に通知します。
useUpperCaseTagNames()（7-17 ページ）	タグ名を大文字に設定します。

OracleXMLQuery()

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 の結果セット・オブジェクト
query	SQL 問合せ文字列
dset	データセット

close()

OracleXML エンジンによって作成された、すべてのオープン・リソースをクローズします。ユーザーが指定したインスタンス結果セットはクローズしません。

構文

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

getNumRowsProcessed()

処理された行の数を戻します。

構文

```
public long getNumRowsProcessed();
```

getXMLDOM()

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 に変換します。XML 文書の表現を戻します。 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 文書の文字列表現を戻します。

構文	説明
`public org.w3c.dom.Document getXMLDOM( org.w3c.dom.Node root);`	null でない場合、引数は、XML 文書のルート要素とみなされます。XML 文書の文字列表現を戻します。
`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 文書の文字列表現を戻します。

パラメータ	説明
metaType	XML メタデータの型（NONE、SCHEMA）
root	新しい XML を追加するルート・ノード

getXMLMetaData()

getXML*() コールによって生成された可能性がある XML 文書の DTD または XML Schema を戻します。たとえば、getXMLDOM()、getXMLSAX()、getXMLSchema()、getXMLString() などです。

構文

```
public String getXMLMetaData( int metaType,
                              boolean withVer);
```

パラメータ	説明
metaType	生成される XML メタデータの型（NONE または DTD）を指定します。
withVer	バージョン処理命令を生成するかどうかを指定します。

getXMLSAX()

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()

指定された問合せに対応する XML Schema を生成して、XML Schema を戻します。

構文

```
public org.w3c.dom.Document[] getXMLSchema();
```

getXMLString()

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();`	引数は取りません。
`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 の作成を要求しても生成されないことに注意してください。

パラメータ	説明
metaType	XML メタデータの型（NONE、DTD または SCHEMA など、この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()

永続化機能を設定します。ResultSet オブジェクトを取らないすべての getXML*() ファンクション（getXMLDOM()、getXMLSAX()、getXMLSchema()、getXMLString() など）のデフォルトの動作は、コールの終わりに ResultSet オブジェクトおよび Statement オブジェクトをクローズすることです。永続化機能が必要な場合、getXML() を繰り返しコールすることで、次の行セットが取得されます。この動作は、このファンクションの値を true に設定してコールすることで、オフにする必要があります。getXML() のコール後、OracleXMLQuery は、ResultSet オブジェクトおよび Statement オブジェクトをクローズしません。カーソルの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かどうか

removeXSLTParam()

トップレベルのスタイルシート・パラメータの値を削除します。注意 : スタイルシートが登録されていない場合、このメソッドは動作しないことに注意してください。

構文

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

パラメータ	説明
name	パラメータ名

setCollIdAttrName()

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渡すと、行 ID 属性は省略されます。

構文

```
public void setCollIdAttrName( String attrName);
```

パラメータ	説明
attrName	属性名

setDataHeader()

XML データ・ヘッダーを設定します。XML データ・ヘッダーは、問合せで生成された XML エンティティ（行セット）の先頭に追加される XML エンティティです。2 つのエンティティが、docTag 引数を介して指定されたタグによって囲まれます。最後に指定されたデータ・ヘッダーが使用されます。header 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

```
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()

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	データ・マスク

setEncoding()

XML 文書に処理命令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	CML 文書のエンコーディング（エンコーディングの IANA 名）

setErrorTag()

XML 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()

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 で処理される例外

setMaxRows()

XML に変換され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()

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）の先頭にヘッダーが挿入されます。最後に指定されたメタ・ヘッダーが使用されます。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()

呼び出された例外を発生させるかどうかを XSU に通知します。これがコールされない場合、または flag 引数に false が渡された場合、XSU は SQL 例外をキャッチし、例外メッセージから XML 文書を生成します。

構文

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

パラメータ	説明
flag	呼び出された例外を発生させるかどうか

setRaiseNoRowsException()

生成された XML 文書が空の場合に、OracleXMLNoRowsException が発生するかどうかを 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

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

パラメータ	説明
flag	データがない場合に、OracleXMLNoRowsException が発生するかどうか

setRowIdAttrName()

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渡すと、行 ID 属性は省略されます。

構文

```
public void setRowIdAttrName( String attrName);
```

パラメータ	説明
attrName	属性名

setRowIdAttrValue()

行の囲みタグの ID 属性に値が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渡すと、行 ID 属性には行カウント値（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	行 ID 属性に割り当てられる値を持つ列

setRowsetTag()

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()

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()

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()

XML 識別子にマップされた SQL オブジェクト名が有効な XML 識別子ではない場合、XML タグのエスケープをオンまたはオフにします。

構文

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

パラメータ	説明
flag	SQL の XML 識別子のエスケープをオンにするかどうか

setStylesheetHeader()

生成された XML 文書に、スタイルシート処理命令を含むスタイルシート・ヘッダーを設定します。引数に null を渡すと、スタイルシート・ヘッダーおよびスタイルシート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 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 URI およびスタイルシート型を使用して、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 URI
type	スタイルシートの型（デフォルトは text/xsl）

setXSLT()

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 値を指定します。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渡されます。
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への URI として渡されます。

パラメータ	説明
stylesheet	スタイルシート
ref	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()

トップレベルのスタイルシート・パラメータの値を設定します。パラメータの値は、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引用符で囲む必要があります。スタイルシートが登録されていない場合、このメソッドは動作しないことに注意してください。

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()

すべてのタグ名を小文字に設定します。すべての希望のタグが設定された後に、これをコールする必要があります。

構文

```
public void useLowerCaseTagNames();
```

useNullAttributeIndicator()

特別な XML 属性を使用するか、または XML 文書からのエンティティを省略することで、null であることを示すかどうかを指定します。

構文

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

パラメータ	説明
flag	null を示すために属性を使用するかどうか

useTypeForCollElemTag()

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。引数の値に 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	タグ名を示すために列の要素型を使用するかどうか

useUpperCaseTagNames()

すべてのタグ名を大文字に設定します。すべての希望のタグが設定された後に、これをコールする必要があります。

構文

```
public void useUpperCaseTagNames();
```

OracleXMLSave クラス

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ビューへの正規の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。ユーザーはまず、これらの DML 操作を行う必要がある表名を指定して、クラスを作成します。その後、この表に対して挿入、更新および削除ができるようになります。

このクラスでは、更新または削除のためのキー列を識別したり、更新される列を制限するために有効な多くの機能が提供されています。

構文

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

表 7-3 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 文書で SQL による XML エスケープが行われたかどうかを示します。

表 7-4 OracleXMLSave のメソッドの概要

メソッド	説明
OracleXMLSave()（7-19 ページ）	Save クラスのパブリック・コンストラクタです。
close()（7-20 ページ）	この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()（7-20 ページ）	XML 文書に基づいて表の行を削除します。
getURL()（7-21 ページ）	指定されたファイル名または URL の URL オブジェクトを戻します。
insertXML()（7-21 ページ）	指定した表に XML 文書を挿入します。
removeXSLTParam()（7-22 ページ）	トップレベルのスタイルシート・パラメータの値を削除します。
setBatchSize()（7-23 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。

表 7-4 OracleXMLSave のメソッドの概要（続き）

メソッド	説明
setCommitBatch()（7-23 ページ）	コミットのバッチ・サイズを設定します。
setDateFormat()（7-24 ページ）	XML 文書に生成日付の書式を設定します。
setIgnoreCase()（7-24 ページ）	データベース列または属性に対して XML 要素の大 / 小文字を区別しないことを XSU に通知します。
setKeyColumnList()（7-25 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()（7-25 ページ）	空白を保持するかどうかを XSU に通知します。
setRowTag()（7-25 ページ）	XML 文書で使用するタグ名を指定し、各行の値に対応する XML 要素を囲みます。
setSQLToXMLNameEscaping()（7-26 ページ）	SQL オブジェクト名が有効な XML 識別子ではない場合、XML タグのエスケープをオンまたはオフにします。
setUpdateColumnList()（7-26 ページ）	更新する列の値を設定します。
setXSLT()（7-27 ページ）	生成された XML に適用する XSL 変換を登録します。
setXSLTParam()（7-27 ページ）	トップレベルのスタイルシート・パラメータの値を設定します。
updateXML()（7-28 ページ）	XML 文書を指定して、表を更新します。

OracleXMLSave()

OracleXMLSave クラスのパブリック・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	Connection 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()

この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()

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、削除する行を識別するために一致する必要がある列のリストを設定し、他の要素は無視されます。このメソッドは、(DELETE 文をキャッシュできるため) 一致を使用して表の 1 つ以上の行を削除する場合に効率的です。このメソッドを使用しない場合、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 形式です。
`public int deleteXML( java.io.InputStream xmlStream);`	XML 文書はストリーム形式です。
`public int deleteXML( java.io.Reader xmlReader);`	XML 文書は Reader 形式です。
`public int deleteXML( String xmlDoc);`	XML 文書は文字列形式です。
`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

パラメータ	説明
xmlReader	Reader 形式の XML 文書
xmlDoc	文字列形式の XML 文書
url	表への行の削除に使用するドキュメントの URL

getURL()

指定されたファイル名または URL のターゲット・エンティティを識別する URL オブジェクトを戻します。渡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「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渡された場合、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()

指定した表に 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値を表に挿入し、その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、その他の列には null 以外の値が挿入されます。それ以外の場合は、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);	InputStream から XML 文書を挿入します。

構文	説明
`public int insertXML( java.io.Reader xmlStream);`	Reader 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	表への行の挿入に使用する DOM
xmlStream	表への行の挿入に使用するデータのストリーム
xmlDOC	表への行の挿入に使用する文字列
url	表への行の挿入に使用するドキュメントの URL

removeXSLTParam()

トップレベルのスタイルシート・パラメータの値を削除します。スタイルシートが登録されていない場合、このメソッドは動作しないことに注意してください。

構文

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

パラメータ	説明
name	パラメータ名

setBatchSize()

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、I/O サイクルを最小限にするためにバッチ処理を行うことをお薦めします。ただし、操作の実行中にバインド値を格納するため、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、バッチ処理中のみコミットが発生します。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。これを防止するには、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()

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挿入の回数または記録を意味します。size 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いません。コミットのデフォルトのバッチ・サイズは 0 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()

XML 文書に生成日付の書式を設定します。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して、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()

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、XSU が大文字 / 小文字を区別しないように通知します。このように大文字 / 小文字の区別をなくすと、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
flag	XML 文書内でタグの大文字 / 小文字を無視するかどうか

setKeyColumnList()

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、キー列を設定する必要があります。これは、削除用のオプションです。このキー列を設定すると、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文字 / 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()

空白を保持するかどうかを XSU に通知します。

構文

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

パラメータ	説明
flag	空白を保持するかどうか

setRowTag()

XML 文書で使用するタグ名を指定し、各行の値に対応する XML 要素を囲みます。この値を null に設定すると、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
tag	タグ名

setSQLToXMLNameEscaping()

XML 識別子にマップされた SQL オブジェクト名が有効な XML 識別子ではない場合、XML タグのエスケープをオンまたはオフにします。

構文

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

パラメータ	説明
flag	SQL による XML エスケープをオンにするかどうか

setUpdateColumnList()

更新する列の値を設定します。挿入および更新に提供されますが、削除には適用されません。

- 挿入の場合、デフォルトでは、表内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、更新または挿入文で更新されます。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
updColNmaes	更新する列の文字列リスト

setXSLT()

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 を指定します。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渡されます。

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への URI として渡されます。

パラメータ	説明
stylesheet	スタイルシート URI
ref	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()

トップレベルのスタイルシート・パラメータの値を設定します。パラメータの値は、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引用符で囲む必要があります)。スタイルシートが登録されていない場合、このメソッドは動作しないことに注意してください。

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()

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、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 文書内の対応する要素の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すべての列を更新します。各 ROW 要素は、表に対する個別の UPDATE 文として受け入れられます。
- 更新する列のリストを提供して、希望する列のみを更新し、XML 文書内の他のすべての要素を無視することもできます。このメソッドは、UPDATE 文自体がキャッシュされバッチ処理が行われるため、入力される XML 文書に複数の行が存在する場合に非常に効果的です。
- 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);`	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( 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 文書の要素値に基づいて、データベース表の列を更新します。

パラメータ	説明
`doc`	XML 文書の DOM ツリー形式
`xmlStream`	XML 文書のストリーム形式
`xmlDoc`	XML 文書の文字列形式
`url`	表の更新に使用するドキュメントの URL

OracleXMLSQLException クラス

XSU によって発生するすべての例外を管理するクラスです。

構文

```
public class OracleXMLSQLException extends java.lang.RuntimeException
```

表 7-5 OracleXMLSQLException のメソッドの概要

メソッド	説明
OracleXMLSQLException()（7-29 ページ）	新しい OracleXMLSQLException を作成します。
getErrorCode()（7-30 ページ）	発生した SQL エラー・コードを戻します。
getParentException()（7-31 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()（7-31 ページ）	XML エラー文字列を、与えられたエラー・タグ名で出力します。
getXMLSQLErrorString()（7-31 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()（7-31 ページ）	XML エラー・レポートの生成に使用されるエラー・タグを設定します。

OracleXMLSQLException()

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);`	渡される親の例外を設定します。
`public OracleXMLSQLException( Exception e, String errorTagName);`	渡されるエラー・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。

構文	説明
`public OracleXMLSQLException( String message, j 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。
`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()

発生したSQLエラー・コードを戻します。

構文

```
public int getErrorCode();
```

getParentException()

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()

XML エラー文字列を、与えられたエラー・タグ名で出力します。

構文

```
public String getXMLErrorString();
```

getXMLSQLErrorString()

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()

エラー・タグ名を設定します。このエラー・タグ名は、XML エラー・レポートを生成するために、getXMLErrorString() および getXMLSQLErrorString() で使用されます。

構文

```
public void setErrorTag( String tagName);
```

パラメータ	説明
tagName	エラーのタグ名

OracleXMLSQLNoRowsException クラス

行が見つからなかったときに発生する例外です。

構文

```
public class OracleXMLSQLNoRowsException extends OracleXMLSQLException
```

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);`	渡される引数としてエラー・タグを設定します。

パラメータ	説明
errorTag	エラー・タグ

8

XSQL Pages Publishing Framework for Java

XSQL Pages Publishing Framework は oracle.xml.xsql パッケージの中に含まれ、Oracle XSQL Servlet とも呼ばれます。

この章の内容は次のとおりです。

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

oracle.xml.xsql パッケージ

Oracle XSQL Pages Publishing Framework エンジン 表 8-1 では、このパッケージのクラスとインタフェースを一覧にしています。

表 8-1 oracle.xml.xsql のクラスとインタフェースの一覧

クラス / 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 の基本となる実装です。
XSQLPageRequest インタフェース	XSQL ページ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 ページへのリクエストをプログラム処理します。
XSQLRequestObjectListener インタフェース	アクション・ハンドラで作成したオブジェクトのインタフェース。現行のページ・リクエスト処理が完了したときに通知を受けるために実装できます。
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機構です。
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してください。
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してください。
XSQLDocumentSerializer インタフェース	XSQL データ・ページを XML 文書としてプリント・ライターにシリアライズする、すべての XSQL シリアライザが実装する必要があります。

XSQLActionHandler インタフェース

すべての XSQL アクション・エレメント・ハンドラが実装しています。XSQL アクション・エレメントが <xsql:xxxx> の形式で XSQL ページに発生すると、XSQL Page Processor は対応する XSQL アクション・ハンドラをコールするために、このハンドラのインスタンスを引数なしのコンストラクタを使用して作成し、XSQL アクション・ハンドラの `handleAction()` メソッドをコールします。

構文

```
public interface XSQLActionHandler
```

表 8-2 XSDLAction ハンドラのメソッドの概要

メソッド	説明
handleAction()（8-3 ページ）	アクションを処理します。
init()（8-4 ページ）	アクション・ハンドラを初期化します。

handleAction()

コードを実行し、新しい子である DOM ノードをルートに追加することによって、アクションを処理します。XSQL Page Processor は、処理中の XSQL Page 内のアクション・エレメントをノードのドキュメント・フラグメントに置き換えます。

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()

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	処理中のアクション・エレメントを表す DOM 要素

XSQLActionHandlerImpl クラス

カスタム・ハンドラを作成する XSQLActionHandler の基本となる実装です。一連の有効な補助メソッドを含みます。このクラスを拡張して `init()` メソッドをオーバーライドする場合は、`super.init(env,e)` をコールする必要があります。

構文

```
public abstract class XSQLActionHandlerImpl extends java.lang.Object implements
XSQLActionHandler Interface
```

表 8-3 XSQLActionHandlerImpl のメソッドの概要

メソッド	説明
XSQLActionHandlerImpl() (8-5 ページ)	クラス・コンストラクタです。
init() (8-5 ページ)	アクション・ハンドラを初期化します。

XSQLActionHandlerImpl()

クラス・コンストラクタです。

構文

```
public  XSQLActionHandlerImpl();
```

init()

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・エレメント

XSQLPageRequest インタフェース

XSQL ページに対するリクエストを表すインタフェースです。

構文

```
public interface XSQLPageRequest
```

表 8-4 XSQLPageRequest のメソッドの概要

メソッド	説明
createNestedRequest()（8-8 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()（8-8 ページ）	このリクエストに使用されている接続名を戻します。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。
getErrorWriter()（8-8 ページ）	この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()（8-9 ページ）	このリクエストに使用されている JDBC 接続を取得します(null の場合もあります)。
getPageEncoding()（8-9 ページ）	この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()（8-9 ページ）	リクエストされたパラメータ値を戻します。
getPostedDocument()（8-9 ページ）	このリクエストに対する Posted XML の内容を、XML 文書として戻します。
getRequestParamsAsXMLDocument()（8-10 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()（8-10 ページ）	作成中のページ・リクエストのタイプを識別する文字列を戻します。
getSourceDocumentURI()（8-10 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。
getStylesheetParameters()（8-10 ページ）	スタイルシート・パラメータを名前で取得します。
getStylesheetParameters()（8-11 ページ）	列挙されたスタイルシート・パラメータ名を取得します。
getStylesheetURI()（8-11 ページ）	結果を処理するために使用するスタイルシートの URI を戻します。

表 8-4 XSQLPageRequest のメソッドの概要（続き）

メソッド	説明
getUserAgent()（8-11 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()（8-11 ページ）	ページ・リクエスト結果の出力に使用するプリント・ライターを戻します。
getXSQLConnection()（8-12 ページ）	この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 の場合もあります。
isIncludedRequest()（8-12 ページ）	この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()（8-12 ページ）	現行の接続が Oracle JDBC Driver を使用している場合、true を戻します。
printedErrorHeader()（8-12 ページ）	エラー・ハンドラが出力されたかどうかの状態を戻します。
requestProcessed()（8-13 ページ）	ページ・リクエストでリクエスト末尾処理を実行します。
setConnectionName()（8-13 ページ）	このリクエストに使用する接続名を設定します。
setContentType()（8-13 ページ）	結果ページのコンテンツ・タイプを設定します。
setIncludingRequest()（8-14 ページ）	この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()（8-14 ページ）	この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()（8-14 ページ）	動的ページ・パラメータ値を設定します。
setPostedDocument()（8-15 ページ）	ポストされたドキュメントのプログラム設定を可能にします。
setPrintedErrorHeader()（8-15 ページ）	エラー・ハンドラが出力されたかどうかを設定します。
setStylesheetParameter()（8-15 ページ）	対応するスタイルシートに渡されるパラメータ値を設定します。
setStylesheetURI()（8-16 ページ）	結果を処理するために使用するスタイルシートの URI を設定します。
translateURL()（8-16 ページ）	このリクエストのベース URI にあわせて変換された絶対 URL を表す文字列を戻します。
useConnectionPooling()（8-16 ページ）	このリクエストに接続プーリングが必要な場合、true を戻します。

表 8-4 XSQLPageRequest のメソッドの概要（続き）

メソッド	説明
useHTMLErrors()（8-16 ページ）	このリクエストに HTML 形式のエラー・メッセージが必要な場合、true を戻します。
setRequestObject()（8-17 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()（8-17 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()

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・ディクショナリ

getConnectionName()

このリクエストに使用されている接続名を戻します。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()

この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()

このリクエストに使用されている JDBC 接続を取得します（null の場合もあります）。

構文

```
public java.sql.Connection getJDBCConnection();
```

getPageEncoding()

この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()

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	パラメータの名前

getPostedDocument()

このリクエストに対する Posted XML の内容を、XML 文書として戻します。

構文

```
public oracle.xml.xsql.Document getPostedDocument();
```

getRequestParamsAsXMLDocument()

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()

作成中のページ・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()

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();
```

getStylesheetParameters()

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・パラメータ名

getStylesheetParameters()

列挙されたスタイルシート・パラメータ名を取得します。

構文

```
public java.util.Enumeration getStylesheetParameters();
```

getStylesheetURI()

結果を処理するために使用するスタイルシートの URI を戻します。

構文

```
public java.lang.String getStylesheetURI();
```

getUserAgent()

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()

ページ・リクエスト結果の出力に使用するプリント・ライターを戻します。

構文

```
public java.io.PrintWriter getWriter();
```

getXSQLConnection()

この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 の場合もあります。

構文

```
public oracle.xml.xsql.XSQLConnection getXSQLConnection();
```

isIncludedRequest()

この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()

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()

エラー・ヘッダーが出力されたかどうかの状態を戻します。

構文

```
public boolean printedErrorHeader();
```

requestProcessed()

ページ・リクエストでリクエスト末尾処理を実行します。

構文

```
public void requestProcessed();
```

setConnectionName()

この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	接続の名前

setContentType()

結果ページのコンテンツ・タイプを設定します。

構文

```
public void setContentType( String mimetype);
```

パラメータ	説明
mimetype	結果ページの内容を表す MIME タイプ

setIncludingRequest()

この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	Including Page の XSQLPageRequest コンテキスト

setPageEncoding()

この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	現行ページのエンコーディング

setPageParam()

この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 setPageParam( String name, String value);
```

パラメータ	説明
name	ページ専用パラメータの名前
value	ページ専用パラメータの値

setPostedDocument()

ポストされたドキュメントのプログラム設定を可能にします。

構文

```
public void setPostedDocument( org.w3c.dom.Document doc);
```

パラメータ	説明
doc	このリクエストの一部としてポストされた XML 文書

setPrintedErrorHeader()

エラー・ヘッダーが出力された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	エラー・ヘッダーが出力されたかどうかを設定します。

setStylesheetParameter()

対応するスタイルシートに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()

結果を処理するために使用す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	このリクエストの変換に使用するスタイルシートの URI。

translateURL()

このリクエストのベース URI にあわせて変換された絶対 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	このリクエストの変換に使用するスタイルシートの URL。

useConnectionPooling()

このリクエストに接続プーリングが必要な場合、true を戻します。

構文

```
public boolean useConnectionPooling();
```

useHTMLErrors()

このリクエストに HTML 形式のエラー・メッセージが必要な場合、true を戻します。

構文

```
public boolean useHTMLErrors();
```

setRequestObject()

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・オブジェクト自体

getRequestObject()

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

一般的な XML 解析ルーチンです。

構文

```
public final class XSQLParserHelper extends java.lang.Object
```

表 8-5 XSQLParserHelper のメソッドの概要

メソッド	説明
XSQLParserHelper()（8-18 ページ）	クラス・コンストラクタです。
newDocument()（8-18 ページ）	新しい空の XML 文書を戻します。
parse()（8-19 ページ）	XML 文書を様々なソースから解析します。
parseFromString()（8-19 ページ）	XML 文書を文字列から解析します。
print()（8-20 ページ）	XML 文書を出力します。

XSQLParserHelper()

クラス・コンストラクタです。

構文

```
public  XSQLParserHelper();
```

newDocument()

新しい空の XML 文書を戻します。

構文

```
public static oracle.xml.xsql.Document newDocument();
```

parse()

XML 文書を様々なソースから解析します。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);`	XML 文書を InputStream オブジェクトから解析します。
`public static oracle.xml.xsql.Document parse( Reader r, PrintWriter errorWriter);`	XML 文書を Reader オブジェクトから解析します。
`public static oracle.xml.xsql.Document parse( URL url, PrintWriter errorWriter);`	XML 文書を URL から解析します。

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・メッセージを受け取るプリント・ライター
`r`	解析する XML を含む Reader
`url`	解析する XML 文書の URL

parseFromString()

XML 文書を文字列から解析します。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解析します。
`public static oracle.xml.xsql.Document parseFromString( String xmlString, PrintWriter errorWriter);`	文字列から解析します。

パラメータ	説明
xmlString	解析する XML を含む文字列
errorWriter	あらゆるエラー・メッセージを受け取るプリント・ライター

print()

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	文書のシリアライズに使用するプリント・ライター

XSQLRequest クラス

XSQL ページへのリクエストをプログラム処理します。

構文

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

表 8-6 XSQLRequest のメソッドの概要

メソッド	説明
XSQLRequest()（8-21 ページ）	クラス・コンストラクタです。XSQL Page へのリクエストを作成します。
process()（8-22 ページ）	リクエストを処理し、出力およびエラーを書き込みます。
processToXML()（8-23 ページ）	リクエストを処理し、出力およびエラーを書き込みます。
setPostedDocument()（8-23 ページ）	リクエストの一部としてポストされた場合と同様に処理されるように、XML 文書のプログラム設定を行います。

XSQLRequest()

XSQLRequest のインスタンスを様々なソースから構成します。

構文

```
public  XSQLRequest( java.lang.String url);
```

パラメータ	説明
url	処理する XSQL ページ・ソースの URL

process()

リクエストを処理し、出力およびエラーを書き込みます。次の表に、オプションを示します。

構文	説明
`public void process();`	リクエストを処理し、出力およびエラーを System.out と System.err に書き込みます。
`public void process( Dictionary params);`	リクエストを処理し、出力およびエラーを System.out と System.err に書き込みます。
`public void process( Dictionary params, PrintWriter out, PrintWriter err);`	リクエストを処理し、出力およびエラーを各プリント・ライターに書き込みます。
`public void process( PrintWriter out, PrintWriter err);`	リクエストを処理し、出力およびエラーを各プリント・ライターに書き込みます。

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`	結果ページの結果を書き込むために使用するプリント・ライター
`err`	結果ページのエラーを書き込むために使用するプリント・ライター

processToXML()

リクエストを処理し、出力およびエラーを書き込みます。次の表に、オプションを示します。

構文	説明
`public org.w3c.dom.Document processToXML()`	リクエストを処理し、出力およびエラーを System.out と System.err に書き込みます。
`public org.w3c.dom.Document processToXML( Dictionary params)`	リクエストを処理し、出力およびエラーを System.out と System.err に書き込みます。
`public org.w3c.dom.Document processToXML( Dictionary params, PrintWriter err)`	リクエストを処理し、出力およびエラーを各プリント・ライターに書き込みます。
`public org.w3c.dom.Document processToXML( PrintWriter err);`	リクエストを処理し、エラーを各プリント・ライターに書き込みます。

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を書き込むために使用するプリント・ライター

setPostedDocument()

リクエストの一部としてポストされた場合と同様に処理されるように、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

アクション・ハンドラで作成したオブジェクトのインタフェースです。現行のページ・リクエスト処理が完了したときに通知を受けるために実装できます。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。

pageProcessingCompleted()

このコールバック・メソッドを使用すると、リクエスト・スコープ・オブジェクトは、ページ処理が完了して割り当てられたリソースが消去可能になったときに通知を受けることができます。XSQLRequestObjectListener インタフェースを実装するリクエスト・スコープ・オブジェクトは、この方法で通知されます。

構文

```
void pageProcessingCompleted();
```

XSQLServletPageRequest クラス

構文

```
public final class XSQLServletPageRequest extends XSQLPageRequestImpl
```

表 8-7 XSQLServletPageRequest のメソッドの概要

メソッド	説明
XSQLServletPageRequest()（8-26 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()（8-26 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()（8-26 ページ）	HTTP リクエスト・クッキーの値を取得します。
getHttpServletRequest()（8-27 ページ）	この XSQL ページ・リクエストを開始した HttpServletRequest を取得します。
getHttpServletResponse()（8-27 ページ）	この XSQL ページ・リクエストに対応する HttpServletResponse を取得します。
getParameter()（8-27 ページ）	パラメータのソースとして、HTTP パラメータを代用します。
getPostedDocument()（8-27 ページ）	この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()（8-28 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()（8-28 ページ）	リクエストのタイプを戻します。
getUserAgent()（8-28 ページ）	ブラウザから渡されたユーザー・エージェント文字列を戻します。
setContentType()（8-28 ページ）	結果ページのコンテンツ・タイプを設定します。
setPageEncoding()（8-29 ページ）	ページのエンコーディングを設定します。
translateURL()（8-29 ページ）	このリクエストのベース URI にあわせて変換された絶対 URL を表す文字列を戻します。
useHTMLErrors()（8-29 ページ）	HTML 形式のエラーを使用するかどうかを設定します。

XSQLServletPageRequest()

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	応答

createNestedRequest()

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 URL
prams	リクエストのパラメータ

getCookie()

HTTP リクエスト・クッキーの値を取得します。

構文

```
public java.lang.String getCookie(   String name);
```

パラメータ	説明
name	取得するクッキーの名前

getHttpServletRequest()

この 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()

この 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()

パラメータのソースとして、HTTP パラメータを代用します。

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()

このリクエストに対してポストされた XML 文書を取得します。ない場合は NULL を戻します。

構文

```
public oracle.xml.xsql.Document getPostedDocument();
```

getRequestParamsAsXMLDocument()

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()

リクエストのタイプ（「サーブレット」、「プログラム」、「コマンドライン」、「カスタム」）を戻します。

構文

```
public java.lang.String getRequestType();
```

getUserAgent()

ブラウザから渡されたユーザー・エージェント文字列を戻します。

構文

```
public java.lang.String getUserAgent();
```

setContentType()

結果ページのコンテンツ・タイプを設定します。

構文

```
public void setContentType( String mimetype);
```

パラメータ	説明
mimtype	結果ページの内容を表す MIME タイプ。

setPageEncoding()

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()

このリクエストのベース URI にあわせて変換された絶対 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()

HTML 形式のエラーを使用するかどうかを設定します。

構文

```
public boolean useHTMLErrors();
```

XSQLStylesheetProcessor クラス

XSLT スタイルシート処理機構です。

構文

```
public final class XSQLStylesheetProcessor extends java.lang.Object
```

表 8-8 XSQLStylesheetProcessor のメソッドの概要

メソッド	説明
XSQLStylesheetProcessor()（8-30 ページ）	CSLT スタイルシート変換を処理します。
processToDocument()（8-31 ページ）	XML 文書を XSLT スタイルシートで変換し、その結果を XML 文書で戻します。
processToWriter()（8-31 ページ）	CML 文書を XSLT スタイルシートで変換し、その結果をプリント・ライターに書き込みます。
getServletContext()（8-31 ページ）	HTTP サーブレット・コンテキストを取得します。

XSQLStylesheetProcessor()

CSLT スタイルシート変換を処理します。

構文

```
public  XSQLStylesheetProcessor();
```

processToDocument()

XML 文書を XSLT スタイルシートで変換し、その結果を XML 文書で戻します。

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 URI
env	XSQLPageRequest コンテキスト

processToWriter()

CML 文書を XSLT スタイルシートで変換し、その結果をプリント・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 URI
env	XSQLPageRequest コンテキスト

getServletContext()

HTTP サーブレット・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

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 Page Processor は、各リクエストに対応付けられた `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 の実装によって、`getConnection()` が戻した XSQLConnection インスタンスは、`releaseConnection()` のコール後に XSQL Page Processor によってそのインスタンスが解放されるまで、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。

構文

```
public interface XSQLConnectionManager;
```

表 8-9 XSQLConnectionManager のメソッドの概要

メソッド	説明
getConnection()（8-32 ページ）	Connection Manager から接続を取得します。
releaseConnection()（8-33 ページ）	接続を解放します。

getConnection()

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()

接続を解放します。

構文

```
void releaseConnection( XSQLConnection theConn,
                        XSQLPageRequest env);
```

パラメータ	説明
theConn	解放する XSQL 接続オブジェクト
env	XSQLPageRequest コンテキスト

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 Page Processor は、各リクエストに対応付けられた 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。

構文

```
public interface XSQLConnectionManagerFactory
```

create()

XSQLConnectionManager のインスタンスを戻します。実装は、これが新しい XSQLConnectionManager かまたは共有 Singleton かを判別できます。

構文

```
XSQLConnectionManager create();
```

XSQLDocumentSerializer インタフェース

XSQL データ・ページを XML 文書としてプリント・ライターにシリアライズする、すべての XSQL シリアライザが実装する必要があるインタフェースです。

処理命令 <?xml-stylesheet?> に serializer="XXX" 形式の擬似属性がある場合、XSQL Page Processor 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構成します。
- 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。

XSQLDocumentSerializer の実装では、次の処理が行われます。

- env.setContentType() をコールして、コンテンツ・タイプを設定します。
- env.getWriter() をコールして、書込み先のライターを取得します。

シリアライザが処理できない例外が発生した場合、XSQL ページ・プロセッサはスタック・トレースをフォーマットします。

例は、oracle.xml.xsql.src.serializers.XSQLSampleSerializer を参照してください。

構文

```
public interface XSQLDocumentSerializer
```

serialize()

結果の XML 文書を出力ライター用にシリアライズします。

構文

```
void serialize( org.w3c.dom.Document doc,
                XSQLPageRequest env);
```

パラメータ	説明
doc	シリアライズする XML 文書
env	XSQLPageRequest コンテキスト

9

TransX Utility for Java

TransX Utility を使用すると、変換済のシード・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、変換用の文字列を準備し、変換し、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、変換データ形式のエラーが最小限に抑えられ、データベースに事前に指定された場所に変換内容が正確にロードされます。

TransX Utility を使用すると、変換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形式に変換されると、TransX Utility は、その形式を検証します。ファイルのエンコーディングでは、エンコーディングを記述する XML を利用しているため、変換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。

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

TransX Utility コマンドライン・インタフェース

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)]
```

表 9-1 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。jdbc:oracle:thin:@ と指定されます。
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 9-2「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。

表 9-2 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。
-x	データベースのデータを事前定義の形式で出力します。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。

表 9-2 TransX Utility コマンドライン・オプション（続き）

オプション	説明
-s	データベースのデータを事前定義の形式でファイルに保存します。これは、アンロードを実行するためのオプションです。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、その結果が事前定義の 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。正規形の XSU 形式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。正規形の XSU 形式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形式の検証をスキップします。
-v	データ形式を検証し、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 9-3 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

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。

構文

```
oracle.xml.transx.loader
```

表 9-4 loader のメソッドの概要

メソッド	説明
getLoader()（9-5 ページ）	TransX のインスタンスを取得します。
main()（9-5 ページ）	コマンドライン・インタフェースです。

getLoader()

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()

コマンドライン・インタフェースのメイン・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

データのロード・ツール API です。

構文

```
public interface TransX
```

表 9-5 TransX のメソッドの概要

メソッド	説明
close() （9-6 ページ）	使用済リソースの再生を支援します。
load() （9-7 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() （9-7 ページ）	データ・ロード・セッションを開始します。
setLoadingMode() （9-8 ページ）	複製に操作モードを設定します。
setPreserveWhitespace() （9-8 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() （9-9 ページ）	検証モードを設定します。
unload() （9-9 ページ）	データセットをアンロードします。
validate() （9-10 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()

使用済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()

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()

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	接続が確立されるデータベース・ユーザー
pwd	ユーザーのパスワード

setLoadingMode()

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()

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()

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、正規形スキーマ定義に対してデータセット形式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	ローダーが検証する必要があるかどうかを判別します。

unload()

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)*;	読取り可能な Reader 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()

ファイルまたは URL のデータセットを検証します。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。正常に検証された場合は TRUE 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

10

Oracle XML JavaBeans

Oracle XML JavaBeans は、次のパッケージと同義です。

- oracle.xml.async パッケージ
- oracle.xml.dbviewer パッケージ
- oracle.xml.srcviewer パッケージ
- oracle.xml.transviewer パッケージ
- oracle.xml.treeviewer パッケージ
- oracle.xml.differ パッケージ

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

oracle.xml.async パッケージ

これは不可視 Bean です。バックグラウンドの個別のスレッドで、非同期の DOM 解析を実行できます。EventHandler インタフェースを使用して、ジョブの完了時にコール・クラスに通知します。表 10-1 に、oracle.xml.async のクラスを示します。

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）	XML 1.0 パーサーをカプセル化し、XML 文書の解析および DOM ツリーの構築を行います。
DOMBuilderBeanInfo クラス（10-13 ページ）	DOMBuilder Bean の情報を提供します。
DOMBuilderErrorEvent クラス（10-15 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（10-17 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（10-18 ページ）	DOMBuilder がすべての登録リスナーに解析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。
DOMBuilderListener インタフェース（10-20 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（10-22 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なセマフォを実装します。
XSLTransformer クラス（10-24 ページ）	バックグラウンド・スレッドで XSL 変換を行います。
XSLTransformerBeanInfo クラス（10-30 ページ）	XSLTransformer Bean の情報を提供します。
XSLTransformerErrorEvent クラス（10-32 ページ）	XSLTransformer が、すべての登録リスナーに変換エラー・イベントを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

XML 1.0 パーサーをカプセル化し、XML 文書の解析および DOM ツリーの構築を行います。ツリーが構築されている場合、解析は別のスレッドで行われ、DOMBuilderListener インタフェースを使用して通知する必要があります。

構文

```
public class DOMBuilder extends java.lang.Object implements java.io.Serializable,
oracle.xml.async.DOMBuilderConstants, java.lang.Runnable
```

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 データの解析元になる 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 データの解析元になる 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
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

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()（10-5 ページ）	DOMBuilderErrorListener を追加します。
addDOMBuilderListener()（10-5 ページ）	DOMBuilderListener を追加します。
getDoctype()（10-6 ページ）	DTD を取得します。
getDocument()（10-6 ページ）	解析するドキュメントを取得します。
getId()（10-6 ページ）	パーサー・オブジェクトの ID を戻します。
getReleaseVersion()（10-6 ページ）	Oracle XML Parser のバージョン番号を戻します。
getResult()（10-6 ページ）	解析中のドキュメントを取得します。
getValidationMode()（10-7 ページ）	検証モードを戻します。
parse()（10-7 ページ）	与えられた入力から XML を解析します。
parseDTD()（10-8 ページ）	XML 外部 DTD を解析します。
removeDOMBuilderErrorListener()（10-9 ページ）	DOMBuilderErrorListener を削除します。
removeDOMBuilderListener()（10-9 ページ）	DOMBuilderListener を削除します。
run()（10-9 ページ）	スレッドで実行します。
setBaseURL()（10-10 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()（10-10 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()（10-10 ページ）	DTD を設定します。
setErrorStream()（10-11 ページ）	エラーおよび警告の出力ストリームを作成します。
setNodeFactory()（10-11 ページ）	ノード・ファクトリを設定します。
setPreserveWhitespace()（10-12 ページ）	空白保持モードを設定します。
setValidationMode()（10-12 ページ）	検証モードを設定します。
showWarnings()（10-12 ページ）	警告を出力するかどうかを決定します。

DOMBuilder()

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);`	与えられた ID で新しいパーサー・オブジェクトを作成します。

パラメータ	説明
`id`	DOMBuilder の ID

addDOMBuilderErrorListener()

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 DOMBuilderListener

addDOMBuilderListener()

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()

DTD を取得します。

構文

```
public synchronized oracle.xml.async.DTD getDoctype();
```

getDocument()

解析するドキュメントを取得します。

構文

```
public synchronized oracle.xml.async.XMLDocument getDocument();
```

getId()

パーサー・オブジェクトの ID（DOMBuilder）を戻します。

構文

```
public int getId();
```

getReleaseVersion()

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()

解析中のドキュメントを取得します。

構文

```
public synchronized org.w3c.dom.Document getResult();
```

getValidationMode()

検証モードを戻します。XML パーサーが検証中の場合は true、検証中でない場合は false を戻します。

構文

```
public synchronized boolean getValidationMode()
```

parse()

与えられた入力から XML を解析します。次の例外が発生します。

- XMLParseException - 構文エラーまたは他のエラーが起きた場合に発生します。
- SAXException - SAX の例外であり、別の例外が隠されている可能性があります。
- IOExceptiopn - I/O エラーです。

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);`	InputStream から XML を解析します。外部エンティティおよび DTD を変換するために、ベース URL を設定する必要があります。
`public final synchronized void parse ( Reader r);`	Reader から XML を解析します。外部エンティティおよび DTD を変換するために、ベース URL を設定する必要があります。
`public final synchronized void parse ( String urlName);`	引数で指定した URL からの XML を解析します。
`public final synchronized void parse ( URL url);`	与えられた 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 Reader
urlName	解析元になる URL を含む String
url	解析する XML 文書を指す ULR

parseDTD()

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起きた場合に発生します。
- SAXException - SAX の例外であり、別の例外が隠されている可能性があります。
- IOExceptiopn - I/O エラーです。

構文	説明
`public final synchronized void parseDTD ( InputSource in, String rootName);`	与えられた入力ソースから解析します。
`public final synchronized void parseDTD ( InputStream in, String rootName);`	与えられた InputStream から解析します。外部エンティティおよび DTD を変換するために、ベース URL を設定する必要があります。
`public final synchronized void parseDTD ( Reader in, String rootName);`	与えられた Reader から解析します。外部エンティティおよび DTD を変換するために、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);`	指定した URL から解析します。
`public final synchronized void parseDTD ( URL url, String rootName);`	与えられた URL が指す外部 DTD ドキュメントを解析し、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
rootName	ルート要素として使用される要素
r	解析する XML データを含む Reader
urlName	解析元になる URL を含む String
utl	解析する XML 文書を指す ULR

removeDOMBuilderErrorListener()

DOMBuilderErrorListener を削除します。

構文

```
public synchronized void removeDOMBuilderErrorListener(
                DOMBuilderErrorListener p0);
```

パラメータ	説明
po	削除する DOMBuilderErrorListener

removeDOMBuilderListener()

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()

メソッドを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() で指定します。

構文

```
public void run();
```

setBaseURL()

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。parse(InputStream) オプションが XML 文書の解析に使用される場合は、このメソッドを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()

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断します。格納する場合は true です。

setDoctype()

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	設定する DTD。解析中に使用されます。

setErrorStream()

エラーおよび警告の出力ストリームを作成します。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、標準のエラー出力ストリームである 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ていない場合は 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
enc	使用するエンコーディング

setNodeFactory()

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

構文

```
public synchronized void setNodeFactory(
                oracle.xml.async.NodeFactory factory);
```

パラメータ	説明
factory	設定するノード・ファクトリ

setPreserveWhitespace()

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 に設定します。

setValidationMode()

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 パーサーが検証を行う必要があるかどうかを決定します。検証を行う場合は true に設定します。

showWarnings()

警告を出力するかどうかを決定します。

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決定します。警告を出力する場合は true、出力しない場合は false に設定します。

DOMBuilderBeanInfo クラス

DOMBuilder Bean の情報を提供します。

構文

```
public class DOMBuilderBeanInfo extends java.beans.SimpleBeanInfo
```

表 10-4 DOMBuilderBeanInfo のメソッドの概要

メソッド	説明
DOMBuilderBeanInfo()（10-13 ページ）	デフォルトのコンストラクタです。
getIcon()（10-13 ページ）	ツールバーなどで DOMBuilder Bean を表すために使用できるイメージ・オブジェクトを取得します。
getPropertyDescriptors()（10-14 ページ）	DOMBuilder Bean の編集可能な PropertyDescriptors の配列を取得します。

DOMBuilderBeanInfo()

デフォルトのコンストラクタです。

構文

```
public  DOMBuilderBeanInfo();
```

getIcon()

ツールバーやツールボックスなどで DOMBuilder Bean を表すために使用できるイメージ・オブジェクトを取得します。要求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。

構文

```
public java.awt.Image getIcon( int iconKind);
```

パラメータ	説明
iconKind	要求されたアイコンの種類

getPropertyDescriptors()

DOMBuilder Bean の編集可能な PropertyDescriptors の配列を取得します。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

DOMBuilderErrorEvent クラス

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。

構文

```
public class DOMBuilderErrorEvent extends java.util.EventObject
```

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()

DOMBuilderErrorEvent 用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	このエラー・イベントを作成したオブジェクト
e	発生中の例外

getException()

発生中の例外を取得します。

構文

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

getMessage()

パーサーが生成したエラー・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

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。

構文

```
public interface DOMBuilderErrorListener extends java.util.EventListener
```

domBuilderErrorCalled()

解析エラーの発生時にコールされます。

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 が生成した DOMBuilderErrorEvent オブジェクト

DOMBuilderEvent クラス

DOMBuilder がすべての登録リスナーに解析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。

構文

```
public class DOMBuilderEvent extends java.util.EventObject
```

表 10-6 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

表 10-7 DOMBuilderEvent のメソッドの概要

メソッド	説明
DOMBuilderEvent() (10-18 ページ)	新しい DOMBuilderEvent を作成します。
getID() (10-19 ページ)	イベントのソース DOMBuilder の一意の ID を戻します。

DOMBuilderEvent()

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	このイベントを作成するオブジェクト
p1	このイベントを作成する DOMBuilder の ID

getID()

複数の DOMBuilder インスタンスがバックグラウンドで動作中の場合に、このイベントを生成した DOMBuilder インスタンスを識別するために使用できるソース DOMBuilder の一意の ID を戻します。

構文

```
public int getID();
```

DOMBuilderListener インタフェース

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。

構文

```
public interface DOMBuilderListener extends java.util.EventListener
```

表 10-8 DOMBuilderListener のメソッドの概要

メソッド	説明
domBuilderError()（10-17 ページ）	解析エラーの発生時にコールされます。
domBuilderOver()（10-21 ページ）	解析の完了時にコールされます。
domBuilderStarted()（10-21 ページ）	解析の開始時にコールされます。

domBuilderError()

解析エラーの発生時にコールされます。

構文

```
public void domBuilderError( DOMBuilderEvent p0);
```

パラメータ	説明
p0	DOMBuilder が作成する DOMBuilderEvent オブジェクト

domBuilderOver()

解析の完了時にコールされます。

構文

```
public void domBuilderOver( DOMBuilderEvent p0);
```

パラメータ	説明
p0	DOMBuilder が作成する DOMBuilderEvent オブジェクト

domBuilderStarted()

解析の開始時にコールされます。

構文

```
public void domBuilderStarted( DOMBuilderEvent p0);
```

パラメータ	説明
p0	DOMBuilder が作成する DOMBuilderEvent オブジェクト

ResourceManager クラス

セマフォを実装して、固定数の論理リソースへのアクセスを維持します。

構文

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

表 10-9 ResourceManager のメソッドの概要

メソッド	説明
ResourceManager()（10-22 ページ）	ResourceManager コンストラクタです。
activeFound()（10-23 ページ）	アクティブに使用されている管理中の論理リソースがあるかどうかを確認します。
getResource()（10-23 ページ）	使用可能なリソース数が 0（ゼロ）以外である場合、使用可能なリソース数を 1 つ減らします。使用可能なリソース数が 0（ゼロ）である場合、リソースが解放され、使用可能になるまで待ちます。
releaseResource()（10-23 ページ）	単一のリソースを解放して、使用可能なリソース数を 1 つ増やします。
sleep()（10-23 ページ）	Thread.sleep() を try/catch せずに使用できるようになります。

ResourceManager()

ResourceManager コンストラクタです。

構文

```
public  ResourceManager(int i);
```

パラメータ	説明
i	管理するリソース数

activeFound()

アクティブに使用されている管理中の論理リソースがあるかどうかを確認します。1 つ以上のリソースが使用されている場合は true、使用されているリソースがない場合は false を戻します。

構文

```
public boolean activeFound();
```

getResource()

使用可能なリソース数が 0（ゼロ）以外である場合、メソッドは、使用可能なリソース数を 1 つ減らします。使用可能なリソース数が 0（ゼロ）である場合、リソースが解放され、使用可能になるまで待ちます。

構文

```
public synchronized void getResource();
```

releaseResource()

リソースを解放します。このメソッドがコールされると、使用可能なリソース数が 1 つ増えます。

構文

```
public void releaseResource();
```

sleep()

Thread.sleep() を try/catch せずに使用できるようになります。

構文

```
public void sleep(int i);
```

パラメータ	説明
i	管理するリソース数

XSLTransformer クラス

バックグラウンド・スレッドで XSL 変換を行います。

構文

```
public class XSLTransformer extends java.lang.Object implements
     java.io.Serializable, oracle.xml.async.XSLTransformerConstants,
     java.lang.Runnable

java.lang.Object
```

表 10-10 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	変換結果ドキュメント

表 10-11 XSLTransformer のメソッドの概要

メソッド	説明
XSLTransformer()（10-25 ページ）	XSLTransformer クラス・コンストラクタです。
addXSLTransformerErrorListener()（10-25 ページ）	XSLTransformerErrorListener を追加します。
addXSLTransformerListener()（10-26 ページ）	XSLTransformerListener を追加します。
getId()（10-26 ページ）	固有の XSLTransformer ID を戻します。
getResult()（10-26 ページ）	結果のドキュメントの XSL 変換のドキュメント・フラグメントを戻します。
processXSL()（10-27 ページ）	バックグラウンドで XSL 変換を開始します。制御はすぐに戻されます。
removeDOMTransformerErrorListener()（10-28 ページ）	XSLTransformerErrorListener を削除します。
removeXSLTransformerListener()（10-28 ページ）	XSLTransformerListener を削除します。
run()（10-28 ページ）	別のスレッドを起動し、XSL 変換を行います。

表 10-11 XSLTransformer のメソッドの概要（続き）

メソッド	説明
setErrorStream()（10-29 ページ）	XSL 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()（10-29 ページ）	XSL プロセッサが使用する showWarnings フラグを設定します。

XSLTransformer()

XSLTransformer 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 コンストラクタ
`public XSLTransformer( int id);`	識別子を受け入れる XSLTransformer コンストラクタ

パラメータ	説明
`id`	イベントの処理中に XSLTransformer インスタンスを識別するために使用できる整数

addXSLTransformerErrorListener()

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()

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()

固有の XSLTransformer ID を戻します。

構文

```
public int getId();
```

getResult()

結果のドキュメントの XSL 変換のドキュメント・フラグメントを戻します。このメソッドは、変換の完了通知を受け取った後にのみコールします。変換はバックグラウンドで非同期に行われるため、processXSL の直後にこのメソッドをコールすると、結果が使用可能になるまで制御が保持されます。

構文

```
public synchronized oracle.xml.async.DocumentFragment getResult();
```

processXSL()

バックグラウンドで XSL 変換を開始します。制御はすぐに戻されます。 XSL 変換中にエラーが起きた場合、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)`	InputStream としてソース XML 文書が提供されます。
`public void processXSL( oracle.xml.async.XSLStylesheet xsl, j Reader xml, URL ref);`	Reader としてソース XML 文書が提供されます。
`public void processXSL( oracle.xml.async.XSLStylesheet xsl, URL xml, URL ref);`	URL を介してソース XML 文書が提供されます。
`public void processXSL( oracle.xml.async.XSLStylesheet xsl, oracle.xml.async.XMLDocument xml);`	DOM ツリーとしてソース XML 文書が提供されます。
`public void processXSL( oracle.xml.async.XSLStylesheet xsl, oracle.xml.async.XMLDocument xml, j OutputStream os);`	DOM ツリーとしてソース XML 文書が提供され、出力が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参照 URL
os	XSL 変換の結果が書き込まれる出力

removeDOMTransformerErrorListener()

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()

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()

別のスレッドを起動し、XSL 変換を行います。java.lang.Runnable インタフェースの java.lang.Runnable.run() で指定します。

構文

```
public void run();
```

setErrorStream()

XSL 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	XSL プロセッサ用のエラー出力ストリーム

showWarnings()

XSL 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	XSL プロセッサ警告を表示する必要があるかどうかを示します。警告を表示する場合は true、表示しない場合は false に設定します。

XSLTransformerBeanInfo クラス

XSLTransformer Bean の情報を提供します。

構文

```
public class XSLTransformerBeanInfo extends java.beans.SimpleBeanInfo
```

表 10-12 XSLTransformerBeanInfo のメソッドの概要

メソッド	説明
XSLTransformerBeanInfo() (10-30 ページ)	デフォルトのコンストラクタです。
getIcon() (10-30 ページ)	要求されたアイコンの種類を表すイメージを戻します。
getPropertyDescriptors() (10-31 ページ)	XSLTransformer Bean の編集可能な PropertyDescriptors の配列を取得します。

XSLTransformerBeanInfo()

デフォルトのコンストラクタです。

構文

```
public  XSLTransformerBeanInfo();
```

getIcon()

ツールバーやツールボックスなどで XSLTransformer Bean 用に要求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	要求されたアイコンの種類

getPropertyDescriptors()

XSLTransformer Bean の編集可能な PropertyDescriptors 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

XSLTransformer が、すべての登録リスナーに変換エラー・イベントを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。

構文

```
public class XSLTransformerErrorEvent extends java.util.EventObject
```

表 10-13 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

表 10-14 XSLTransformerErrorEvent のメソッドの概要

メソッド	説明
XSLTransformerErrorEvent() (10-32 ページ)	XSLTransformerErrorEvent 用のコンストラクタです。
getException() (10-33 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() (10-33 ページ)	XSLTransformer に発生したエラーを説明するエラー・メッセージを戻します。

XSLTransformerErrorEvent()

XSLTransformerErrorEvent 用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	このイベントを作成したオブジェクト
e	発生した例外

getException()

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()

XSLTransformer に発生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

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。

構文

```
public interface XSLTransformerErrorListener extends
     java.util.EventListener
```

xslTransformerErrorCalled()

解析エラーまたは変換エラーの発生時にコールされます。

構文

```
public void xslTransformerErrorCalled( XSLTransformerErrorEvent p0);
```

パラメータ	説明
p0	XSLTransformer が作成する XSLTransformerErrorEvent オブジェクト

XSLTransformerEvent クラス

このクラスは、XSLTransformer が使用するイベント・オブジェクトを表し、登録されたすべてのリスナーに XSL 変換イベントを通知します。

構文

```
public class XSLTransformerEvent extends java.util.EventObject
```

表 10-15 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

表 10-16 XSLTransformerEvent のメソッドの概要

メソッド	説明
XSLTransformerEvent()（10-35 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 オブジェクトを構成します。
getID()（10-36 ページ）	XSLTransformer オブジェクトの固有の ID を戻します。

XSLTransformerEvent()

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 オブジェクトを構成します。

構文

```
public  XSLTransformerEvent(java.lang.Object p0, int p1);
```

パラメータ	説明
p0	イベントを起動するソース 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()

複数の XSLTransformer インスタンスがバックグラウンドで動作中の場合に、このイベントを生成した XSLTransformer インスタンスを識別するために使用できる XSLTransformer オブジェクトの一意の ID を戻します。

構文

```
public int getID();
```

XSLTransformerListener インタフェース

非同期変換中にイベントの通知を受け取るために実装されます。addXSLTransformerListener メソッドを使用して、実装クラスを追加する必要があります。

構文

```
public interface XSLTransformerListener extends java.util.EventListener
```

xslTransformerError()

解析エラーまたは変換エラーの発生時にコールされます。

構文

```
public void xslTransformerError( XSLTransformerEvent p0);
```

パラメータ	説明
p0	XSLTransformer が作成する XSLTransformerEvent オブジェクト

xslTransformerOver()

変換の完了時にコールされます。

構文

```
public void xslTransformerOver( XSLTransformerEvent p0);
```

パラメータ	説明
p0	XSLTransformer が作成する XSLTransformerEvent オブジェクト

xslTransformerStarted()

変換の開始時にコールされます。

構文

```
public void xslTransformerStarted( XSLTransformerEvent p0);
```

パラメータ	説明
p0	XSLTransformer が作成する XSLTransformerEvent オブジェクト

oracle.xml.dbviewer パッケージ

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 バッファ間で転送できます。このパッケージは、XML へのデータベース問合せの変換に使用できます。XML ファイルは、XSL を使用して解析および変換できます。表 10-17 に、`oracle.xml.dbviewer` のクラスを示します。

表 10-17 oracle.xml.dbviewer のクラスの概要

クラス	説明
DBViewer クラス（10-40 ページ）	XSL スタイルシートを適用し、結果の HTML をビジュアル化して、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（10-65 ページ）	DBViewer Bean の情報を提供します。

DBViewer クラス

XSL スタイルシートを適用し、結果の HTML をビジュアル化して、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という 3 つのバッファがあります。Bean API を使用すると、コール側のプログラムが、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 バッファの内容は、ソースまたはツリー構造として表示されます。結果バッファの内容も、HTML としてレンダリングされ、ソースまたはツリー構造として表示できます。XML バッファは、データベース問合せからロードできます。

すべてのバッファは、Oracle データベース内のキャラクタ・ラージ・オブジェクト（Character Large Object: CLOB）表およびファイル・システムからファイルをロードおよび保存できます。そのため、データベース内のユーザー・スキーマとファイル・システム間でのファイル移動を制御できます。

構文

```
public class DBViewer extends javax.swing.JPanel implements java.io.Serializable
```

表 10-18 DBViewer のメソッドの概要

メソッド	説明
DBViewer()（10-44 ページ）	DBViewer の新しいインスタンスを構成します。
getHostname()（10-44 ページ）	データベース・ホスト名を取得します。
getInstancename()（10-44 ページ）	データベース・インスタンス名を取得します。
getPassword()（10-44 ページ）	ユーザー・パスワードを取得します。
getPort()（10-44 ページ）	データベース・ポート番号を文字列として取得します。
getResBuffer()（10-45 ページ）	結果バッファの内容を取得します。
getResCLOBFileName()（10-45 ページ）	結果の CLOB ファイル名を取得します。
getResCLOBTableName()（10-45 ページ）	結果の CLOB 表名を取得します。
getResFileName()（10-45 ページ）	結果のファイル名を取得します。
getUsername()（10-45 ページ）	ユーザー名を取得します。
getXmlBuffer()（10-46 ページ）	XML バッファの内容を取得します。

表 10-18 DBViewer のメソッドの概要（続き）

メソッド	説明
getXmlCLOBFileName()（10-46 ページ）	XML CLOB ファイル名を取得します。
getXmlCLOBTableName()（10-46 ページ）	XML CLOB 表名を取得します。
getXmlFileName()（10-46 ページ）	XML ファイル名を取得します。
getXMLStringFromSQL()（10-47 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()（10-47 ページ）	XSL バッファの内容を取得します。
getXslCLOBFileName()（10-47 ページ）	XSL CLOB ファイル名を取得します。
getXslCLOBTableName()（10-47 ページ）	XSL CLOB 表名を取得します。
getXslFileName()（10-48 ページ）	XSL ファイル名を取得します。
loadResBuffer()（10-48 ページ）	結果バッファをロードします。
loadResBufferFromClob()（10-48 ページ）	CLOB ファイルから結果バッファをロードします。
loadResBufferFromFile()（10-49 ページ）	ファイルから結果バッファをロードします。
loadXmlBuffer()（10-49 ページ）	XML バッファをロードします。
loadXmlBufferFromClob()（10-49 ページ）	CLOB ファイルから XML バッファをロードします。
loadXmlBufferFromFile()（10-50 ページ）	ファイルから XML バッファをロードします。
loadXMLBufferFromSQL()（10-50 ページ）	SQL 結果セットから XML バッファをロードします。
loadXslBuffer()（10-50 ページ）	ファイルから XSL バッファをロードします。
loadXslBufferFromClob()（10-51 ページ）	CLOB ファイルから XSL バッファをロードします。
loadXslBufferFromFile()（10-51 ページ）	ファイルから XSL バッファをロードします。
parseResBuffer()（10-51 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、XMLDocument を戻します。

表 10-18 DBViewer のメソッドの概要（続き）

メソッド	説明
parseXmlBuffer()（10-51 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、XMLDocument を戻します。
parseXslBuffer()（10-52 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、XMLDocument を戻します。
saveResBuffer()（10-52 ページ）	結果バッファをファイルに保存します。
saveResBufferToClob()（10-52 ページ）	結果バッファを CLOB ファイルに保存します。
saveResBufferToFile()（10-53 ページ）	結果バッファをファイルに保存します。
saveXmlBuffer()（10-53 ページ）	XML バッファをファイルに保存します。
saveXmlBufferToClob()（10-53 ページ）	XML バッファを CLOB ファイルに保存します。
saveXmlBufferToFile()（10-54 ページ）	XML バッファをファイルに保存します。
saveXslBuffer()（10-54 ページ）	XSL バッファをファイルに保存します。
saveXslBufferToClob()（10-54 ページ）	XSL バッファを CLOB ファイルに保存します。
saveXslBufferToFile()（10-55 ページ）	XSL バッファをファイルに保存します。
setHostname()（10-55 ページ）	データベース・ホスト名を設定します。
setInstancename()（10-55 ページ）	データベース・インスタンス名を設定します。
setPassword()（10-56 ページ）	ユーザー・パスワードを設定します。
setPort()（10-56 ページ）	データベース・ポート番号を設定します。
setResBuffer()（10-56 ページ）	結果バッファの新しいテキストを設定します。
setResCLOBFileName()（10-57 ページ）	結果の CLOB ファイル名を設定します。
setResCLOBTableName()（10-57 ページ）	結果の CLOB 表名を設定します。
setResFileName()（10-57 ページ）	結果のファイル名を設定します。
setResHtmlView()（10-58 ページ）	レンダリングされた HTML として結果バッファを表示します。

表 10-18 DBViewer のメソッドの概要（続き）

メソッド	説明
setResSourceEditView()（10-58 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()（10-58 ページ）	結果バッファを XML ソースとして表示します。
setResTreeView()（10-59 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()（10-59 ページ）	ユーザー名を設定します。
setXmlBuffer()（10-59 ページ）	XML バッファの新しいテキストを設定します。
setXmlCLOBFileName()（10-60 ページ）	XML CLOB ファイル名を設定します。
setXmlCLOBTableName()（10-60 ページ）	XML CLOB 表名を設定します。
setXmlFileName()（10-60 ページ）	XML ファイル名を設定します。
setXmlSourceEditView()（10-61 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()（10-61 ページ）	XML バッファを XML ソースとして表示します。
setXmlTreeView()（10-61 ページ）	XML バッファをツリーとして表示します。
setXslBuffer()（10-62 ページ）	XSL バッファの新しいテキストを設定します。
setXslCLOBFileName()（10-62 ページ）	XSL CLOB ファイル名を設定します。
setXslCLOBTableName()（10-62 ページ）	XSL CLOB 表名を設定します。
setXslFileName()（10-63 ページ）	XSL ファイル名を設定します。
setXslSourceEditView()（10-63 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()（10-63 ページ）	XSL バッファを XML ソースとして表示します。
setXslTreeView()（10-64 ページ）	XSL バッファをツリーとして表示します。
transformToDoc()（10-64 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()（10-64 ページ）	XSL バッファのスタイルシート変換を XML バッファの XML に適用し、その結果を結果バッファに格納します。
transformToString()（10-64 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()

DBViewer の新しいインスタンスを構成します。

構文

```
public  DBViewer();
```

getHostname()

データベース・ホスト名を取得します。

構文

```
public java.lang.String getHostname();
```

getInstancename()

データベース・インスタンス名を取得します。

構文

```
public java.lang.String getInstancename();
```

getPassword()

ユーザー・パスワードを取得します。

構文

```
public java.lang.String getPassword();
```

getPort()

データベース・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()

結果バッファの内容を取得します。

構文

```
public java.lang.String getResBuffer();
```

getResCLOBFileName()

結果の CLOB ファイル名を取得します。

構文

```
public java.lang.String getResCLOBFileName();
```

getResCLOBTableName()

結果の CLOB 表名を取得します。

構文

```
public java.lang.String getResCLOBTableName();
```

getResFileName()

結果のファイル名を取得します。

構文

```
public java.lang.String getResFileName();
```

getUsername()

ユーザー名を取得します。

構文

```
public java.lang.String getUsername();
```

getXmlBuffer()

XML バッファの内容を取得します。

構文

```
public java.lang.String getXmlBuffer();
```

getXmlCLOBFileName()

XML CLOB ファイル名を取得します。

構文

```
public java.lang.String getXmlCLOBFileName();
```

getXmlCLOBTableName()

XML CLOB 表名を取得します。

構文

```
public java.lang.String getXmlCLOBTableName();
```

getXmlFileName()

XML ファイル名を取得します。

構文

```
public java.lang.String getXmlFileName();
```

getXMLStringFromSQL()

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()

XSL バッファの内容を取得します。

構文

```
public java.lang.String getXslBuffer();
```

getXslCLOBFileName()

XSL CLOB ファイル名を取得します。

構文

```
public java.lang.String getXslCLOBFileName();
```

getXslCLOBTableName()

XSL CLOB 表名を取得します。

構文

```
public java.lang.String getXslCLOBTableName();
```

getXslFileName()

XSL ファイル名を取得します。

構文

```
public java.lang.String getXslFileName();
```

loadResBuffer()

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);`	XMLDocument 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 表名
`clobFilename`	CLOB ファイル名
`resDoc`	XMLDocument

loadResBufferFromClob()

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()

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()

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);`	XMLDocument 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 表名
`clobFilename`	CLOB ファイル名
`resDoc`	XMLDocument

loadXmlBufferFromClob()

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()

ファイルから XML バッファをロードします。

構文

```
public void loadXmlBufferFromFile();
```

loadXMLBufferFromSQL()

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()

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);	XMLDocument 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 表名

パラメータ	説明
clobFilename	CLOB ファイル名
xslDoc	XMLDocument

loadXslBufferFromClob()

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()

ファイルから XSL バッファをロードします。

構文

```
public void loadXslBufferFromFile();
```

parseResBuffer()

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、XMLDocument を戻します。

構文

```
public oracle.xml.parser.v2.XMLDocument parseResBuffer();
```

parseXmlBuffer()

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、XMLDocument を戻します。

構文

```
public oracle.xml.parser.v2.XMLDocument parseXmlBuffer();
```

parseXslBuffer()

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、XMLDocument を戻します。

構文

```
public oracle.xml.parser.v2.XMLDocument parseXslBuffer();
```

saveResBuffer()

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 表名
`filename`	CLOB ファイル名

saveResBufferToClob()

結果バッファを CLOB ファイルに保存します。

構文

```
public void saveResBufferToClob();
```

saveResBufferToFile()

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 表名
`filename`	CLOB ファイル名

saveXmlBufferToClob()

XML バッファを CLOB ファイルに保存します。

構文

```
public void saveXmlBufferToClob();
```

saveXmlBufferToFile()

XML バッファをファイルに保存します。

構文

```
public void saveXmlBufferToFile();
```

saveXslBuffer()

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・システム内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()

XSL バッファを CLOB ファイルに保存します。

構文

```
public void saveXslBufferToClob();
```

saveXslBufferToFile()

XSL バッファをファイルに保存します。

構文

```
public void saveXslBufferToFile();
```

setHostname()

データベース・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()

データベース・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・インスタンス名

setPassword()

ユーザー・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・パスワード

setPort()

データベース・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()

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()

結果の CLOB 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 ファイル名

setResCLOBTableName()

結果の CLOB 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 表名

setResFileName()

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()

レンダリングされた HTML として結果バッファを表示します。

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決定します。表示する場合は true、表示しない場合は false に設定します。

setResSourceEditView()

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決定します。表示する場合は true に設定します。

setResSourceView()

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決定します。表示する場合は true、表示しない場合は false に設定します。

setResTreeView()

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 ツリーとして表示するかどうかを決定します。表示する場合は true、表示しない場合は false に設定します。

setUsername()

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()

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()

XML CLOB 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 CLOB ファイル名

setXmlCLOBTableName()

XML CLOB 表名を設定します。

構文

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

パラメータ	説明
name	XML CLOB 表名

setXmlFileName()

XML 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 ファイル名

setXmlSourceEditView()

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決定します。表示する場合は true に設定します。

setXmlSourceView()

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決定します。表示する場合は true、表示しない場合は false に設定します。

setXmlTreeView()

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決定します。表示する場合は true、表示しない場合は false に設定します。

setXslBuffer()

XSL バッファの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()

XSL CLOB 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 CLOB ファイル名

setXslCLOBTableName()

XSL CLOB 表名を設定します。

構文

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

パラメータ	説明
name	XSL CLOB 表名

setXslFileName()

XSL 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 ファイル名

setXslSourceEditView()

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決定します。表示する場合は true に設定します。

setXslSourceView()

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決定します。表示する場合は true、表示しない場合は false に設定します。

setXslTreeView()

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 XML ツリーとして表示するかどうかを決定します。表示する場合は true、表示しない場合は false に設定します。

transformToDoc()

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()

XSL バッファのスタイルシート変換を XML バッファの XML に適用し、その結果を結果バッファに格納します。

構文

```
public void transformToRes();
```

transformToString()

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

DBViewer Bean の情報を提供します。

構文

```
public class DBViewerBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
```

表 10-19 DBViewerBeanInfo のメソッドの概要

メソッド	説明
DBViewerBeanInfo()（10-65 ページ）	クラス・コンストラクタです。
getIcon()（10-65 ページ）	要求されたアイコンの種類のイメージを戻します。
getPropertyDescriptors()（10-66 ページ）	DBViwer Bean の編集可能な PropertyDescriptors の配列を取得します。

DBViewerBeanInfo()

クラス・コンストラクタです。

構文

```
public  DBViewerBeanInfo();
```

getIcon()

ツールバーやツールボックスなどで DBViewer Bean 用に要求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	要求されたアイコンの種類

getPropertyDescriptors()

DBViwer Bean の編集可能な PropertyDescriptors の配列を取得します。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

oracle.xml.srcviewer パッケージ

構文がハイライトされた XML ソース文書の表示に使用される可視 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 10-20 に、oracle.xml.srcviewer のクラスを示します。

表 10-20 oracle.xml.srcviewer のクラスの概要

クラス	説明
XMLSourceView クラス （10-68 ページ）	XML 文書を表示します。
XMLSourceViewBeanInfo クラス （10-86 ページ）	XMLSourceView Bean の情報を提供します。

XMLSourceView クラス

XML 文書を表示します。XML トークン型（タグ、属性名、属性値、コメント、CDATA、PCDATA、処理命令データ、処理命令名および表記法）を認識します。各トークン型には、フォアグラウンド・カラーおよびフォントがあります。デフォルトのカラーおよびフォント設定は、ユーザーが変更できます。入力として org.w3c.dom.Document オブジェクトを受け取ります。

構文

```
public class XMLSourceView extends javax.swing.JPanel implements
java.io.Serializable
```

表 10-21 ElementDecl のフィールド

フィールド	構文	説明
inputDOMDocument	protected org.w3c.dom.Document inputDOMDocument	表示する XML 文書です。
jScrollPane	protected javax.swing.JScrollPane jScrollPane	スクロールを可能にするために XMLSourveView が使用する Java Swing コンポーネントです。
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 コンポーネントです。
xmlStyledDocument	protected oracle.xml.srcviewer.XMLStyledDocument xmlStyledDocument	定型化された XML 文書を表し、XML トークンをフォントやカラーなどの属性に対応付けます。

表 10-22 XMLSourceView のメソッドの概要

メソッド	説明
XMLSourceView()（10-71 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()（10-71 ページ）	与えられた属性セットからフォントを戻します。
fontSet()（10-72 ページ）	可変属性セットのフォントを設定します。

表 10-22 XMLSourceView のメソッドの概要（続き）

メソッド	説明
getAttributeNameFont()（10-72 ページ）	属性値のフォントを戻します。
getAttributeNameForeground()（10-72 ページ）	属性名のフォアグラウンド・カラーを戻します。
getAttributeValueFont()（10-72 ページ）	属性値のフォントを戻します。
getAttributeValueForeground()（10-73 ページ）	属性値のフォアグラウンド・カラーを戻します。
getBackground()（10-73 ページ）	バックグラウンド・カラーを戻します。
getCDATAFont()（10-73 ページ）	CDATA のフォントを戻します。
getCDATAForeground()（10-73 ページ）	CDATA のフォアグラウンド・カラーを戻します。
getCommentDataFont()（10-74 ページ）	コメント文のフォントを戻します。
getCommentDataForeground()（10-74 ページ）	コメント文のフォアグラウンド・カラーを戻します。
getEditedText()（10-74 ページ）	編集済テキストを戻します。
getJTextPane()（10-74 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()（10-75 ページ）	XMLSourceView の最小サイズを戻します。
getNodeAtOffset()（10-75 ページ）	与えられたオフセットにある XML ノードを戻します。
getPCDATAFont()（10-75 ページ）	PCDATA のフォントを戻します。
getPCDATAForeground()（10-75 ページ）	PCDATA のフォアグラウンド・カラーを戻します。
getPIDataFont()（10-76 ページ）	処理命令データのフォントを戻します。
getPIDataForeground()（10-76 ページ）	処理命令データのフォアグラウンド・カラーを戻します。
getPINameFont()（10-76 ページ）	処理命令名のフォントを戻します。
getPINameForeground()（10-76 ページ）	処理命令データのフォアグラウンド・カラーを戻します。
getSymbolFont()（10-76 ページ）	表記法のフォントを戻します。

表 10-22 XMLSourceView のメソッドの概要（続き）

メソッド	説明
getSymbolForeground()（10-77 ページ）	表記法のフォアグラウンド・カラーを戻します。
getTagFont()（10-77 ページ）	タグのフォントを戻します。
getTagForeground()（10-77 ページ）	タグのフォアグラウンド・カラーを戻します。
getText()（10-77 ページ）	XML 文書を文字列として戻します。
isEditable()（10-77 ページ）	このオブジェクトが編集可能かどうかを示すブーリアンを戻します。
selectNodeAt()（10-78 ページ）	指定し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()（10-78 ページ）	属性名のフォントを設定します。
setAttributeNameForeground()（10-78 ページ）	属性名のフォアグラウンド・カラーを設定します。
setAttributeValueFont()（10-79 ページ）	属性値のフォントを設定します。
setAttributeValueForeground()（10-79 ページ）	属性値のフォアグラウンド・カラーを設定します。
setBackground()（10-79 ページ）	バックグラウンド・カラーを設定します。
setCDATAFont()（10-80 ページ）	CDATA のフォントを設定します。
setCDATAForeground()（10-80 ページ）	CDATA のフォアグラウンド・カラーを設定します。
setCommentDataFont()（10-80 ページ）	コメント文のフォントを設定します。
setCommentDataForeground()（10-81 ページ）	コメント文のフォアグラウンド・カラーを設定します。
setEditable()（10-81 ページ）	このオブジェクトを編集可能にするかどうかを示すブーリアンを設定します。
setPCDATAFont()（10-81 ページ）	PCDATA のフォントを設定します。
setPCDATAForeground()（10-82 ページ）	PCDATA のフォアグラウンド・カラーを設定します。
setPIDataFont()（10-82 ページ）	処理命令データのフォントを設定します。
setPIDataForeground()（10-82 ページ）	処理命令データのフォアグラウンド・カラーを設定します。

表 10-22 XMLSourceView のメソッドの概要（続き）

メソッド	説明
setPINameFont()（10-83 ページ）	処理命令名のフォントを設定します。
setPINameForeground()（10-83 ページ）	処理命令名のフォアグラウンド・カラーを設定します。
setSelectedNode()（10-83 ページ）	選択した XML ノードにカーソル位置を設定します。
setSymbolFont()（10-84 ページ）	表記法のフォントを設定します。
setSymbolForeground()（10-84 ページ）	表記法のフォアグラウンド・カラーを設定します。
setTagFont()（10-84 ページ）	タグのフォントを設定します。
setTagForeground()（10-85 ページ）	タグのフォアグラウンド・カラーを設定します。
setXMLDocument()（10-85 ページ）	XMLviewer と XML 文書を対応付けます。

XMLSourceView()

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()

与えられた属性セットからフォントを戻します。

構文

```
public static java.awt.Font fontGet(
                javax.swing.text.AttributeSet attributeSet);
```

パラメータ	説明
attributeSet	ソースになる属性セット

fontSet()

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	可変属性セット用の新しい Font

getAttributeNameFont()

属性値のフォントを戻します。

構文

```
public java.awt.Font getAttributeNameFont();
```

getAttributeNameForeground()

属性名のフォアグラウンド・カラーを戻します。

構文

```
public java.awt.Color getAttributeNameForeground();
```

getAttributeValueFont()

属性値のフォントを戻します。

構文

```
public java.awt.Font getAttributeValueFont();
```

getAttributeValueForeground()

属性値のフォアグラウンド・カラーを戻します。

構文

```
public java.awt.Color getAttributeValueForeground();
```

getBackground()

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()

CDATA のフォントを戻します。

構文

```
public java.awt.Font getCDATAFont();
```

getCDATAForeground()

CDATA のフォアグラウンド・カラーを戻します。

構文

```
public java.awt.Color getCDATAForeground();
```

getCommentDataFont()

コメント文のフォントを戻します。

構文

```
public java.awt.Font getCommentDataFont();
```

getCommentDataForeground()

コメント文のフォアグラウンド・カラーを戻します。

構文

```
public java.awt.Color getCommentDataForeground();
```

getEditedText()

編集済テキストを戻します。

構文

```
public java.lang.String getEditedText();
```

getJTextPane()

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()

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()

与えら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()

PCDATA のフォントを戻します。

構文

```
public java.awt.Font getPCDATAFont();
```

getPCDATAForeground()

PCDATA のフォアグラウンド・カラーを戻します。

構文

```
public java.awt.Color getPCDATAForeground();
```

getPIDataFont()

処理命令データのフォントを戻します。

構文

```
public java.awt.Font getPIDataFont();
```

getPIDataForeground()

処理命令データのフォアグラウンド・カラーを戻します。

構文

```
public java.awt.Color getPIDataForeground();
```

getPINameFont()

処理命令名のフォントを戻します。

構文

```
public java.awt.Font getPINameFont();
```

getPINameForeground()

処理命令データのフォアグラウンド・カラーを戻します。

構文

```
public java.awt.Color getPINameForeground();
```

getSymbolFont()

表記法のフォントを戻します。

構文

```
public java.awt.Font getSymbolFont();
```

getSymbolForeground()

表記法のフォアグラウンド・カラーを戻します。

構文

```
public java.awt.Color getSymbolForeground();
```

getTagFont()

タグのフォントを戻します。

構文

```
public java.awt.Font getTagFont();
```

getTagForeground()

タグのフォアグラウンド・カラーを戻します。

構文

```
public java.awt.Color getTagForeground();
```

getText()

XML 文書を文字列として戻します。

構文

```
public java.lang.String getText();
```

isEditable()

このオブジェクトが編集可能かどうかを示すブーリアンを戻します。

構文

```
public boolean isEditable();
```

selectNodeAt()

指定し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()

属性名の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 Font の属性名

setAttributeNameForeground()

属性名のフォアグラウンド・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 Color

setAttributeValueFont()

属性値の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 Font の属性値

setAttributeValueForeground()

属性値のフォアグラウンド・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 Color

setBackground()

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()

CDATA の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 Font

setCDATAForeground()

CDATA のフォアグラウンド・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 Color

setCommentDataFont()

コメント文の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 Font

setCommentDataForeground()

コメント文のフォアグラウンド・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 Color

setEditable()

このオブジェクトを編集可能にするかどうかを示すブーリアンを設定します。

構文

```
public void setEditable( boolean edit);
```

パラメータ	説明
edit	オブジェクトが編集可能かどうかを示すフラグ。表示されるテキストが編集可能な場合は `true`、編集できない場合は `false` に設定します。

setPCDATAFont()

PCDATA の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 Font

setPCDATAForeground()

PCDATA のフォアグラウンド・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 Color

setPIDataFont()

処理命令データの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 Font

setPIDataForeground()

処理命令データのフォアグラウンド・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 Color

setPINameFont()

処理命令名の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 Font

setPINameForeground()

処理命令名のフォアグラウンド・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 Color

setSelectedNode()

選択した 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	選択したノード

setSymbolFont()

表記法の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 Font

setSymbolForeground()

表記法のフォアグラウンド・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 Color

setTagFont()

タグの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 Font

setTagForeground()

タグのフォアグラウンド・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 Color

setXMLDocument()

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

XMLSourceView Bean の情報を提供します。

構文

```
public class XMLSourceViewBeanInfo extends java.beans.SimpleBeanInfo
```

表 10-23 XMLSourceViewBeanInfo のメソッドの概要

メソッド	説明
XMLSourceViewBeanInfo()（10-86 ページ）	クラス・コンストラクタです。
getIcon()（10-87 ページ）	ツールバーやツールボックスなどで XMLSourceView Bean 用に要求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()（10-87 ページ）	XMLSourveView Bean の編集可能な PropertyDescriptors の配列を取得します。

XMLSourceViewBeanInfo()

クラス・コンストラクタです。

構文

```
public  XMLSourceViewBeanInfo();
```

getIcon()

ツールバーやツールボックスなどで XMLSourceView Bean 用に要求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	要求されたアイコンの種類

getPropertyDescriptors()

XMLSourveView Bean の編集可能な PropertyDescriptors 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

これは可視 Bean です。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 つの列があります。ファイル名を保持する文字列型の列と、ファイル・データを保持する CLOB 型の列です。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 10-24 に、`oracle.xml.transviewer` のクラスを示します。

表 10-24 oracle.xml.transviewer のクラスの概要

クラス	説明
DBAccess クラス（10-89 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できる CLOB 表をメンテナンスします。
DBAccessBeanInfo クラス（10-99 ページ）	DBAccess Bean の情報を提供します。
XMLTransformPanel クラス（10-101 ページ）	XML 文書に XSL 変換を適用し、結果をビジュアル化します。
XMLTransformPanelBeanInfo クラス（10-102 ページ）	XMLTransformPanel Bean の情報を提供します。
XMLTransViewer クラス（10-104 ページ）	XMLTransformPanel を使用する単純なアプリケーションです。コマンドラインから使用できます。XML ファイルの編集と解析、XSL 変換の編集と適用、XML ファイル、XSL ファイル、ファイル・システム内またはデータベース内の結果ファイルの取出しおよび保存を行います。

DBAccess クラス

複数の XML 文書およびテキスト・ドキュメントを格納できる CLOB 表をメンテナンスします。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）文書は、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、更新または削除するための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、CLOB オブジェクトである FILEDATA フィールドに格納されます。
- これらの CLOB 表は、Transviewer Bean によって自動的にメンテナンスされます。
- このクラスでメンテナンスされる CLOB 表は、Transviewer Bean で使用できます。
- クラスは、CLOB 表の作成と削除、CLOB 表の内容のリスト、およびこれらの CLOB 表内のテキスト・ドキュメントの追加、置換または削除を行います。

構文

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

表 10-25 DBAcceess のメソッドの概要

メソッド	説明
DBAccess()（10-90 ページ）	クラス・コンストラクタです。
createBLOBTable()（10-91 ページ）	BLOB 表を作成します。成功した場合は true を戻します。
createXMLTable()（10-91 ページ）	XML 表を作成します。成功した場合は true を戻します。
deleteBLOBName()（10-92 ページ）	BLOB 表からバイナリ・ファイルを削除します。成功した場合は true を戻します。
deleteXMLName()（10-92 ページ）	XML 表からファイルを削除します。成功した場合は true を戻します。
dropBLOBTable()（10-93 ページ）	BLOB 表を削除します。成功した場合は true を戻します。
dropXMLTable()（10-93 ページ）	XML 表を削除します。成功した場合は true を戻します。
getBLOBData()（10-94 ページ）	BLOB 表からバイナリ・ファイルをバイト配列として取り出します。成功した場合は true を戻します。

表 10-25 DBAcceess のメソッドの概要（続き）

メソッド	説明
getNameSize()（10-94 ページ）	ファイル名が保持されているフィールドのサイズを戻します。
getXMLData()（10-94 ページ）	XML 表からテキスト・ファイルを文字列として取り出します。
getXMLNames()（10-95 ページ）	XML 表にあるすべてのファイル名を文字列の配列として戻します。
getXMLTableNames()（10-95 ページ）	ユーザーが指定した文字列で始まるすべての XML 表名の配列を取得します。
insertBLOBData()（10-96 ページ）	バイナリ・ファイルを BLOB 表に行として挿入します。成功した場合は `true` を戻します。
insertXMLData()（10-96 ページ）	テキスト・ファイルを XML 表に行として挿入します。成功した場合は `true` を戻します。
isXMLTable()（10-97 ページ）	表が XML 表であるかどうかを確認します。成功した場合は `true` を戻します。
replaceXMLData()（10-97 ページ）	テキスト・ファイルを XML 表の行に置き換えます。成功した場合は `true` を戻します。
xmlTableExists()（10-98 ページ）	XML 表が存在するかどうかを確認します。成功した場合は `true` を戻します。

DBAccess()

クラス・コンストラクタです。

構文

```
public  DBAccess();
```

createBLOBTable()

BLOB 表を作成します。成功した場合は 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()

XML 表を作成します。成功した場合は 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()

BLOB 表からバイナリ・ファイルを削除します。成功した場合は 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()

XML 表からファイルを削除します。成功した場合は 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()

BLOB 表を削除します。成功した場合は 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()

XML 表を削除します。成功した場合は 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()

BLOB 表からバイナリ・ファイルをバイト配列として取り出します。成功した場合は 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()

ファイル名が保持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()

XML 表からテキスト・ファイルを文字列として取り出します。

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()

XML 表にある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()

ユーザーが指定した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	XML 表名の取得した配列を開始する表の接頭辞文字列

insertBLOBData()

バイナリ・ファイルを BLOB 表に行として挿入します。成功した場合は 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()

テキスト・ファイルを XML 表に行として挿入します。成功した場合は 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()

表が XML 表であるかどうかを確認します。成功した場合は 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()

テキスト・ファイルを XML 表の行に置き換えます。成功した場合は 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()

XML 表が存在するかどうかを確認します。成功した場合は 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

DBAccess Bean の情報を提供します。

構文

```
public class DBAccessBeanInfo extends java.beans.SimpleBeanInfo
```

表 10-26 DBAccessBeanInfo のメソッドの概要

メソッド	説明
DBAccessBeanInfo() (10-99 ページ)	クラス・コンストラクタです。
getIcon() (10-99 ページ)	要求されたアイコンを表すイメージ・オブジェクトを取得します。
getPropertyDescriptors() (10-100 ページ)	編集可能な PropertyDescriptors の配列を取得します。

DBAccessBeanInfo()

クラス・コンストラクタです。

構文

```
public  DBAccessBeanInfo();
```

getIcon()

ツールバーやツールボックスなどで DBAcceess Bean 用に要求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	要求されたアイコンの種類

getPropertyDescriptors()

DBAccess Bean の編集可能な PropertyDescriptors の配列を取得します。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

XMLTransformPanel クラス

XMLTransformPanel は可視 Bean です。XML 文書に XSL 変換を適用し、結果をビジュアル化します。入力 XML および XSL ドキュメント / ファイルの編集を可能にします。

構文

```
public class XMLTransformPanel extends javax.swing.JPanel
```

XMLTransformPanel()

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

XMLTransformPanel Bean の情報を提供します。

構文

```
public class XMLTransformPanelBeanInfo extends java.beans.SimpleBeanInfo
```

表 10-27 XMLTransformPanelBeanInfo のメソッドの概要

メソッド	説明
XMLTransformPanelBeanInfo() (10-102 ページ)	クラス・コンストラクタです。
getIcon() (10-102 ページ)	XMLTransformPanel Bean 用に要求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() (10-103 ページ)	XMLTransformPanel Bean の編集可能な PropertyDescriptors の配列を取得します。

XMLTransformPanelBeanInfo()

クラス・コンストラクタです。

構文

```
public  XMLTransformPanelBeanInfo();
```

getIcon()

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	要求されたアイコンの種類

getPropertyDescriptors()

XMLTransformPanel Bean の編集可能な PropertyDescriptors の配列を取得します。

構文

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors();
```

XMLTransViewer クラス

XMLTransformPanel を使用する単純なアプリケーションです。コマンドラインから使用できます。XML ファイルの編集と解析、XSL 変換の編集と適用、XML ファイル、XSL ファイル、ファイル・システム内またはデータベース内の結果ファイルの取出しおよび保存を行います。

構文

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

表 10-28 XMLTransViewer のメソッドの概要

メソッド	説明
XMLTransViewer()（10-104 ページ）	クラス・コンストラクタです。
getReleaseVersion()（10-104 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()（10-105 ページ）	新しい XMLTransViewer を開始します。

XMLTransViewer()

クラス・コンストラクタです。

構文

```
public  XMLTransViewer();
```

getReleaseVersion()

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()

新しい XMLTransViewer を開始するメイン・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	XMLTransViewer インスタンスの引数

oracle.xml.treeviewer パッケージ

これは可視 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、ノードを閉じたり開いたりできます。

表 10-29 に、oracle.xml.treeviewer のクラスを示します。

表 10-29 oracle.xml.treeviewer のクラスの概要

クラス	説明
XMLTreeView クラス（10-107 ページ）	XML 文書をツリーとして表示します。
XMLTreeViewBeanInfo クラス（10-110 ページ）	XMLTreeView Bean の情報を提供します。

XMLTreeView クラス

XML 文書をツリーとして表示します。XML DOM ノード（タグ、属性名、属性値、コメント、CDATA、PCDATA、処理命令データ、処理命令名および表記法）を認識します。入力として org.w3c.dom.Document オブジェクトを受け取ります。

構文

```
public class XMLTreeView extends javax.swing.JPanel
```

表 10-30 XMLTreeView のフィールド

フィールド	構文	説明
model	protected oracle.xml.treeviewer.XMLTreeModel model	DOM ノードから作成された JTree のデータ・モデルです。
scrollPane	protected transient javax.swing.JScrollPane scrollPane	ツリーを表示するコンテナです。水平方向および垂直方向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 コンポーネントです。

表 10-31 XMLTreeView のメソッドの概要

メソッド	説明
XMLTreeView()（10-108 ページ）	クラス・コンストラクタです。
getPreferredSize()（10-108 ページ）	XMLTreeView の推奨サイズを戻します。
getTree()（10-108 ページ）	可視ツリーを JTree として戻します。
getXMLTreeModel()（10-108 ページ）	JTree のデータ・モデルを XMLTreeModel として戻します。
setXMLDocument()（10-109 ページ）	XMLTreeViewer と XML 文書を対応付けます。
updateUI()（10-109 ページ）	XMLTreeView 画面を再描画します。

XMLTreeView()

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()

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()

可視ツリーを JTree として戻します。

構文

```
protected javax.swing.JTree getTree();
```

getXMLTreeModel()

JTree 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()

XMLTreeViewer 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
doc	表示するドキュメント

updateUI()

XMLTreeView 画面を再描画します。

構文

```
public void updateUI();
```

XMLTreeViewBeanInfo クラス

XMLTreeView Bean の情報を提供します。

構文

```
public class XMLTreeViewBeanInfo extends java.beans.SimpleBeanInfo
```

表 10-32 XMLTreeViewBeanInfo のメソッドの概要

メソッド	説明
XMLTreeViewBeanInfo() (10-110 ページ)	クラス・コンストラクタです。
getIcon() (10-110 ページ)	XMLTreeView 用に要求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() (10-111 ページ)	XMLTreeView Bean の編集可能な PropertyDescriptors の配列を取得します。

XMLTreeViewBeanInfo()

クラス・コンストラクタです。

構文

```
public  XMLTreeViewBeanInfo();
```

getIcon()

XMLTreeView 用に要求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	要求されたアイコンの種類

getPropertyDescriptors()

XMLTreeView Bean の編集可能な PropertyDescriptors 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 は、可視デモを含む不可視 Bean です。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 番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表示できます。

表 10-33 に、oracle.xml.differ のクラスを示します。

表 10-33 oracle.xml.differ のクラスの概要

クラス	説明
XMLDiff クラス（10-113 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（10-124 ページ）	XMLDiff Bean の情報を提供します。

XMLDiff クラス

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。このクラスを使用して、2 つの XML ファイルを比較し、等価かどうかを確認できます。差異（存在する場合）を表示するオブジェクトを図表の形式で表します。差異を XSL として表すこともできます。差異を含む対応する 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 番目の XML ファイルに変換できます。

構文

```
oracle.xml.differ
```

表 10-34 XMLDiff のメソッドの概要

メソッド	説明
XMLDiff()（10-115 ページ）	クラス・コンストラクタです。
setFiles()（10-115 ページ）	比較する XML ファイルを設定します。
setDocuments()（10-116 ページ）	比較する XML 文書を設定します。
setInput1()（10-116 ページ）	比較する最初の XML ファイルまたは文書を設定します。
setInput2()（10-117 ページ）	比較する 2 番目の XML ファイルまたは文書を設定します。
getDocument1()（10-117 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。
getDocument2()（10-118 ページ）	ドキュメント・ルートを 2 番目の XML ツリーの XMLDocument オブジェクトとして取得します。
diff()（10-118 ページ）	2 つの XML ファイルまたは 2 つの XMLDocument オブジェクト間の差異を検索します。可視テキスト・パネルを JTextPane オブジェクトとして取得します。JTextPane オブジェクトは、最初の XML ファイルまたは XML 文書の差異を視覚的に表示します。
getDiffPane1()（10-118 ページ）	可視テキスト・パネルを JTextPane オブジェクトとして取得します。JTextPane オブジェクトは、最初の XML ファイルまたは XML 文書の差異を視覚的に表示します。
getDiffPane2()（10-118 ページ）	可視テキスト・パネルを JTextPane オブジェクトとして取得します。JTextPane オブジェクトは、2 番目の XML ファイルまたは XML 文書の差異を視覚的に表示します。

表 10-34 XMLDiff のメソッドの概要（続き）

メソッド	説明
setIndentIncr()（10-119 ページ）	XSL 生成のインデントを設定します。
setNewNodeIndentIncr()（10-119 ページ）	ユーザー定義のファイル名を使用して、XSL ファイルを生成します。ファイル名は、最初に設定された 2 つの XML ファイル間の差異を表します。
generateXSLFile()（10-120 ページ）	ユーザー定義のファイル名を使用して、XSL ファイルを生成します。ファイル名は、最初に設定された 2 つの XML ファイル間の差異を表します。
generateXSLDOC()（10-120 ページ）	XSL スタイルシートを XMLDocument として生成します。ファイル名は、最初に設定された 2 つの XML 文書間の差異を表します。
equals()（10-121 ページ）	2 つのノードを比較します。差分取得アルゴリズムによってコールされます。このファンクションは、必要に応じて、比較をカスタマイズするために上書きされます。
domBuilderErrorCalled()（10-121 ページ）	解析中にエラーが発生した場合、DOM パーサーによってのみコールされる DOMBuilderErrorListener インタフェースを実装するメソッドです。
domBuilderError()（10-122 ページ）	DOM パーサーによってのみコールされる DOMBuilderErrorListener インタフェースを実装するメソッドです。
domBuilderOver()（10-122 ページ）	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る DOMBuilderListener インタフェースを実装するメソッドです。
domBuilderStarted()（10-122 ページ）	解析の開始時に、DOM パーサーによってのみコールされる DOMBuilderListener インタフェースを実装するメソッドです。
printDiffTree()（10-123 ページ）	アルゴリズムによって差異として識別されたノード名および値を含む差異ツリーを出力します。デバッグに役立ちます。
setNoMoves()（10-123 ページ）	差分取得アルゴリズムによって検出される移動がないことを前提とします。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。これにより、パフォーマンスが向上します。

XMLDiff()

クラス・コンストラクタです。

構文

```
public XMLDiff();
```

setFiles()

比較する XML ファイルを設定します。2 つのファイルは、比較のために DOM ツリーに解析されます。これは、setInput1() および setInput2() をコールするより高速です。次の例外が発生します。

- `java.io.IOException` - I/O エラーの場合に発生します。
- `XMLParseException` - XML 文書を解析する場合に発生します。
- `SAXException` - XML 文書を解析する場合に発生します。
- `java.lang.InterruptedException` - スリープ中のスレッドに割込みがあった場合に発生します。

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 番目の XML ファイル

setDocuments()

比較する 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

setInput1()

比較する最初の XML ファイルまたは文書を設定します。入力ファイルは、比較のために DOM ツリーに解析されます。次の例外が発生します。

- java.io.IOException - I/O エラーの場合に発生します。
- XMLParseException - XML 文書を解析する場合に発生します。
- SAXException - XML 文書を解析する場合に発生します。
- java.lang.InterruptedException - スリープ中のスレッドに割込みがあった場合に発生します。

次の表に、オプションを示します。

構文	説明
public void setInput1(File file1);	比較する最初の XML ファイルを設定します。
public void setInput1(XMLDocument doc1);	比較する最初の 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()

比較する 2 番目の XML ファイルまたは文書を設定します。入力ファイルは、比較のために DOM ツリーに解析されます。次の例外が発生します。

- `java.io.IOException` - I/O エラーの場合に発生します。
- `XMLParseException` - XML 文書を解析する場合に発生します。
- `SAXException` - XML 文書を解析する場合に発生します。
- `java.lang.InterruptedException` - スリープ中のスレッドに割込みがあった場合に発生します。

次の表に、オプションを示します。

構文	説明
`public void setInput2( File file2);`	比較する 2 番目の XML ファイルを設定します。
`public void setInput2( XMLDocument doc2);`	比較する 2 番目の XML 文書を設定します。

パラメータ	説明
`file2`	2 番目の XML ファイル
`doc2`	2 番目の XML 文書

getDocument1()

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()

ドキュメント・ルートを 2 番目の 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()

2 つの XML ファイルまたは 2 つの XMLDocument オブジェクト間の差異を検索します。XML ファイルまたは文書が同じ場合は `true`、異なる場合は `false` を戻します。XML ファイルが正常に解析されなかった場合、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()

可視テキスト・パネルを JTextPane オブジェクトとして取得します。JTextPane オブジェクトは、最初の XML ファイルまたは XML 文書の差異を視覚的に表示します。

構文

```
public javax.swing.JTextPane getDiffPane1();
```

getDiffPane2()

可視テキスト・パネルを JTextPane オブジェクトとして取得します。JTextPane オブジェクトは、2 番目の XML ファイルまたは XML 文書の差異を視覚的に表示します。

構文

```
public javax.swing.JTextPane getDiffPane2();
```

setIndentIncr()

XSL 生成のインデントを設定します。このファンクションは、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、すべての属性のみに適用されます。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	属性の空白数で表したインデントの増分

setNewNodeIndentIncr()

XSL 生成のインデントを設定します。このファンクションは、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。属性のインデントの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	新しいノードの空白数で表したインデントの増分

generateXSLFile()

ユーザー定義のファイル名を使用して、XSL ファイルを生成します。ファイル名は、最初に設定された 2 つの XML ファイル間の差異を表します。入力ファイル名が null の場合、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 番目の XML ファイルに変換できます。XML が同じ場合、生成された XSL は最初の XML ファイルを 2 番目の XML ファイルに変換して、2 つのファイルを等価にします。XSL ファイルが正常に作成されなかった場合は、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 ファイル名

generateXSLDOC()

XSL スタイルシートを XMLDocument として生成します。ファイル名は、最初に設定された 2 つの XML 文書間の差異を表します。入力ファイル名が null の場合、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 番目の XML ファイルに変換できます。XML が同じ場合、生成された XSL は最初の XML ファイルを 2 番目の XML ファイルに変換して、2 つのファイルを等価にします。次の例外が発生します。

- java.io.IOException - I/O エラーの場合に発生します。
- java.io.FileNotFoundException - 生成された XSL ファイルが見つからない場合に発生します。
- SAXException - XML 文書を解析する場合に発生します。
- java.lang.InterruptedException - スリープ中のスレッドに割込みがあった場合に発生します。

構文

```
public void generateXSLDoc();
```

equals()

2 つのノードを比較します。差分取得アルゴリズムによってコールされます。このファンクションは、必要に応じて、比較をカスタマイズするために上書きされます。

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 番目のノード

domBuilderErrorCalled()

解析中にエラーが発生した場合、DOM パーサーによってのみコールされる DOMBuilderErrorListener インタフェースを実装するメソッドです。DOMBuilderErrorListener インタフェースの domBuilderErrorCalled で指定します。

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()

DOM パーサーによってのみコールされる DOMBuilderErrorListener インタフェースを実装するメソッドです。DOMBuilderListener インタフェースの domBuilderError で指定します。

構文

```
public void domBuilderError( DOMBuilderEvent p0);
```

パラメータ	説明
p0	domBuilderErrorCalled() で処理されるパーサー・イベント・エラー

domBuilderOver()

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る DOMBuilderListener インタフェースを実装するメソッドです。DOMBuilderListener インタフェースの domBuilderOver で指定します。

構文

```
public void domBuilderOver ( DOMBuilderEvent p0);
```

パラメータ	説明
p0	パーサー・イベント

domBuilderStarted()

解析の開始時に、DOM パーサーによってのみコールされる DOMBuilderListener インタフェースを実装するメソッドです。DOMBuilderListener インタフェースの domBuilderStarted で指定します。

構文

```
public void domBuilderStarted( DOMBuilderEvent p0);
```

パラメータ	説明
p0	パーサー・イベント

printDiffTree()

アルゴリズムによって差異として識別されたノード名および値を含む差異ツリーを出力します。デバッグに役立ちます。XSL ファイルが正常に作成されなかった場合は、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	出力された差異ツリーを含む BufferedWriter

setNoMoves()

差分取得アルゴリズムによって検出される移動がないことを前提とします。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。これにより、パフォーマンスが向上します。

構文

```
public void setNoMoves();
```

XMLDiffBeanInfo クラス

XMLDiff Bean の情報を提供します。

構文

```
public class XMLDiffBeanInfo extends java.beans.SimpleBeanInfo
```

表 10-35 XMLDiffBeanInfo のメソッドの概要

メソッド	説明
XMLDiffBeanInfo() (10-124 ページ)	クラス・コンストラクタです。
getPropertyDescriptors() (10-124 ページ)	ツールバーやツールボックスなどで XMLDiff Bean 用に要求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() (10-125 ページ)	XMLDiff Bean の編集可能な PropertyDescriptors の配列を取得します。

XMLDiffBeanInfo()

クラス・コンストラクタです。

構文

```
public XMLDiffBeanInfo();
```

getPropertyDescriptors()

XMLDiff Bean の編集可能な PropertyDescriptors 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()

ツールバーやツールボックスなどで XMLDiff Bean 用に要求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	要求されたアイコンの種類

11

Java 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、読取り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形式で保持されます。

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。

- CXMLHandlerBase クラス
- CXMLParser クラス

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

CXMLHandlerBase クラス

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この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。

構文

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

表 11-1 CXMLHandlerBase のメソッドの概要

メソッド	説明
CXMLHandlerBase()（11-3 ページ）	新しい CXMLHandlerBase を作成します。
cDATASection()（11-4 ページ）	CDATA セクションの通知を受け取ります。
characters()（11-4 ページ）	要素内の文字データの通知を受け取ります。
comment()（11-5 ページ）	コメントの通知を受け取ります。
endDoctype()（11-5 ページ）	DTD の終わりの通知を受け取ります。
endDocument()（11-5 ページ）	ドキュメントの終わりの通知を受け取ります。
endElement()（11-6 ページ）	要素の終わりの通知を受け取ります。
endPrefixMapping()（11-6 ページ）	接頭辞マッピングの有効範囲の終わりの通知を受け取ります。
getCXMLContext()（11-6 ページ）	圧縮に使用された CXML コンテキストを戻します。
getProperty()（11-7 ページ）	プロパティの値を検索し、戻します。
ignorableWhitespace()（11-7 ページ）	要素内容内の無視できる空白の通知を受け取ります。
processingInstruction()（11-8 ページ）	処理命令の通知を受け取ります。
setDoctype()（11-8 ページ）	DTD の通知を受け取るように、DTD を設定します。
setDocumentLocator()（11-8 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。

表 11-1 CXMLHandlerBase のメソッドの概要（続き）

メソッド	説明
setError()（11-9 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()（11-9 ページ）	プロパティの値を設定します。
setTextDecl()（11-10 ページ）	テキスト宣言の通知を受け取るように、テキスト宣言を設定します。
setXMLDecl()（11-10 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()（11-11 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()（11-11 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()（11-12 ページ）	ドキュメントの始まりの通知を受け取ります。
startElement()（11-12 ページ）	要素の始まりの通知を受け取ります。
startPrefixMapping()（11-12 ページ）	URI の接頭辞マッピングの有効範囲の始まりの通知を受け取ります。

CXMLHandlerBase()

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 を構成します。
`public CXMLHandlerBase( oracle.xml.io.XMLObjectOutput out);`	XMLObjectOutputStream を使用して、CXMLHandlerBase を構成します。

パラメータ	説明
`out`	出力ストリーム

cDATASection()

CDATA セクションの通知を受け取ります。パーサーは、CDATA セクションが検出されるたびに、このメソッドをコールします。`org.xml.sax.SAXException` が発生します。これは SAX の例外であり、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 セクションの文字列
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()

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()

コメントの通知を受け取ります。パーサーは、コメントが検出されるたびにこのメソッドをコールします。コメントは、主なドキュメント要素の前後で発生する場合があることに注意してください。org.xml.sax.SAXException が発生します。これは SAX の例外であり、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()

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。これは SAX の例外であり、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()

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。これは SAX の例外であり、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()

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()

URI の接頭辞マッピングの有効範囲の終わりの通知を受け取ります。

構文

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

パラメータ	説明
prefix	マップされていた接頭辞

getCXMLContext()

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()

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()

要素内容内の無視できる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	配列のうちの読み取る文字数

processingInstruction()

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()

DTD の通知をすぐ後に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() をコールしてからこのメソッドをコールし、使用する DTD を登録します。org.xml.sax.SAXException が発生します。これは SAX の例外であり、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()

ドキュメント・イベント用の Locator オブジェクトの通知をすぐ後に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()

XMLError ハンドラの通知をすぐ後に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。これは SAX の例外であり、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()

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()

テキスト宣言の通知をすぐ後に受け取ることができるように、テキスト宣言を登録します。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。これは SAX の例外であり、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()

XML 宣言の通知をすぐ後に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。これは SAX の例外であり、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()

XMLSchema オブジェクトの通知をすぐ後に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。これは SAX の例外であり、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()

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

構文

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、このメソッドをコールします。検証を行わないプロセッサは、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。これは SAX の例外であり、別の例外が隠されている可能性があります。

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()

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()

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()

URI の接頭辞マッピングの有効範囲の始まりの通知を受け取ります。

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

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。

構文

```
public class CXMLParser
```

表 11-2 CXMLHandlerBase のメソッドの概要

メソッド	説明
startPrefixMapping()（11-12 ページ）	圧縮ストリーム読取りのための新しい XML パーサー・オブジェクトを作成します。
CXMLParser()（11-13 ページ）	コンテンツ・ハンドラを取得します。
getContentHandler()（11-14 ページ）	現行のエラー・ハンドラを取得します。
getErrorHandler()（11-14 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()（11-14 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()（11-15 ページ）	コンテンツ・イベント・ハンドラを登録します。
setErrorHandler()（11-15 ページ）	エラー・イベント・ハンドラを登録します。

CXMLParser()

圧縮ストリーム読取りのための新しい XML パーサー・オブジェクトを作成します。

構文

```
public  CXMLParser();
```

getContentHandler()

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()

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()

圧縮された SAXException を解析します。

- IOException - I/O 操作によるエラー

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()

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()

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

12

Simple Object Access Protocol（SOAP）

OracleAS SOAP は、Simple Object Access Protocol の実装です。OracleAS 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 メッセージ形式を提供します。SOAP は、World Wide Web Consortium（W3C）の XML Schema 標準を使用します。

OracleAS SOAP API は、次のパッケージに含まれています。

- oracle.soap.server パッケージ
- oracle.soap.transport パッケージ
- oracle.soap.transport.http パッケージ
- oracle.soap.util.xml パッケージ

関連項目：

- `http://www.w3.org/TR/SOAP/`
- `http://xml.apache.org/soap/`
- 『Oracle アプリケーション開発者ガイド - XML』

oracle.soap.server パッケージ

oracle.soap.server 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 12-1 に、XDK for Java で OracleAS 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 12-1 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（12-3 ページ）	SOAP サーバーの交換可能なハンドラ用のインタフェースを定義します。
Provider インタフェース（12-7 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（12-10 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（12-14 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（12-18 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（12-22 ページ）	ログ出力の実装によってサポートされる必要がある機能を定義します。ログ出力は、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（12-27 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（12-32 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（12-40 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（12-55 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 は、SOAP サーバーの交換可能なハンドラ用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、引数なしのコンストラクタを提供し、スレッド・セーフである必要があります。

構文

```
public interface Handler
```

表 12-2 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。

表 12-3 Handler のメソッドの概要

メソッド	説明
destroy()（12-4 ページ）	ハンドラをクリーンアップします（1 回のみ）。
getName()（12-4 ページ）	このハンドラの名前を戻します。
getOptions()（12-4 ページ）	ハンドラ実装固有のオプションを戻します。
init()（12-4 ページ）	ハンドラを初期化します（1 回のみ）。
invoke()（12-5 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()（12-5 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()（12-6 ページ）	後で init により使用されるハンドラのオプションを設定します。

destroy()

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()

このハンドラの名前を戻します。

構文

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

getOptions()

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()

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グ出力を含む SOAP サーバー・コンテキスト

invoke()

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。対照的に、エラー連鎖内のすべてのハンドラは、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 chainType 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()

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()

後で init により使用され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

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合せ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、引数なしのコンストラクタを提供し、スレッド・セーフである必要があります。

構文

```
public interface Provider
```

表 12-4 Provider のメソッドの概要

メソッド	説明
destroy()（12-8 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()（12-8 ページ）	この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()（12-8 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()（12-9 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()

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()

この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()

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グ出力を含む SOAP サーバー・コンテキスト

invoke()

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	リクエストの処理に使用される情報を含む RequestContext

ProviderManager インタフェース

プロバイダ・マネージャは、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・マネージャでは、リクエストが受け入れられるためにリクエストを行う必要がある URL を設定できます。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。

構文

```
public interface ProviderManager
```

表 12-5 ProviderManager のメソッドの概要

メソッド	説明
deploy()（12-11 ページ）	指定されたプロバイダをデプロイします。
destroy()（12-11 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()（12-11 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()（12-12 ページ）	プロバイダ・マネージャを初期化します。
list()（12-12 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()（12-12 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()（12-13 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()（12-13 ページ）	指定されたプロバイダをアンデプロイします。

deploy()

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()

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()

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()

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()

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()

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()

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
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

undeploy()

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

サービス・マネージャは、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・マネージャでは、リクエストが受け入れられるためにリクエストを行う必要がある URL を設定できます。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。

構文

```
public interface ServiceManager
```

表 12-6 ServiceManager のメソッドの概要

メソッド	説明
getRequiredRequestURI()（12-15 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()（12-15 ページ）	指定されたサービスをデプロイします。
destroy()（12-15 ページ）	サービス・マネージャをクリーンアップします。
init()（12-16 ページ）	サービス・マネージャを初期化します。
list()（12-16 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、サービス ID の配列を戻します。
query()（12-17 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()（12-17 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()

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()

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()

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()

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()

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、サービス 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()

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()

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 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。実際の内容は、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。このクラスには、コンテナ固有の内容のみが含まれます。

構文

```
public class ContainerContext extends Object
```

表 12-7 ContainerContext のフィールド

フィールド	構文	説明
SERVLET_CONTAINER	public static final String SERVLET_CONTAINER	サーブレット・コンテナ・タイプの値

表 12-8 ContainerContext のメソッドの概要

メソッド	説明
ContainerContext()（12-19 ページ）	クラス・コンストラクタです。
getAttribute()（12-19 ページ）	指定された名前を持つ属性を戻します。
getAttributeNames()（12-19 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()（12-19 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()（12-20 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。
removeAttribute()（12-20 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()（12-20 ページ）	この 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()（12-21 ページ）	コンテナ・タイプを設定します。
setHttpServlet()（12-21 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()

クラス・コンストラクタです。

構文

```
public ContainerContext();
```

getAttribute()

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()

この SOAP コンテキスト内で使用可能な属性名を含む Enumeration を戻します。

構文

```
public Enumeration getAttributeNames();
```

getContainerType()

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()

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()

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()

この 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()

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()

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 HttpServlet

Logger クラス

Logger は、ログ出力の実装によりサポートされる必要がある機能を定義します。ログ出力は、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度が指定されます。指定された重大度以上である場合、情報を記録する必要があります。

重大度を低い順に示すと、次のようになります。

- SEVERITY_ERROR
- SEVERITY_STATUS
- SEVERITY_DEBUG

たとえば、重大度が SEVERITY_STATUS に設定されていると、重大度が 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。

構文

```
public abstract class Logger extends Object
```

表 12-9 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度レベル。
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度レベル。
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度レベル。
SEVERITY_INVALID	protected static final int SEVERITY_INVALID	無効な重大度設定を示します。
SEVERITY_NAMES	public static String SEVERITY_NAMES[]	重大度レベルごとに出力可能な名前。重大度により索引付けされています。
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、デフォルトの重大度レベルの設定。デフォルトは SEVERITY_STATUS です。
OPTION_SEVERITY	public static final String OPTION_SEVERITY	ログ出力の重大度を指定する構成オプション。
m_severity	protected int m_severity	ログ出力の重大度設定。

Logger のメソッド

表 12-10 Logger のメソッドの概要

メソッド	説明
Logger()（12-23 ページ）	クラス・コンストラクタです。
getSeverity()（12-23 ページ）	ログ出力に対する現在の重大度の設定を戻します。
getSeverityName()（12-24 ページ）	指定された重大度に対応する重大度の名前を戻します。
getSeverityValue()（12-24 ページ）	指定された重大度の名前に対応する重大度の値を戻します。
init()（12-24 ページ）	構成パラメータを指定して、ログ出力を初期化します（1 回のみ）。
isLoggable()（12-25 ページ）	指定された重大度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()（12-25 ページ）	メッセージを記録します。
setSeverity()（12-26 ページ）	現在の重大度を設定します。

Logger()

クラス・コンストラクタです。

構文

```
public Logger();
```

getSeverity()

ログ出力に対する現在の重大度の設定を戻します。

構文

```
public int getSeverity();
```

getSeverityName()

指定された重大度に対応する重大度の名前を戻します。

構文

```
protected final String getSeverityName( int severity);
```

パラメータ	説明
severity	重大度レベル（SEVERITY_xxx）

getSeverityValue()

指定された重大度の名前に対応する重大度の値を戻します。

構文

```
protected final int getSeverityValue( String severityName);
```

パラメータ	説明
severityName	重大度レベルの名前（エラーなど）

init()

構成パラメータを指定して、ログ出力を初期化します（1 回のみ）。ログ出力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	ログ出力の構成オプション。
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。ログ出力で使用できる情報が含まれています。

isLoggable()

指定された重大度レベルでメッセージを記録するかどうかを判別します。指定された重大度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	確認する重大度レベル

log()

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);`	指定された重大度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);`	指定された重大度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);`	指定された重大度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度
t	ログに記録可能な例外

setSeverity()

現在の重大度を設定します。

構文

```
public void setSeverity(int severity);
```

パラメータ	説明
severity	ログ出力に対する新しい重大度の設定

ProviderDeploymentDescriptor クラス

ProviderDeploymentDescriptor 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。

構文

```
public final class ProviderDeploymentDescriptor extends Object implements
Serializable
```

表 12-11 ProviderDeploymentDescriptor のメソッドの概要

メソッド	説明
ProviderDeploymentDescriptor()（12-28 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構成します。
fromXML()（12-28 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()（12-28 ページ）	このプロバイダを実装するクラスの名前を戻します。
getId()（12-28 ページ）	このプロバイダの一意の ID を戻します。
getOptions()（12-29 ページ）	プロバイダ固有のオプションを戻します。
getProviderType()（12-29 ページ）	プロバイダ・タイプを戻します。
setClassname()（12-29 ページ）	このプロバイダを実装するクラスの名前を設定します。
setId()（12-29 ページ）	プロバイダ ID を設定します。
setOptions()（12-30 ページ）	オプションを設定します。
setProviderType()（12-30 ページ）	プロバイダ・タイプを設定します。
toString()（12-30 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()（12-31 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()

プロバイダ・ディスクリプタの新しいインスタンスを構成します。

構文

```
public ProviderDeploymentDescriptor();
```

fromXML()

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()

この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()

このプロバイダの一意の ID を戻します。

構文

```
public String getId();
```

getOptions()

プロバイダ固有のオプション、またはこの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合せを戻します。

構文

```
public Hashtable getOptions();
```

getProviderType()

プロバイダ・タイプを戻します。

構文

```
public String getProviderType();
```

setClassname()

この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()

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()

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	この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合せ

setProviderType()

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()

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()

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 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、プロバイダは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記述するセキュリティ・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。

構文

```
public class RequestContext extends Object
```

表 12-12 RequestContext のメソッドの概要

メソッド	説明
RequestContext()（12-34 ページ）	このクラスのデフォルトのコンストラクタです。
getMethodName()（12-34 ページ）	この 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()（12-34 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()（12-34 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()（12-35 ページ）	この SOAP リクエスト用のレスポンス・ストリームを戻します。
getResponseEnvelope()（12-35 ページ）	SOAP レスポンスを表すエンベロープを戻します。
getResponseMap()（12-35 ページ）	SOAP レスポンスのシリアライズに必要なマッピング・レジストリを戻します。
getServiceDeploymentDescriptor()（12-35 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()（12-36 ページ）	この SOAP リクエストに対するサービス ID（URI）を戻します。
getUserContext()（12-36 ページ）	この 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()（12-36 ページ）	この SOAP リクエストに対するメソッド名を設定します。
setRequestEncodingStyle()（12-36 ページ）	リクエストに使用されたエンコーディング・スタイルを設定します。
setRequestEnvelope()（12-37 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()（12-37 ページ）	この 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()（12-37 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()（12-38 ページ）	SOAP レスポンス・エンベロープのシリアライズに必要なマッピング・レジストリを設定します。
setServiceDeploymentDescriptor（12-38 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・ディスクリプタを設定します。

表 12-12 RequestContext のメソッドの概要（続き）

メソッド	説明
setServiceId()（12-38 ページ）	この SOAP リクエストに対するサービス ID（URI）を設定します。
setUserContext()（12-39 ページ）	この 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()

このクラスのデフォルトのコンストラクタです。

構文

```
public RequestContext();
```

getMethodName()

この 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()

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()

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()

この 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()

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();
```

パラメータ	説明
smr	SOAP レスポンス・エンベロープに対するマッピング・レジストリ

getResponseMap()

SOAP レスポンスのシリアライズ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()

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 AutonomousProvider である場合は NULL を戻します。

構文

```
public ServiceDeploymentDescriptor getServiceDeploymentDescriptor();
```

getServiceId()

この 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()

この 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()

この 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()

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()

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()

この 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 ByteArrayOutputStream

setResponseEnvelope()

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()

SOAP レスポンス・エンベロープのシリアライズ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

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	この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()

この 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	このリクエストの送信先であるサービスの URI

setUserContext()

この 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 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簡単に拡張できます。

構文

```
public final class ServiceDeploymentDescriptor extends Object implements
Serializable
```

表 12-13 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。

表 12-14 ServiceDeploymentDescriptor のメソッドの概要

メソッド	説明
ServiceDeploymentDescriptor()（12-42 ページ）	新しいサービス・ディスクリプタを構成します。
buildFaultRouter()（12-42 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()（12-43 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 シリアライズ・レジストリを生成します。

表 12-14 ServiceDeploymentDescriptor のメソッドの概要（続き）

メソッド	説明
buildSqlClassMap()（12-43 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 クラスへのマップを生成します。マップの生成に失敗した場合は、SOAPException が発生します。
fromXML()（12-43 ページ）	ServiceDeploymentDescriptor 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()（12-44 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()（12-44 ページ）	この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()（12-44 ページ）	サービス ID（URI）を戻します。
getMethods()（12-44 ページ）	この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()（12-44 ページ）	このサービスのプロバイダ ID を戻します。
getProviderOptions()（12-45 ページ）	この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合せを戻します。
getProviderType()（12-45 ページ）	プロバイダ・タイプを戻します。
getScope()（12-45 ページ）	有効範囲（SCOPE_xxx 定数のいずれか）を戻します。
getServiceType()（12-45 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()（12-45 ページ）	SQL 型から Java 型へのマップを戻します。
getTypeMappings()（12-46 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 へのデシリアライズ、および Java から XML へのシリアライズの方法が定義されます。
isMethodValid()（12-46 ページ）	指定されたメソッドがこのサービスに有効であるかどうかを判別します。
setDefaultSMRClass()（12-46 ページ）	デフォルトの SOAP マッピング・レジストリ・クラスを設定します。
setFaultListener()（12-46 ページ）	障害リスナー・リストを設定します。
setId()（12-47 ページ）	サービス ID（有効な URI）を設定します。
setMethods()（12-47 ページ）	この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。

表 12-14 ServiceDeploymentDescriptor のメソッドの概要（続き）

メソッド	説明
setProviderId()（12-47 ページ）	このサービスのプロバイダの ID を設定します。
setProviderOptions()（12-48 ページ）	プロバイダ固有のオプションを設定します。
setProviderType()（12-48 ページ）	プロバイダ・タイプを設定します。
setScope()（12-48 ページ）	実行の有効範囲を設定します。
setServiceType()（12-49 ページ）	サービス・タイプを設定します。
setSqlMap()（12-49 ページ）	SQL 型から Java 型へのマップを設定します。
setTypeMappings()（12-49 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 へのデシリアライズ、および Java から XML へのシリアライズの方法が定義されます。
toXML()（12-50 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()（12-50 ページ）	このディスクリプタの出力可能な表現を戻します。

ServiceDeploymentDescriptor()

新しいサービス・ディスクリプタを構成します。

構文

```
public ServiceDeploymentDescriptor();
```

buildFaultRouter()

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()

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 シリアライズ・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()

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 クラスへのマップを生成します。マップの生成に失敗した場合は、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()

ServiceDeploymentDescriptor 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 ServiceDeploymentDescriptor 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()

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()

この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()

サービス ID（URI）を戻します。

構文

```
public String getId();
```

getMethods()

この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()

このサービスのプロバイダ ID を戻します。

構文

```
public String getProviderId();
```

getProviderOptions()

この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合せを戻します。

構文

```
public Hashtable getProviderOptions();
```

getProviderType()

プロバイダ・タイプを戻します。

構文

```
public String getProviderType();
```

getScope()

有効範囲（SCOPE_xxx 定数のいずれか）を戻します。

構文

```
public int getScope();
```

getServiceType()

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()

SQL 型から Java 型へのマップを戻します。

構文

```
public Hashtable getSqlMap();
```

getTypeMappings()

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 へのデシリアライズ、および Java から XML へのシリアライズの方法が定義されます。

構文

```
public TypeMapping[] getTypeMappings();
```

isMethodValid()

指定されたメソッドがこのサービスに有効であるかどうかを判別します。メソッドがこの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()

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()

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	この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()

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()

この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()

この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	このサービスのプロバイダ ID

setProviderOptions()

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	プロバイダ固有のオプションを表す名前と値の組合せ

setProviderType()

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()

実行の有効範囲を設定します。

構文

```
public void setScope( int scope);
```

パラメータ	説明
scope	実行の有効範囲（SCOPE_xxx 定数のいずれか)

setServiceType()

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()

SQL 型から Java 型へのマップを設定します。

構文

```
public void setSqlMap( Hashtable sqlMap);
```

パラメータ	説明
sqlMap	SQL 型から Java クラスへのマップ

setTypeMappings()

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 へのデシリアライズ、および Java から XML へのシリアライズ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()

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()

この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 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。

構文

```
public class SOAPServerContext extends Object
```

表 12-15 SOAPServerContext のメソッドの概要

メソッド	説明
SOAPServerContext()（12-52 ページ）	デフォルトのコンストラクタです。
getAttribute()（12-52 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()（12-52 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()（12-52 ページ）	グローバル・コンテキストを戻します。
getLogger()（12-53 ページ）	SOAP ログ出力を戻します。
removeAttribute()（12-53 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()（12-53 ページ）	この 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 ()（12-54 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()（12-54 ページ）	ログ出力を設定します。ログ出力は、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()

デフォルトのコンストラクタです。

構文

```
public SOAPServerContext();
```

getAttribute()

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()

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()

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()

SOAP ログ出力を戻します。ログ出力は、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()

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()

この 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 ()

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()

ログ出力を設定します。ログ出力は、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 ログ出力

UserContext クラス

UserContext 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るのではなく、JavaProvider で必要とされます。

構文

```
public class UserContext extends Object
```

表 12-16 UserContext のメソッドの概要

メソッド	説明
UserContext()（12-56 ページ）	デフォルトのコンストラクタです。
getAttribute()（12-56 ページ）	指定された名前を持つ属性を戻します。
getAttributeNames()（12-57 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()（12-57 ページ）	SOAP リクエストを行うユーザー用のユーザー証明書を戻します。
getHttpServlet()（12-57 ページ）	SOAP リクエストを処理している HttpServlet を戻します。
getHttpSession()（12-57 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()（12-58 ページ）	リクエストを送信したリモート・クライアントの Internet Protocol（IP）アドレスを戻します。
getRemoteHost()（12-58 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()（12-58 ページ）	リクエストの URI を戻します。
getSecureChannel()（12-58 ページ）	チャネルが保護されているかどうかの表示を戻します。
getUsername()（12-59 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()（12-59 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()（12-59 ページ）	この 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()（12-60 ページ）	ユーザー証明書を設定します。

表 12-16 UserContext のメソッドの概要（続き）

メソッド	説明
setHttpServlet()（12-60 ページ）	HTTP サーブレットを設定します。
setHttpSession()（12-60 ページ）	HTTP セッションを設定します。
setRemoteAddress()（12-61 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()（12-61 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()（12-61 ページ）	リクエストの URI を設定します。
setSecureChannel()（12-62 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()（12-62 ページ）	プロトコル固有のユーザー名を設定します。

UserContext()

デフォルトのコンストラクタです。

構文

```
public UserContext();
```

getAttribute()

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()

この SOAP コンテキスト内で使用可能な属性名を含む Enumeration を戻します。

構文

```
public Enumeration getAttributeNames();
```

getCertificate()

SOAP リクエストを行うユーザー用のユーザー証明書を戻します。この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()

SOAP リクエストを処理している HttpServlet 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()

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()

リクエストを送信したリモート・クライアントの Internet Protocol（IP）アドレスを戻します。

構文

```
public String getRemoteAddress();
```

getRemoteHost()

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()

リクエストの URI を戻します。

構文

```
public String getRequestURI();
```

getSecureChannel()

チャネルが保護されているかどうかの表示を示します。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()

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。この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()

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()

この 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()

ユーザー証明書を設定します。

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証明書

setHttpServlet()

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 HttpServlet

setHttpSession()

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 HttpSession

setRemoteAddress()

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()

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()

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()

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()

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

OracleSOAPTransport インタフェースが含まれており、XDK for Java で OracleAS SOAP をサポートします。

OracleSOAPTransport インタフェース

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。

構文

```
public interface OracleSOAPTransport extends SOAPTransport
```

表 12-17 OracleSOAPTransport のメソッドの概要

メソッド	説明
close()（12-64 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()（12-64 ページ）	接続プロパティを戻します。
setProperties()（12-65 ページ）	接続プロパティを設定します。

close()

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()

接続プロパティを戻します。

構文

```
public abstract Properties getProperties();
```

setProperties()

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 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。OracleAS 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（secure HTTP）が含まれます。

OracleSOAPHTTPConnection クラス

このクラスは、OracleSOAPTransport を実装します。

構文

```
public class OracleSOAPHTTPConnection extends Object
```

表 12-18 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。
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。

表 12-18 OracleSOAPHTTPConnection のフィールド（続き）

フィールド	構文	説明
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。

表 12-19 OracleSOAPHTTPConnection のメソッドの概要

メンバー	説明
OracleSOAPHTTPConnection()（12-68 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構成します。
close()（12-69 ページ）	接続をクローズします。
finalize()（12-69 ページ）	接続をファイナライズします。
getHeaders()（12-69 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()（12-69 ページ）	接続プロパティを戻します。
receive()（12-70 ページ）	受信したレスポンスの読取り元であるバッファ済 Reader を戻します。
send()（12-70 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()（12-71 ページ）	接続プロパティを設定します。

OracleSOAPHTTPConnection()

指定されたプロパティから OracleSOAPHTTPConnection の新しいインスタンスを構成します。

構文

```
public OracleSOAPHTTPConnection( Properties prop);
```

パラメータ	説明
prop	接続プロパティ

close()

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()

接続をファイナライズします。

構文

```
public void finalize();
```

getHeaders()

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()

接続プロパティを戻します。

構文

```
public Properties getProperties();
```

receive()

受信したレスポンスの読取り元であるバッファ済 Reader 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()

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、適切な理由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<->Java 間の型マッピング・レジストリ
ctx	リクエスト SOAPContext

setProperties()

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 パッケージには、XmlUtils クラスが含まれています。

XmlUtils クラス

XmlUtils 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構成する XML 文書を生成し、SOAP レスポンスを処理できます。OracleAS 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。

構文

```
public class XmlUtils
```

表 12-20 XmlUtils のメソッドの概要

メンバー	説明
XmlUtils()（12-73 ページ）	デフォルトのコンストラクタです。
extractServiceId()（12-74 ページ）	エンベロープからサービス ID を戻します。
extractMethodName()（12-74 ページ）	エンベロープからメソッド名を戻します。
parseXml()（12-75 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()（12-75 ページ）	ドキュメントを作成します。

XmlUtils()

デフォルトのコンストラクタです。

構文

```
public XmlUtils();
```

extractServiceId()

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()

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()

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、指定された 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、Reader 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 Reader
is	入力ストリーム・ソース

createDocument()

Document を作成します。Document を作成できない場合は、SOAPException が発生します。

構文

```
public static Document createDocument();
```

第 II 部

XML の C サポート

第 II 部に含まれる章は、次のとおりです。

13

XML Schema Processor for C

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。

この章の内容は次のとおりです。

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

C 用の XML Schema のメソッド

表 13-1 では、C パーサーのメソッドの概要を示します。

表 13-1 C 用の XML Schema のメソッドの概要

メソッド	説明
schemaInitialize()（13-2 ページ）	XML Schema Processor を初期化します。
schemaValidate()（13-3 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。
schemaTerminate()（13-3 ページ）	XML Schema Processor を終了（停止）します。

schemaInitialize()

XML Schema Processor を初期化します。XML Schema Processor を使用してドキュメントを検証する前にコールします。XML パーサー・コンテキストを使用して、スキーマ・コンテキスト用のメモリーを割り当てます。スキーマ・コンテキストは戻され、すべての後続のスキーマ・ファンクションに渡される必要があります。このコンテキスト・ポインタは不透明であり、そのメンバーを参照することはできません。戻りコンテキストが null の場合、初期化は正常に実行されません。また、err には、問題を示す数値のエラー・コードが設定されます。

構文

```
xsdctx *schemaInitialize(xmlctx *ctx, uword *err)
```

パラメータ	IN/OUT	説明
ctx	IN	XML パーサー・コンテキスト
err	OUT	戻されるエラー・コード

schemaValidate()

インスタンス・ドキュメントを単一または複数のスキーマに対して検証します。schemaInitialize によって戻されたスキーマ・コンテキストを渡す必要があります。検証するドキュメントは、ドキュメントの解析に使用された XML パーサー・コンテキスト inst によって指定されます。ドキュメントは解析済である必要があることに注意してください。インスタンス・ドキュメントで明示的に参照されるスキーマがない場合は、デフォルトのスキーマ（URL によって指定）が想定されます。ドキュメントにすべての必要なスキーマが指定され、デフォルト・スキーマも提供されている場合は、デフォルト・スキーマは無視されます。ドキュメントがスキーマを参照せず、デフォルト・スキーマも提供されていない場合は、エラーが発生します。

構文

```
uword schemaValidate(xsdctx *scctx, xmlctx *inst, oratext *schema)
```

パラメータ	IN/OUT	説明
scctx	IN	スキーマ・コンテキスト
inst	IN	インスタンス・ドキュメントのコンテキスト
schema	IN	デフォルト・スキーマの URL

schemaTerminate()

XML Schema Processor を終了（停止）し、schemaInitialize に渡された元の XML パーサー・コンテキストに割り当てられたすべてのメモリーを解放します。XML Schema Processor の終了後、スキーマ・コンテキストは無効になります。XML Schema Processor を継続して使用するには、schemaInitialize を使用して新しいスキーマを作成する必要があります。

構文

```
void schemaTerminate(xsdctx *scctx)
```

パラメータ	IN/OUT	説明
scctx	IN	スキーマ・コンテキスト

14

XML Parser for C

この章では、XML Parser の C 言語実装について説明します。

この章の内容は次のとおりです。

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

パーサー API

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、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、SGML（ISO 8879）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ディングします。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 文書の読取り、およびその内容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 C 実装は、W3C の XML 仕様（改訂 REC-xml-19980210）に準拠しており、XML データを読み取るために 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。

このパーサーのデフォルトの動作を次に示します。

- キャラクタ・セットのエンコーディングは UTF-8 です。すべてのドキュメントが ASCII である場合は、パフォーマンスを向上させるために、エンコーディングを US-ASCII に設定することをお薦めします。
- msghdlr が指定されないかぎり、メッセージは stderr に出力されます。
- saxcb が SAX コールバック API を使用するように設定されないかぎり、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。どの SAX コールバック・ファンクションも、不要な場合は null に設定できます。
- パーサーのデフォルト動作では、入力が整形式であるかどうかは確認されますが、入力が妥当であるかどうかは確認されません。フラグ XML_FLAG_VALIDATE を設定すると、入力を検証できます。空白処理のデフォルトの動作は、XML 1.0 仕様に完全に準拠します。この場合、すべての空白がアプリケーションに通知されますが、無視できる空白がアプリケーションに対して示されます。ただし、アプリケーションによっては、XML_FLAG_DISCARD_WHITESPACE を設定し、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を削除する方が適切な場合もあります。

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
```

```
Parsing multiple documents, but only the latest document needs to be available:
```

```
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、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。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。

- xmlparse***xxx*** が他のドキュメントを解析するためにコールされる。
- 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

開始、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。

表 14-1 に、C パーサーのメソッドを示します。

表 14-1 C パーサーのメソッドの概要

メソッド	説明
xmlinit()（14-4 ページ）	XML パーサーを初期化します。
xmlclean()（14-6 ページ）	解析中に使用されたメモリーをクリーンアップします。
xmlparse()（14-6 ページ）	URL によって指定されたドキュメントを解析します。
xmlparsebuf()（14-7 ページ）	メモリー内にあるドキュメントを解析します。
xmlparsefile()（14-8 ページ）	ファイル・システムのドキュメントを解析します。

表 14-1 C パーサーのメソッドの概要（続き）

メソッド	説明
xmlparsestream()（14-8 ページ）	ユーザー定義ストリームのドキュメントを解析します。
xmlterm()（14-9 ページ）	XML パーサーを停止します。
createDocument()（14-9 ページ）	新しいドキュメントを作成します。
isStandalone()（14-10 ページ）	ドキュメントのスタンドアロン・フラグを戻します。
isSingleChar()（14-10 ページ）	シングルバイト / マルチバイトのエンコーディング・フラグを戻します。
getEncoding()（14-10 ページ）	ドキュメントのエンコーディング名を戻します。

xmlinit()

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）以外の場合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。

エラー・コードは、`XMLERR_LEH_INIT`、`XMLERR_BAD_ENCODING`、`XMLERR_NLS_INIT`、`XMLERR_NO_MEMORY`、`XMLERR_NULL_PTR` です。

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、キャラクタ・セットのエンコーディングは UTF-8 です。すべてのドキュメントが ASCII である場合は、パフォーマンスを向上させるために、エンコーディングを US-ASCII に設定することをお薦めします。

デフォルトでは、`msghdlr` が指定されないかぎり、メッセージは `stderr` に出力されます。

デフォルトでは、`saxcb` が設定（この場合、SAX コールバック API が起動されます）されないかぎり、（DOM API がアクセス可能な）解析ツリーが構築されます。どの SAX コールバック・ファンクションも、不要な場合は 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。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。

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・ファンクションのコールバック・ルーチンに情報を渡すために定義および使用できる構造体です。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、現在は使用されていないため、null に設定します。このパラメータは、将来のリリースで、エラー・メッセージの言語を決定するために使用される予定です。

構文

```
xmlctx *xmlinit(uword *err, const oratext *encoding,
                void (*msghdlr)(void *msgctx, const oratext *msg,
                                uword errcode),
                void *msgctx, const xmlsaxcb *saxcb, void *saxcbctx,
                const xmlmemcb *memcb, void *memcbctx, const oratext *lang);
```

パラメータ	IN/OUT	説明
er	OUT	エラー（存在する場合、C のみ）
encoding	IN	デフォルトのキャラクタ・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・ハンドラのコンテキスト
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 コールバックの内容
memcb	IN	メモリー・ファンクション・コールバック
memcbctx	IN	メモリー・ファンクション・コールバックのコンテキスト
lang	IN	エラー・メッセージの言語

xmlclean()

前回の解析中に使用されたメモリーを解放します。このファンクションは、単一のコンテキストを使用して複数のドキュメントを解析する必要がある場合に有効です。2 つ目以降のドキュメントを解析する前に、xmlclean をコールして、前のドキュメントが使用したメモリーを解放します。

このコール後に、メモリーは内部で再利用されます。メモリーは、xmlterminate がコールされるまで、システムに戻されません。

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

xmlparse()

URL 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 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。

フラグ・ビットは、OR 計算して、XML パーサーのデフォルトの動作をオーバーライドする必要があります。次のフラグ・ビットを設定できます。

- XML_FLAG_VALIDATE は、検証を有効にします。デフォルトの動作では、入力が検証されません。
- XML_FLAG_DISCARD_WHITESPACE は、無視できる空白を削除します。空白処理のデフォルトの動作は、XML 1.0 仕様に完全に準拠します。この場合、すべての空白がアプリケーションに通知されますが、無視できる空白がアプリケーションに対して示されます。ただし、アプリケーションによっては、XML_FLAG_DISCARD_WHITESPACE を設定し、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を削除する方が適切な場合もあります。
- XML_FLAG_DTD_ONLY は、入力が外部 DTD のみであり、完全なドキュメントではないことを XML パーサーに通知します。
- XML_FLAG_STOP_ON_WARNING は、検証に関する警告が発生した場合、すぐに XML パーサーを停止します。デフォルトでは、検証に関する警告が出力されますが、検証は継続されます。

SAX コールバックに渡されたメモリーまたは DOM 解析ツリーを使用して格納されたメモリーは、次のいずれかの操作が実行されるまで解放されません。

- xmlparse*xxx* が他のドキュメントを解析するためにコールされる。
- xmlclean がコールされる。
- xmlterm がコールされる。

構文

```
uword xmlparse(xmlctx *ctx, const oratext *url,
               const oratext *encoding, 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
url	IN	XML 文書の URL
encoding	IN	デフォルトのキャラクタ・セットのエンコーディング
flags	IN	使用するオプション

xmlparsebuf()

メモリー内にある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 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。このファンクションは、URI からではなくユーザーのバッファから入力されることを除き、xmlparse と同一です。

構文

```
uword xmlparsebuf(xmlctx *ctx, const oratext *buffer, size_t len,
                  const oratext *encoding, 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	メモリー内のドキュメントへのポインタ
len	IN	バッファ長
encoding	IN	デフォルトのキャラクタ・セットのエンコーディング
flags	IN	使用するオプション

xmlparsefile()

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 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。このファンクションは、URL からではなくユーザーのファイル・システムから入力されることを除き、xmlparse と同一です。

構文

```
uword xmlparsefile(xmlctx *ctx, const oratext *path,
                   const oratext *encoding, 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
encoding	IN	デフォルトのキャラクタ・セットのエンコーディング
flags	IN	使用するオプション

xmlparsestream()

ユーザー定義ストリームから読み取られる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 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。このファンクションは、URL からではなくユーザー定義ストリームから入力されることを除き、xmlparse と同一です。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 *encoding, 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	ストリームまたはストリーム・コンテキストへのポインタ

パラメータ	IN/OUT	説明
encoding	IN	デフォルトのキャラクタ・セットのエンコーディング
flags	IN	使用するオプション

xmlterm()

XML パーサーを終了します。このファンクションは、xmlinit のコール後で、メイン・プログラムの終了前にコールします。このファンクションは、XML パーサーが使用したすべてのメモリーを解放し、コンテキストを終了します。終了したコンテキストは、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）。

構文

```
uword xmlterm(xmlctx *ctx);
```

パラメータ	IN/OUT	説明
ctx	IN	XML パーサー・コンテキスト

createDocument()

メモリー内に新しいドキュメントを作成します。このファンクションは、メモリー内に新しいドキュメントを構成する場合に使用します。常に、DOCUMENT_NODE 型のノードが XML 文書のルートになります。このファンクションは、そのルート・ノードを作成し、それをコンテキストで設定します。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

isStandalone()

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?> 処理命令で指定した、ドキュメントのスタンドアロン・フラグのブーリアン値を戻します。

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

isSingleChar()

現行のドキュメントをシングルバイト・キャラクタ（ASCII など）またはマルチバイト・キャラクタ（UTF-8 など）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。ドキュメントのエンコーディングの実際の名前を戻す getEncoding と比較してください。

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

getEncoding()

現行のドキュメントの文字コード体系の名前（ASCII、UTF8 など）を戻します。現行のエンコーディングがシングルバイトかマルチバイトかを示すブーリアン・フラグのみを戻す isSingleChar と比較してください。

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

XSLT API

XSLT は、XML 文書を他の XML 文書に変換するための言語です。

XSLT は、XML のスタイルシート言語である XSL の一部として使用できるように設計されています。XSLT の他に、XSL にはフォーマットを指定するための XML ボキャブラリが含まれています。XSL は、XSLT を使用して XML 文書のスタイルを指定し、ドキュメントをフォーマット・ボキャブラリを使用する他の XML 文書に変換する方法を記述します。

また、XSLT は、XSL とは無関係に使用できるように設計されています。ただし、XSLT は、完全な汎用 XML 変換言語というわけではありません。XSLT は、主に、XSLT を XSL の一部として使用する場合に必要な種類の変換を行うために設計されています。

XSLT に記述されている変換は、ソース・ツリーを結果ツリーに変換するための規則を記述しています。変換は、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。パターンは、ソース・ツリー内の要素に対して一致検索されます。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。結果ツリーは、ソース・ツリーからは切り離されています。結果ツリーの構造は、ソース・ツリーの構造と完全に異なる場合があります。結果ツリーを構成する場合、ソース・ツリーの要素をフィルタにかけ、再度順序付けたり、任意の構造を追加することができます。

XSLT に記述されている変換をスタイルシートといいます。これは、XSLT が XSL フォーマット・ボキャブラリに変換される場合に、変換がスタイルシートとして機能するためです。

スタイルシートには、一連のテンプレート規則が含まれています。テンプレート規則には、ソース・ツリー内のノードに対して一致検索されるパターン、および結果ツリーの一部を形成するようにインスタンス化できるテンプレートという 2 つの部分があります。これによって、1 つのスタイルシートを、同様のソース・ツリー構造を持つ様々なドキュメントに適用できます。

テンプレートは、特定のソース要素が結果ツリーの一部を作成できるようにインスタンス化されます。テンプレートは、リテラル結果要素構造を指定する要素を含むことができます。また、テンプレートは、結果ツリーの一部を作成するための命令である XSLT 名前空間の要素も含むことができます。テンプレートがインスタンス化されると、各命令が実行され、その命令が作成した結果ツリーの一部によって置換されます。命令は、子孫ソース要素を選択および処理できます。子孫要素を処理すると、適切なテンプレート規則が検索され、そのテンプレートがインスタンス化されることによって、結果ツリーの一部が作成されます。要素は命令の実行によって選択されている場合にのみ処理されることに注意してください。結果ツリーは、ルート・ノード用のテンプレート規則を検索し、そのテンプレートをインスタンス化することによって構成されます。

XML 文書を読み取り、それらを異なるスタイルを持つ他の XML 文書に変換するには、XML プロセッサというソフトウェア・モジュールが使用されます。

この XSL プロセッサの C 実装は、W3C の XSLT 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、XML プロセッサに必要な動作を含んでいます。

xslprocess()

これらのファンクションは、XML 文書ソースを使用して XSL スタイルシートを処理し、正常に実行されたことを示すか、またはエラー・コードを戻します。

構文

```
uword xslprocess(xmlctx *docctx, xmlctx *xslctx,
                 xmlctx *resctx, xmlnode **result);
```

パラメータ	IN/OUT	説明
xmlctx	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

W3C の SAX API

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。

XML（または SGML）API には、主に次の 2 つのタイプがあります。

- ツリーベースの API
- イベントベースの API

ツリーベースの API は、XML 文書を内部ツリー構造にコンパイルし、アプリケーションが DOM（XML および HTML ドキュメント用のツリーベースの標準 API）を使用してそのツリーをナビゲートできるようにします。

一方、イベントベースの API は、コールバックを介して解析イベント（要素の始まりや終わりなど）をアプリケーションに直接通知し、通常、内部ツリーを構築しません。アプリケーションは、ハンドラを実装して、Graphical User Interface（GUI）でイベントを処理する場合と同様に様々なイベントを処理します。

ツリーベースの API は様々なアプリケーションで有効ですが、ドキュメントが大きい場合、多くのシステム・リソースが消費される場合があります（詳細に制御された環境では、簡単な方法でツリーを構築し、この問題の一部を回避できる場合もあります）。さらに、アプリケーションによっては、独自のデータ・ツリーを個別に構築する必要があり、新しいツリーにマップするためにのみ解析ノードのツリーを構築することは非常に非効率的です。

いずれの場合も、イベントベースの API は、XML 文書への単純でより低レベルのアクセスを提供します。ユーザーは、使用可能なシステム・メモリーよりサイズの大きいドキュメントを解析したり、コールバック・イベント・ハンドラを使用して独自のデータ構造を構築できます。

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へのポインタも含まれるため、コンテキスト・ポインタは各 SAX 関数に渡されます。

次に、SAX のコールバック構造を示します。

```
typedef struct
{
   sword (*startDocument)(void *ctx);
   sword (*endDocument)(void *ctx);
   sword (*startElement)(void *ctx, const oratext *name,
                         const struct xmlarray *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);
} xmlsaxcb;
```

表 14-2 に、コールバック・ファンクションを示します。nsStartElement() の例外を持つファンクションはすべて、SAX 標準に準拠します。

表 14-2 SAX コールバック・ファンクションの概要

メソッド	説明
characters()（14-15 ページ）	要素内の文字データの通知を受け取ります。
endDocument()（14-15 ページ）	ドキュメントの終わりの通知を受け取ります。
endElement()（14-16 ページ）	要素の終わりの通知を受け取ります。
ignorableWhitespace()（14-16 ページ）	要素内容内の無視できる空白の通知を受け取ります。
notationDecl()（14-17 ページ）	表記法宣言の通知を受け取ります。
processingInstruction()（14-17 ページ）	処理命令の通知を受け取ります。
startDocument()（14-18 ページ）	ドキュメントの始まりの通知を受け取ります。
startElement()（14-18 ページ）	要素の始まりの通知を受け取ります。
unparsedEntityDecl()（14-19 ページ）	解析対象外エンティティ宣言の通知を受け取ります。
nsStartElement()（14-19 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()

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	クライアント・コンテキスト・ポインタ
ch	IN	文字
len	IN	文字ポインタから使用する文字数

endDocument()

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()

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	要素型の名前

ignorableWhitespace()

要素内容内の無視できる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	クライアント・コンテキスト
ch	IN	空白文字
len	IN	文字ポインタから使用する文字数

notationDecl()

用途

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	クライアント・コンテキスト
name	IN	表記法の名前
publicId	IN	表記法の公開識別子（公開識別子が使用不可の場合は null）
systemId	IN	表記法のシステム識別子

processingInstruction()

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	クライアント・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（指定されていない場合は null）

startDocument()

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	クライアント・コンテキスト

startElement()

要素の始まりの通知を受け取ります。

構文

```
word (*startElement)(
   void *ctx,
   const oratext *name,
   const struct xmlattrs *attrs);
```

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	要素型の名前
attrs	IN	指定した値またはデフォルト値の属性

unparsedEntityDecl()

解析対象外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	クライアント・コンテキスト
name	IN	エンティティの名前
publicId	IN	エンティティの公開識別子（公開識別子が使用不可の場合は null）
systemId	IN	エンティティのシステム識別子
notationName	IN	対応付けられた表記法の名前

nsStartElement()

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 xmlattrs *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

パラメータ	IN/OUT	説明
namespace	IN	要素の名前空間（URI）
attrs	IN	指定した値またはデフォルト値の属性

W3C の DOM API

DOM は、HTML および XML 文書用の 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、要素および内容を追加、変更および削除することができます。HTML または XML 文書のすべての内容は、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、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` を与え、`getNodeValue` によって確立されるパターンと一致させます。
- 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、ノードの子ノードの数を戻すファンクションが定義されていないため、`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。表 14-3 に、C DOM ファンクションを示します。

表 14-3 W3C DOM メソッドの概要

メソッド	説明
appendChild()（14-25 ページ）	現行のノードに子ノードを追加します。
appendData()（14-25 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()（14-26 ページ）	現行のノードと同一の新しいノードを作成します。
createAttribute()（14-26 ページ）	要素ノードの新しい属性を作成します。
createCDATASection()（14-27 ページ）	CDATA ノードを作成します。
createComment()（14-27 ページ）	コメント・ノードを作成します。

表 14-3 W3C DOM メソッドの概要（続き）

メソッド	説明
createDocumentFragment()（14-27 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()（14-28 ページ）	要素ノードを作成します。
createEntityReference()（14-28 ページ）	実体参照ノードを作成します。
createProcessingInstruction()（14-29 ページ）	処理命令ノードを作成します。
createTextNode()（14-29 ページ）	テキスト・ノードを作成します。
deleteData()（14-30 ページ）	ノードの文字データから文字列を削除します。
getAttribute()（14-30 ページ）	属性の値を戻します。
getAttributeIndex()（14-31 ページ）	索引が与えられた要素の属性を戻します。
getAttributeNode()（14-31 ページ）	名前が与えられた要素の属性ノードを取得します。
getAttributes()（14-32 ページ）	要素の属性の配列を戻します。
getAttrName()（14-32 ページ）	属性の名前を戻します。
getAttrSpecified()（14-32 ページ）	属性の specified フラグの値を戻します。
getAttrValue()（14-33 ページ）	属性の値（定義）を戻します。
getCharData()（14-33 ページ）	テキスト・ノードの文字データを戻します。
getCharLength()（14-33 ページ）	テキスト・ノードの文字データの長さを戻します。
getChildNode()（14-34 ページ）	ノードの配列から、索引で指定したノードを戻します。
getChildNodes()（14-34 ページ）	ノードの子ノードの配列を戻します。
getContentModel()（14-35 ページ）	DTD から要素の内容モデルを戻します。
getDocType()（14-35 ページ）	トップレベルのドキュメント・ノードを戻します。
getDocTypeEntities()（14-35 ページ）	最上位（ルート）の要素ノードを戻します。
getDocTypeName()（14-36 ページ）	現行の DTD を戻します。
getDocTypeNotations()（14-36 ページ）	DTD の汎用エンティティの配列を戻します。
getElementsByTagName()（14-36 ページ）	DTD の名前を戻します。

表 14-3 W3C DOM メソッドの概要（続き）

メソッド	説明
getDocument()（14-37 ページ）	DTD の表記法の配列を戻します。
getDocumentElement()（14-37 ページ）	一致する名前を持つ要素のリストを戻します。
getEntityNotation()（14-37 ページ）	エンティティの NDATA を戻します。
getEntityPubID()（14-38 ページ）	エンティティの公開識別子を戻します。
getEntitySysID()（14-38 ページ）	エンティティのシステム識別子を戻します。
getFirstChild()（14-38 ページ）	ノードの最初の子ノードを戻します。
getImplementation()（14-39 ページ）	DOM インプリメンテーション構造（定義されている場合）を戻します。
getLastChild()（14-39 ページ）	ノードの最後の子ノードを戻します。
getNamedItem()（14-39 ページ）	ノードのリストから、名前で指定したノードを戻します。
getNextSibling()（14-40 ページ）	ノードの直後のノードを戻します。
getNodeMapLength()（14-40 ページ）	ノード・マップ内のエンティティの数を戻します。
getNodeName()（14-40 ページ）	ノードの名前を戻します。
getNodeType()（14-41 ページ）	ノードの型コードを戻します（列挙）。
getNodeValue()（14-41 ページ）	ノードの値（文字データ）を戻します。
getNotationPubID()（14-41 ページ）	表記法の公開識別子を戻します。
getNotationSysID()（14-42 ページ）	表記法のシステム識別子を戻します。
getOwnerDocument()（14-42 ページ）	与えられたノードを含むドキュメント・ノードを戻します。
getParentNode()（14-42 ページ）	処理命令のデータを戻します。
getPIData()（14-43 ページ）	処理命令のターゲットを戻します。
getPITarget()（14-43 ページ）	ノードの親ノードを戻します。
getPreviousSibling()（14-43 ページ）	ノードの直前のノードを戻します。
getTagName()（14-44 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()（14-44 ページ）	要素ノードに属性があるかどうかを判断します。
hasChildNodes()（14-45 ページ）	ノードに子があるかどうかを判断します。

表 14-3 W3C DOM メソッドの概要（続き）

メソッド	説明
hasFeature()（14-45 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。
insertBefore()（14-46 ページ）	与えられた参照ノードの前に新しい子ノードを挿入します。
insertData()（14-46 ページ）	ノードの既存データに新しい文字データを挿入します。
isStandalone()（14-47 ページ）	ドキュメントがスタンドアロンかどうかを判別します。
nodeValid()（14-47 ページ）	ノードを現行の DTD に対して検証します。
normalize()（14-48 ページ）	隣接するテキスト・ノードをマージすることによって、ノードを正規化します。
numAttributes()（14-48 ページ）	要素ノードの属性の数を戻します。
numChildNodes()（14-48 ページ）	ノードの子ノードの数を戻します。
removeAttribute()（14-49 ページ）	与えられた名前を持つ、要素の属性を削除します。
removeAttributeNode()（14-49 ページ）	与えられたポインタが指す、要素の属性を削除します。
removeChild()（14-49 ページ）	ノードをその親ノードの子リストから削除します。
removeNamedItem()（14-50 ページ）	与えられた名前を持つノードをノード・リストから削除します。
replaceChild()（14-50 ページ）	ノードを他のノードで置換します。
replaceData()（14-51 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。
setAttribute()（14-51 ページ）	与えられた属性名および属性値を持つ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()（14-52 ページ）	新しい属性へのポインタを与えると、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。
setNamedItem()（14-52 ページ）	親ノードの子リストに新しいノードを設定（追加または置換）します。
setNodeValue()（14-53 ページ）	ノードの値（文字データ）を設定します。
setPIData()（14-53 ページ）	処理命令のデータを設定します。
setPIData()（14-53 ページ）	ノードの文字データを 2 つの部分に分割します。
substringData()（14-54 ページ）	ノードの文字データの部分文字列を戻します。

appendChild()

与えられた親ノードの子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
newnode	(IN)	追加する新しいノード

appendData()

与えられた文字列をテキスト・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)	ノードへのポインタ
arg	(IN)	追加する新しいデータ

cloneNode()

この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、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、子であるテキスト・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。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。

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()

与えら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おらず、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

createCDATASection()

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()

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

createDocumentFragment()

新しい DOCUMENT_FRAGMENT 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

createElement()

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

createEntityReference()

新しい ENTITY_REFERENCE 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	実体参照の名前

createProcessingInstruction()

与えられたターゲットおよび内容を持つ新しい PROCESSING_INSTRUCTION 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()

与えられた内容を持つ新しいテキスト・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()

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	ノードへのポインタ
offset	IN	部分文字列の開始オフセット（最初の char は 0）
count	IN	部分文字列の長さ

getAttribute()

指定された名前の属性の値を戻します。

構文

```
const oratext *getAttribute(const xmlnode *node, const oratext *name)
```

パラメータ	IN/OUT	説明
node	IN	属性をスキャンするノード
name	IN	属性の名前

getAttributeIndex()

属性の配列から、与えられた索引（0 から開始）が付いた 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	属性ノード構造へのポインタ
index	IN	戻す 0（ゼロ）ベースの属性番号

getAttributeNode()

与えられた名前を持つ、要素ノードの属性への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	要素ノードへのポインタ
name	IN	属性の名前

getAttributes()

与えら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のポインタをフェッチしたり、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

getAttrName()

属性へのポインタが与えられると、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	属性へのポインタ

getAttrSpecified()

属性の specified フラグを戻します。元のドキュメントでこの属性に値が明示的に与えられている場合は true を、それ以外の場合は 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	属性へのポインタ

getAttrValue()

属性へのポインタが与えられると、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	属性へのポインタ

getCharData()

テキスト・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	ノードへのポインタ

getCharLength()

テキスト・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

```
ub4 getCharLength(const xmlnode *node)
```

パラメータ	IN/OUT	説明
node	IN	ノードへのポインタ

getChildNode()

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 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
index	IN	0（ゼロ）ベースの子ノード番号

getChildNodes()

与えら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することができます。

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()

現行の DTD から、名前で指定した要素の内容モデルを戻します。内容モデルは XML ノードで構成されているため、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。「?」、「*」および「+」修飾子を内容モデル・ノードに戻す getModifier ファンクションも参照してください。

構文

```
xmlnode *LpxGetContentModel(xmldtd *dtd, oratext *name)
```

パラメータ	IN/OUT	説明
dtd	IN	DTD へのポインタ
name	IN	要素の名前

getDocType()

現行ドキュメントの（不透明）DTD へのポインタを戻します。

構文

```
xmldtd* getDocType(xmlctx *ctx)
```

パラメータ	IN/OUT	説明
ctx	IN	XML パーサー・コンテキスト

getDocTypeEntities()

与えられた 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(xmldtd* dtd)
```

パラメータ	IN/OUT	説明
dtd	IN	DTD へのポインタ

getDocTypeName()

与えられた DTD の名前を戻します。

構文

```
oratext *getDocTypeName(xmldtd* dtd)
```

パラメータ	IN/OUT	説明
dtd	IN	DTD へのポインタ

getDocTypeNotations()

与えられた 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(xmldtd* dtd)
```

パラメータ	IN/OUT	説明
dtd	IN	DTD へのポインタ

getElementsByTagName()

与えられたタグ名を持つ（与えられたノードがルートであるツリー内にある）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、ドキュメント全体が検索されます。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	ルート・ノード
name	IN	要素のタグ名

getDocument()

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・ノードの子ノード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()

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

getEntityNotation()

エンティティ・ノードの NDATA 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	エンティティへのポインタ

getEntityPubID()

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	エンティティへのポインタ

getEntitySysID()

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	エンティティへのポインタ

getFirstChild()

与えら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	ノードへのポインタ

getImplementation()

この実装の DOM インプリメンテーション構造への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()

与えら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	ノードへのポインタ

getNamedItem()

ノードの配列から、名前で指定したノードを戻します。また、ユーザーの索引（提供されている場合）をそのノードの子ノード番号（0 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	検索するノードの索引

getNextSibling()

与えられたノードの直後のノード（親ノードの次の子ノード）へのポインタを戻します。ノードが最後の子ノードである場合は、null が戻されます。

構文

```
xmlnode* getNextSibling(const xmlnode *node)
```

パラメータ	IN/OUT	説明
node	IN	ノードへのポインタ

getNodeMapLength()

ノードの配列（getChildNodes によって戻される）を与えられると、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()

与えられたノードの名前を戻します。ノードに名前がない場合は、null を戻します。現在は「getName」と「name」が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	ノードへのポインタ

getNodeType()

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	ノードへのポインタ

getNodeValue()

ノードの値（対応付けられた文字データ）を戻します。ノードにデータがない場合は、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	ノードへのポインタ

getNotationPubID()

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
node	IN	ノードへのポインタ

getNotationSysID()

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
node	IN	ノードへのポインタ

getOwnerDocument()

与えら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	ノードへのポインタ

getParentNode()

与えら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である場合は、null が戻されます。

構文

```
xmlnode* getParentNode(const xmlnode *node)
```

パラメータ	IN/OUT	説明
node	IN	ノードへのポインタ

getPIData()

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	処理命令ノードへのポインタ

getPITarget()

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	処理命令ノードへのポインタ

getPreviousSibling()

与えられたノードの直前のノード（このノードの前にある同じレベルのノード）を戻します。ノードが最初の子ノードである場合は、null が戻されます。

構文

```
xmlnode* getPreviousSibling(const xmlnode *node)
```

パラメータ	IN/OUT	説明
node	IN	ノードへのポインタ

getTagName()

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じです。「getNodeName」を参照してください。W3C のワーキング・グループでは、DOM 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	ノードへのポインタ

hasAttributes()

与えられたノードに定義済の属性があるかどうかを判別し、ある場合は true を、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 の拡張機能です。

構文

```
boolean hasAttributes(const xmlnode *node)
```

パラメータ	IN/OUT	説明
node	IN	ノードへのポインタ

hasChildNodes()

与えられたノードに子ノードがあるかどうかを判別し、ある場合は true を、ない場合は false を戻します。getChildNodes がポインタ（子ノードがある場合）または null（子ノードがない場合）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	ノードへのポインタ

hasFeature()

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（大文字 / 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です。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、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(xmlctx *ctx, const oratext *feature, const oratext *version)
```

パラメータ	IN/OUT	説明
ctx	IN	XML コンテキスト
featue	IN	パッケージ名
version	IN	パッケージのバージョン番号

insertBefore()

与えられた親ノードの子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・フラグメントである場合は、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	親ノード
newChild	IN	新しい子ノード
refChild	IN	前にノードが挿入される参照ノード

insertData()

指定し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	ノードへのポインタ
offset	IN	挿入点（開始位置）
refChild	IN	挿入する新しい文字列

isStandalone()

ドキュメントの <?xml?> 処理命令で指定される、standalone 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()

ノードを DTD に対して検証します。検証結果が正常である場合は 0（ゼロ）を、それ以外の場合は 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 XML Class Generator（あるいはその両方）を介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, const xmlnode *node)
```

パラメータ	IN/OUT	説明
ctx	IN	XML コンテキスト
node	IN	ノードへのポインタ

normalize()

隣接するテキスト・ノードをマージして、要素を正規化します。隣接するテキスト・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	要素ノードへのポインタ

numAttributes()

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()

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	不透明ノード構造へのポインタ

removeAttribute()

指定した属性名の属性を要素ノードから削除します。削除し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	要素ノードへのポインタ
name	IN	削除する属性の名前

removeAttributeNode()

属性へのポインタを与えると、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	要素ノードへのポインタ
attr	IN	削除する属性ノード

removeChild()

与えられたノードをその親ノードから削除し、削除したノードを戻します。

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()

指定し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()

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。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 *replaceChild(xmlctx *ctx, xmlnode *newChild, xmlnode *oldChild)
```

パラメータ	IN/OUT	説明
ctx	IN	XML コンテキスト
newChild	IN	新しく置換するノード
oldChild	IN	置換される古いノード

replaceData()

与えられた文字オフセットから始まる、与えら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, oratext *arg)
```

パラメータ	IN/OUT	説明
ctx	IN	XML コンテキスト
node	IN	ノードへのポインタ
offset	IN	置換する部分文字列の開始オフセット（0 から開始）
count	IN	古い部分文字列の長さ
arg	IN	置換テキスト

setAttribute()

要素の新しい属性を作成します。指定する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	要素ノードへのポインタ
name	IN	新しい属性の名前
value	IN	新しい属性の値

setAttributeNode()

与えられた要素に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は、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	要素ノードへのポインタ
newNode	IN	新しい属性へのポインタ
oldNode	OUT	古い属性の戻りポインタ

setNamedItem()

親ノードのマップ内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 *parent, xmlnode *node, xmlnode **old)
```

パラメータ	IN/OUT	説明
ctx	IN	XML コンテキスト
parent	IN	新しいノードを追加する親ノード
node	IN	新しいノードの名前
old	IN	置換されるノードへのポインタ

setNodeValue()

ノードに対応付けられる値（文字データ）を設定します。

構文

```
boolean setNodeValue(xmlnode *node, const oratext *data)
```

パラメータ	IN/OUT	説明
node	IN	ノードへのポインタ
data	IN	ノードの新しいデータ

setPIData()

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	処理命令ノードへのポインタ
data	IN	処理命令の新しいデータ

splitText()

テキスト・ノードを、指定したオフセットで 2 つのテキスト・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。元のノードの直後に兄弟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()

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	ノードへのポインタ
offset	IN	置換する部分文字列の開始オフセット（0 から開始）
count	IN	部分文字列の長さ
arg	IN	置換テキスト

名前空間 API

名前空間 API は、DOM の拡張機能であるインタフェースを提供し、ドキュメント名前空間に関する情報を提供します。

XML 名前空間は、XML 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。単一の 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、このマークアップを新しく再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場合があります。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生成されます。接頭辞の有効範囲を決定したり、デフォルト値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ベースの構文が、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。

表 14-4 に、C 名前空間のデータ構造およびデータ型を示します。

表 14-4 C 名前空間のデータ構造およびデータ型

型	定義	説明
oratext	typedef unsigned char oratext;	—
xmlattr	typedef struct xmlattr xmlattr;	XMLATTR の内容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;	XMLNODE の内容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 14-5 に、C 名前空間のメソッドを示します。

表 14-5 C 名前空間のメソッドの概要

型	説明
getAttrLocal()（14-56 ページ）	属性のローカル名を戻します。
getAttrNamespace()（14-57 ページ）	属性の名前空間（URI）を戻します。
getAttrPrefix()（14-57 ページ）	属性の接頭辞を戻します。
getAttrQualifiedName()（14-57 ページ）	属性の完全修飾名を戻します。
getNodeLocal()（14-58 ページ）	ノードのローカル名を戻します。
getNodeNamespace()（14-58 ページ）	ノードの名前空間（URI）を戻します。
getNodePrefix()（14-58 ページ）	ノードの接頭辞を戻します。
getNodeQualifiedName()（14-59 ページ）	ノードの修飾名を戻します。

getAttrLocal()

この属性のローカル名を戻します。

構文

```
const oratext *getAttrLocal(const xmlattr *attr);
```

パラメータ	IN/OUT	説明
attr	IN	不透明属性構造へのポインタ（「getAttribute」を参照）

getAttrNamespace()

この属性の名前空間を戻します。

構文

```
const oratext *getAttrNamespace(const xmlattr *attr);
```

パラメータ	IN/OUT	説明
attr	IN	不透明属性構造へのポインタ（「getAttribute」を参照）

getAttrPrefix()

この属性の接頭辞を戻します。

構文

```
const oratext *getAttrPrefix(const xmlattr *attr);
```

パラメータ	IN/OUT	説明
attr	IN	不透明属性構造へのポインタ（「getAttribute」を参照）

getAttrQualifiedName()

この属性の完全修飾名を戻します。

構文

```
const oratext *getAttrQualifiedName(const xmlattr *attr);
```

パラメータ	IN/OUT	説明
attr	IN	不透明属性構造へのポインタ（「getAttribute」を参照）

getNodeLocal()

この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	ローカル名を取得するノード

getNodeNamespace()

この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	名前空間を取得するノード

getNodePrefix()

このノード接頭辞を戻します。

構文

```
const oratext *getNodePrefix(const xmlnode *node);
```

パラメータ	IN/OUT	説明
node	IN	接頭辞を取得するノード

getNodeQualifiedName()

この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	名前を取得するノード

データ型

表 14-6 に、データ型を示します。

表 14-6 データ型

型	定義	説明
oratext	typedef unsigned char oratext;	基本文字ポインタ型です。
String	typedef unsigned char String;	基本文字ポインタ型です。
xmlctx	typedef struct xmlctx xmlctx;	トップレベルの XML コンテキストです。XMLCTX の内容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。

xmlmemcb	struct xmlmemcb
	{
	void *(*alloc)(void *ctx, size_t size);
	void (*free)(void *ctx, void *ptr);
	void *(*realloc)(void *ctx, void *ptr, size_t size);
	};
	typedef struct xmlmemcb xmlmemcb
		xmlinit に渡されたメモリー・コールバック構造です。

表 14-6 データ型（続き）

型	定義	説明

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);
	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);
	};
	typedef struct xmlsaxcb xmlsaxcb;
		xmlinit に渡された SAX コールバック構造です。

ub4	typedef unsigned int ub4;	最小値が 4 バイトの符号なし整数です。
uword	typedef unsigned int uword;	ネイティブ・ワード・サイズの符号なし整数です。
boolean	typedef int boolean;	
oratext	typedef unsigned char oratext;	
xmlcpmod	XMLCPMOD_NONE = 0 /* no modifier */ XMLCPMOD_OPT = 1 /* '?' optional */ XMLCPMOD_0MORE = 2 /* '*' zero or more */ XMLCPMOD_1MORE = 3 /* '+' one or more */	内容モデル・ノードの修飾子です。「getModifier」を参照してください。
xmlctx	typedef struct xmlctx xmlctx;	XMLCTX の内容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 14-6 データ型（続き）

型	定義	説明
xmlnode	typedef struct xmlnode xmlnode;	XMLNODE の内容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;	XMLNODES の内容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 */ DOCUMENT_FRAGMENT_NODE = 11 /* document fragment*/ NOTATION_NODE = 12 /* notation */	解析ツリーのノード型です。「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。

第 III 部

XML の C++ サポート

第 III 部に含まれる章は、次のとおりです。

15

XML Schema Processor for C++

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。

この章の内容は次のとおりです。

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

C++ 用の XML Schema のメソッド

表 15-1 では、C++ パーサーのメソッドの概要を示します。

表 15-1 C++ 用の XML Schema のメソッドの概要

メソッド	説明
XMLSchema::initialize() (15-2 ページ)	XML Schema Processor を初期化します。
XMLSchema::validate() (15-3 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。
XMLSchema::terminate() (15-3 ページ)	XML Schema Processor を終了（停止）します。

XMLSchema::initialize()

XML Schema Processor を初期化します。XML Schema Processor を使用してドキュメントを検証する前にコールします。XML パーサー・コンテキストを使用して、スキーマ・コンテキスト用のメモリーを割り当てます。スキーマ・コンテキストは戻され、すべての後続のスキーマ・ファンクションに渡される必要があります。このコンテキスト・ポインタは不透明であり、そのメンバーを参照することはできません。戻りコンテキストが null の場合、初期化は正常に実行されません。また、err には、問題を示す数値のエラー・コードが設定されます。

構文

```
uword initialize(xmlctx *ctx)
```

パラメータ	IN/OUT	説明
ctx	IN	XML パーサー・コンテキスト

XMLSchema::validate()

インスタンス・ドキュメントを単一または複数のスキーマに対して検証します。XMLSchema::initialize() によって戻されたスキーマ・コンテキストを渡す必要があります。検証するドキュメントは、ドキュメントの解析に使用された XML パーサー・コンテキストによって指定されます。ドキュメントは解析済である必要があることに注意してください。インスタンス・ドキュメントで明示的に参照されるスキーマがない場合は、デフォルトのスキーマ（URL によって指定）が想定されます。ドキュメントにすべての必要なスキーマが指定され、デフォルト・スキーマも提供されている場合は、デフォルト・スキーマは無視されます。ドキュメントがスキーマを参照せず、デフォルト・スキーマも提供されていない場合は、エラーが発生します。

構文

```
uword validate(xmlctx *inst, oratext *schema)
```

パラメータ	IN/OUT	説明
inst	IN	インスタンス・ドキュメントのコンテキスト
schema	IN	デフォルト・スキーマの URL

XMLSchema::terminate()

XML Schema Processor を終了（停止）し、XMLSchema::initialize() に渡された元の XML パーサー・コンテキストに割り当てられたすべてのメモリーを解放します。XML Schema Processor の終了後、スキーマ・コンテキストは無効になります。XML Schema Processor を継続して使用するには、XMLSchema::initialize() を使用して新しいスキーマを作成する必要があります。

構文

```
void terminate()
```

16

XML Parser for C++

この章の内容は次のとおりです。

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

クラス Attr

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します。

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()

属性の名前を戻します。

構文

```
String getName()
```

getValue()

属性の定義を戻します。

構文

```
String getValue()
```

getSpecified()

属性の specified フラグの値を戻します。DOM では、元のドキュメントでこの属性に値が明示的に与えられている場合は true を、それ以外の場合は false を戻すよう指定されています。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る場合でも）、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 に設定された 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()

属性の値を設定します。

構文

```
void setValue(String value)
```

パラメータ	説明
value	属性の新しい値

クラス CDATASection

このクラスは、**Text** のサブクラスである CDATA ノード・タイプを実装します。

クラス Comment

このクラスは、**CharacterData** のサブクラスである COMMENT ノード・タイプを実装します。

クラス Document

このクラスには、ノードを作成し、取り出すためのメソッドが含まれます。

表 16-2 Document のメソッドの概要

メソッド	説明
createAttribute()（16-6 ページ）	ATTRIBUTE ノードを作成します。
createCDATASection()（16-7 ページ）	CDATA ノードを作成します。
createComment()（16-7 ページ）	COMMENT ノードを作成します。
createDocumentFragment()（16-7 ページ）	DOCUMENT_FRAGMENT ノードを作成します。
createElement()（16-8 ページ）	ELEMENT ノードを作成します。
createEntityReference()（16-8 ページ）	ENTITY_REFERENCE ノードを作成します。
createProcessingInstruction()（16-8 ページ）	PROCESSING_INSTRUCTION ノードを作成します。
createTextNode()（16-9 ページ）	TEXT ノードを作成します。
getElementsByTagName()（16-9 ページ）	タグ名に基づいて、ノードを選択します。
getImplementation()（16-9 ページ）	ドキュメントの DTD を戻します。

createAttribute()

新しい属性ノードを作成します。setValue を使用して、その値を設定します。ノードへのポインタを戻します。

構文

```
Attr* createAttribute(String name)
```

パラメータ	説明
name	属性の名前

createCDATASection()

与えられた内容を含む新しい CDATA ノードを作成します。ノードへのポインタを戻します。

構文

```
Attr* createCDATASection(String name)
```

パラメータ	説明
data	ノードの内容

createComment()

与えられた内容を含む新しいコメント・ノードを作成します。ノードへのポインタを戻します。

構文

```
Comment* createComment(String data)
```

パラメータ	説明
data	ノードの内容

createDocumentFragment()

新しいドキュメント・フラグメント・ノードを作成します。作成されたノードへのポインタを戻します。

構文

```
DocumentFragment* createDocumentFragment()
```

createElement()

与えられた（タグの）名前を含む新しい要素ノードを作成します。ノードへのポインタを戻します。

構文

```
Element* createElement(String tagName)
```

パラメータ	説明
tagName	要素のタグ名

createEntityReference()

新しい実体参照ノードを作成します。作成されたノードへのポインタを戻します。

構文

```
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

createProcessingInstruction()

新しい処理命令ノードを作成します。作成されたノードへのポインタを戻します。

構文

```
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

createTextNode()

新しい TEXT ノードを作成します。作成されたノードへのポインタを戻します。

構文

```
Text* createTextNode(String data)
```

パラメータ	説明
data	ノードのデータ

getElementsByTagName()

与えられたタグ名を持つすべての要素のノード・リストを、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。特殊値「*」を指定すると、すべてのタグに一致します。一致するタグのリスト（一致しない場合は null）を戻します。

構文

```
NodeList* getElementsByTagName(String tagname)
```

パラメータ	説明
tagname	選択するタグ名

getImplementation()

現在は使用可能でない DOM インプリメンテーション構造を戻します。将来のバージョンの DOM で使用される可能性があります。構造に対するポインタを戻します。

構文

```
DOMImplementation* getImplementation()
```

クラス DocumentType

このクラスには、ドキュメントの DTD に関する情報へアクセスするためのメソッドが含まれます。

表 16-3 DocumentType のメソッドの概要

メソッド	説明
getName()（16-10 ページ）	DTD の名前を戻します。
getEntities()（16-10 ページ）	DTD のエンティティのマップを戻します。
getNotations()（16-10 ページ）	DTD の表記法のマップを戻します。

getName()

DTD の名前を戻します。

構文

```
String getName()
```

getEntities()

DTD の（汎用）エンティティのマップを戻します。

構文

```
NamedNodeMap* getEntities()
```

getNotations()

DTD の表記法のマップを戻します。

構文

```
NamedNodeMap* getNotations()
```

クラス DOMImplementation

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。

hasFeature()

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。機能がサポートされている場合は TRUE 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 の場合、正当な値は「HTML」および「XML」です（大文字 / 小文字は区別されません）。
version	テストするパッケージ名のバージョン番号。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、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。

クラス Element

このクラスには、要素ノードに関するメソッドが含まれます。

表 16-4 Element のメソッドの概要

メソッド	説明
getTagName()（16-12 ページ）	ノードのタグ名を戻します。
getAttribute()（16-13 ページ）	指定された名前の属性を選択します。
setAttribute()（16-13 ページ）	指定された名前および値を持つ新しい属性を作成します。
removeAttribute()（16-13 ページ）	指定された名前の属性を削除します。
getAttributeNode()（16-14 ページ）	指定された名前の属性を削除します。
setAttributeNode()（16-14 ページ）	新しい属性ノードを追加します。
removeAttributeNode()（16-14 ページ）	属性ノードを削除します。
getElementsByTagName()（16-15 ページ）	与えられたタグ名を持つ要素ノードのリストを戻します。
normalize()（16-15 ページ）	要素を正規化します（隣接するテキスト・ノードをマージします）。

getTagName()

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じです。「getNodeName」を参照してください。W3C のワーキング・グループでは、DOM 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

```
String getTagName()
```

getAttribute()

指定された名前の属性の値（定義）を戻します。

構文

```
String getAttribute(String name)
```

パラメータ	説明
name	属性の名前

setAttribute()

新しい属性を作成します。

構文

```
Attr* setAttribute(String name, String 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

removeAttribute()

指定された名前の属性を削除します。

構文

```
void removeAttribute(String name)
```

パラメータ	説明
name	削除する属性の名前

getAttributeNode()

指定された名前の属性に対するポインタを戻します。

構文

```
Attr* getAttributeNode(DOMString name)
```

パラメータ	説明
naem	属性の名前

setAttributeNode()

新しい属性を設定（追加）します。成功の場合は TRUE を戻します。

構文

```
boolean setAttributeNode(Attr* newAttr, Attr** oldAttr)
```

パラメータ	説明
newAttr	新しい属性へのポインタ
oldAttr	置換された属性に対して戻されたポインタ

removeAttributeNode()

指定された名前の属性を削除します。

構文

```
Attr* removeAttributeNode(Attr* oldAttr)
```

パラメータ	説明
oldAttr	削除する属性

getElementsByTagName()

一致する要素のリストを作成します。

構文

```
NodeList* getElementsByTagName(DOMString name)
```

パラメータ	説明
name	一致させるタグ名（すべてに一致させる場合は「*」）

normalize()

要素を正規化します（隣接するすべてのテキスト・ノードをマージします)。

構文

```
void normalize(void)
```

クラス Entity

このクラスは、**Node** のサブクラスである ENTITY ノード・タイプを実装します。

表 16-5 Entity のメソッドの概要

メソッド	説明
getNotationName()（16-16 ページ）	エンティティの NDATA（表記法名）を戻します。
getPublicId()（16-16 ページ）	エンティティの公開識別子を戻します。
getSystemId()（16-16 ページ）	エンティティのシステム識別子を戻します。

getNotationName()

エンティティ・ノードの表記法名 NDATA を戻します。

構文

```
String* getNotationName()
```

getPublicId()

エンティティ・ノードの公開識別子を戻します。

構文

```
String getPublicId()
```

getSystemId()

エンティティ・ノードのシステム識別子を戻します。

構文

```
String getSystemId()
```

クラス EntityReference

このクラスは、Node のサブクラスである ENTITY_REFERENCE ノード・タイプを実装します。

クラス NamedNodeMap

このクラスには、ノード・マップ内のノード数にアクセスし、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。

表 16-6 NamedNodeMap のメソッドの概要

メソッド	説明
item()（16-18 ページ）	マップ内の *n* 番目のノードを戻します。
getLength()（16-18 ページ）	マップ内のノード数を戻します。
getNamedItem()（16-19 ページ）	名前でノードを選択します。
setNamedItem()（16-19 ページ）	ノードをマップに設定します。
removeNamedItem()（16-19 ページ）	マップから指定のノードを削除します。

item()

ノード・マップ内の 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	0（ゼロ）を基準としたノード数

getLength()

マップ内のノード数を戻します。

構文

```
size_t getLength()
```

getNamedItem()

マップから指定された名前のノードを選択します。

構文

```
Node* getNamedItem(String name)
```

パラメータ	説明
name	選択するノードの名前

setNamedItem()

ノードをマップに追加し、同じ名前の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（新規ノードの場合は null）

removeNamedItem()

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。

構文

```
Node* removeNamedItem(String name)
```

パラメータ	説明
name	削除するノードの名前

クラス Node

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。

表 16-7 Node のメソッドの概要

メソッド	説明
appendChild()（16-21 ページ）	現行ノードの子のリストの最後に、新しい子ノードを追加します。
cloneNode()（16-21 ページ）	既存のノード、およびそのすべての子（オプション）を複製します。
getAttributes()（16-22 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()（16-22 ページ）	指定されたノードの索引付けされた特定の子ノードを戻します。
getChildNodes()（16-22 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。
getFirstChild()（16-22 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()（16-23 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()（16-23 ページ）	ノードのローカル名を戻します。
getNamespace()（16-23 ページ）	ノードの名前空間を戻します。
getNextSibling()（16-23 ページ）	ノードの次の兄弟関係を戻します。
getName()（16-23 ページ）	ノード名を戻します。
getType()（16-24 ページ）	ノードの数値型コードを戻します。
getValue()（16-24 ページ）	ノードの値（データ）を戻します。
getOwnerDocument()（16-24 ページ）	ノードを含むドキュメント・ノードを戻します。
getParentNode()（16-24 ページ）	指定されたノードの親ノードを戻します。
getPrefix()（16-25 ページ）	ノードの名前空間の接頭辞を戻します。
getPreviousSibling()（16-25 ページ）	現行ノードの以前の兄弟関係を戻します。
getQualifiedName()（16-25 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。
hasAttributes()（16-25 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()（16-25 ページ）	ノードに子ノードがあるかどうかを判別します。

表 16-7 Node のメソッドの概要（続き）

メソッド	説明
insertBefore()（16-26 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()（16-26 ページ）	子ノード数のカウントを戻します。
removeChild()（16-26 ページ）	現行ノードから子ノードを削除します。
replaceChild()（16-27 ページ）	子ノードを別のノードと置き換えます。
setValue()（16-27 ページ）	ノードの値（データ）を設定します。

appendChild()

現行ノードの子のリストに、新しい子ノードを追加します。

構文

```
Node* appendChild(Node *newChild)
```

パラメータ	説明
newChild	新しい子ノード

cloneNode()

この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、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、子であるテキスト・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。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。

構文

```
Node* cloneNode(boolean deep)
```

パラメータ	説明
deep	再帰フラグ

getAttributes()

ノードのすべての属性構造を戻します。

構文

```
NamedNodeMap* getAttributes()
```

getChildNode()

ノードの子ノードの 1 つを戻します。

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()

ノードの子ノードを戻します。

構文

```
NodeList* getChildNodes()
```

getFirstChild()

ノードの最初の子ノードを戻します。

構文

```
Node* getFirstChild()
```

getLastChild()

ノードの最後の子ノードを戻します。

構文

```
Node* getLastChild()
```

getLocal()

ノードのローカル名を戻します。

構文

```
String getLocal()
```

getNamespace()

ノードの名前空間を戻します。

構文

```
String getNamespace()
```

getNextSibling()

ノードの次の兄弟関係を戻します。

構文

```
Node* getNextSibling()
```

getName()

ノード名を戻します。ノードに名前が付いていない場合は、null を戻します。

構文

```
String getName()
```

getType()

ノードの数値型コードを戻します。これらには、ELEMENT_NODE、ATTRIBUTE_NODE、TEXT_NODE、CDATA_SECTION_NODE、ENTITY_REFERENCE_NODE、ENTITY_NODE、PROCESSING_INSTRUCTION_NODE、COMMENT_NODE、DOCUMENT_NODE、DOCUMENT_TYPE_NODE、DOCUMENT_FRAGMENT_NODE、NOTATION_NODE があります。

構文

```
short getType()
```

getValue()

ノードの値（データ）を戻します。ノードに値がない場合は、null を戻します。

構文

```
String getValue()
```

getOwnerDocument()

現行ノードを含むドキュメント・ノードを戻します。

構文

```
Document* getOwnerDocument()
```

getParentNode()

ノードの親を戻します。

構文

```
Node* getParentNode()
```

getPrefix()

ノードの名前空間の接頭辞を戻します。

構文

```
String getPrefix()
```

getPreviousSibling()

ノードの以前の兄弟関係を戻します。

構文

```
Node* getPreviousSibling()
```

getQualifiedName()

ノードの完全修飾名（名前空間）を戻します。

構文

```
String getQualifiedName()
```

hasAttributes()

ノードに定義済属性があるかどうかを判別します。

構文

```
boolean hasAttributes()
```

hasChildNodes()

ノードに子ノードがあるかどうかを判別します。

構文

```
boolean hasChildNodes()
```

insertBefore()

新しい子ノードを、子ノードのリストの参照ノードの前に挿入します。

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（このノードの前に新しいノードを挿入します）

numChildNodes()

ノードの子ノードのカウントを戻します。

構文

```
uword numChildNodes()
```

removeChild()

現行ノードの子のリストから子ノードを削除します。

構文

```
Node* removeChild(Node *oldChild)
```

パラメータ	説明
oldChild	削除中の古いノード

replaceChild()

子ノードのリスト内のノードを他のノードで置換します。

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()

ノードの値（データ）を設定します。

構文

```
void setValue(String data)
```

パラメータ	説明
data	ノードの新しいデータ

クラス NodeList

このクラスには、NodeList からノードを抽出するためのメソッドが含まれます。

表 16-8 NodeList のメソッドの概要

メソッド	説明
item()（16-28 ページ）	リスト内の n 番目のノードを戻します。
getLength()（16-28 ページ）	リスト内のノード数を戻します。

item()

ノード・リスト内の 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	0（ゼロ）を基準としたノード数

getLength()

リスト内のノード数を戻します。

構文

```
size_t getLength()
```

クラス Notation

このクラスは、**Node** のサブクラスである NOTATION ノード・タイプを実装します。

表 16-9 Notation のメソッドの概要

メソッド	説明
getData()（16-29 ページ）	表記法のデータを戻します。
getTarget()（16-29 ページ）	表記法のターゲットを戻します。
setData()（16-29 ページ）	表記法のデータを設定します。

getData()

表記法のデータを戻します。

構文

```
String getData()
```

getTarget()

表記法のターゲットを戻します。

構文

```
String getTarget()
```

setData()

表記法のデータを設定します。

構文

```
void setData(String data)
```

パラメータ	説明
data	新しいデータ

クラス ProcessingInstruction

Node のサブクラスである PROCESSING_INSTRUCTION ノード・タイプを実装します。

表 16-10 ProcessingInstruction のメソッドの概要

メソッド	説明
getData()（16-30 ページ）	処理命令のデータを戻します。
getTarget()（16-30 ページ）	処理命令のターゲットを戻します。
setData()（16-30 ページ）	処理命令のデータを設定します。

getData()

処理命令のデータを戻します。

構文

```
String getData()
```

getTarget()

処理命令のターゲット値を戻します。

構文

```
String getTarget()
```

setData()

処理命令のデータを設定します。

構文

```
void setData(String data)
```

パラメータ	説明
data	処理命令の新しいデータ

クラス Text

テキスト・ノードに対応付けられたデータにアクセスし、変更します（サブクラス CharacterData）。

splitText()

テキスト・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、次の新しいテキスト・ノードに含まれます。新しいテキスト・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

クラス XMLParser

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。

表 16-11 XMLParser のメソッドの概要

メソッド	説明
xmlinit()（16-33 ページ）	XML パーサーを初期化します。
xmlterm()（16-33 ページ）	XML パーサーを終了します。
xmlparse()（16-34 ページ）	ファイルからドキュメントを解析します。
xmlparseBuffer()（16-34 ページ）	バッファからドキュメントを解析します。
getContent()（16-35 ページ）	要素の内容モデルを戻します。
getModifier()（16-35 ページ）	内容モデル・ノードの修飾子を戻します。
getDocument()（16-35 ページ）	解析済ドキュメントのルート・ノードを戻します。
getDocumentElement()（16-36 ページ）	解析済ドキュメントのルート要素を戻します。
getDocType()（16-36 ページ）	ドキュメント・タイプ文字列を戻します。
isStandalone()（16-36 ページ）	スタンドアロン・フラグの値を戻します。
isSingleChar()（16-36 ページ）	ドキュメントのエンコーディングを判別します。
getEncoding()（16-37 ページ）	ドキュメントの文字コードの名前を戻します。

xmlinit()

XML パーサーを初期化します。エラー・コード（成功の場合は 0（ゼロ））を戻します。

構文

```
uword xmlinit(oratext *encoding,
              void (*msghdlr)(void *msgctx, oratext *msg, ub4 errcode),
              void *msgctx, lpxsaxcb *saxcb, void *saxcbctx, oratext *lang)
```

パラメータ	説明
encoding	入力ファイルのエンコーディング（デフォルトは UTF8）
msghdlr	エラー・メッセージのコールバック
msgctx	msghdlr 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）
saxcbctx	SAX コールバック関数に渡される、ユーザー定義の SAX コンテキスト構造
lang	エラー・メッセージの言語（使用されません）

xmlterm()

XML パーサーの終了、停止、メモリーの解放などを行います。

構文

```
void xmlterm()
```

xmlparse()

ドキュメントを解析します。エラー・コード（成功の場合は 0（ゼロ））を戻します。

構文

```
uword xmlparse(oratext *doc, oratext *encoding, ub4 flags)
```

パラメータ	説明
doc	ドキュメントのパス
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 ■ XML_FLAG_VALIDATE - DTD に対するドキュメントの検証 ■ XML_FLAG_DISCARD_WHITESPACE - 無視できる空白の削除

xmlparseBuffer()

ドキュメントを解析します。エラー・コード（成功の場合は 0（ゼロ））を戻します。

構文

```
uword xmlparseBuffer(oratext *buffer, size_t len, oratext *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 ■ XML_FLAG_VALIDATE - DTD に対するドキュメントの検証 ■ XML_FLAG_DISCARD_WHITESPACE - 無視できる空白の削除

getContent()

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 であり、解析済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調査できます。

構文

```
Node* getContent(Node *node)
```

パラメータ	説明
node	修飾子を戻す内容モデルのノード

getModifier()

内容モデル・ノードの修飾子を戻します。修飾子は、XMLCPMOD_NONE（修飾子なし）、XMLCPMOD_OPT（オプションの「?」）、XMLCPMOD_0MORE（0（ゼロ）個以上の「*」）または XMLCPMOD_1MORE（1 個以上の「+」）のいずれかです。

構文

```
xmlcpmod getContent(Node *node)
```

パラメータ	説明
node	修飾子を戻す内容モデルのノード

getDocument()

ドキュメントが正常に解析された後、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。ルート要素（ノード）を戻す getDocumentElement と比較してください。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。

構文

```
Node* getDocument()
```

getDocumentElement()

解析済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()
```

戻り値

`Element*` - ドキュメントのルート要素（ノード）に対するポインタ

getDocType()

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()
```

isStandalone()

ドキュメントが、`<?xml?>` 行上にスタンドアロンとして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。スタンドアロン・フラグの値を戻します。

構文

```
boolean isStandalone()
```

isSingleChar()

現行のドキュメントをシングルバイト・キャラクタ（ASCII）またはマルチバイト・キャラクタ（UTF-8）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。ドキュメントのエンコーディングの実際の名前を戻す `getEncoding` と比較してください。

構文

```
boolean isSingleChar()
```

getEncoding()

現行のドキュメントの文字コード体系の名前（ASCII、UTF8 など）を戻します。現行のエンコーディングがシングルバイトかマルチバイトかを示すブーリアン・フラグのみを戻す isSingleChar と比較してください。

構文

```
String getEncoding()
```

C++ SAX API

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・コールバック関数が起動されます。各コールバック関数は、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 を使用するために、xmlsaxcb 構造はファンクション・ポインタで初期化され、xmlinit() コールに渡されます。ユーザー定義のコンテキスト構造体へのポインタも含まれるため、コンテキスト・ポインタは各 SAX 関数に渡されます。

SAX コールバック構造

```
typedef struct
{
   sword (*)(void *ctx);
   sword (*)(void *ctx);
   sword (*)(void *ctx, const oratext *name, struct xmlarray *attrs);
   sword (*)(void *ctx, const oratext *name);
   sword (*)(void *ctx, const oratext *ch, size_t len);
   sword (*)(void *ctx, const oratext *ch, size_t len);
   sword (*)(void *ctx, const oratext *target, const oratext *data);
   sword (*)(void *ctx, const oratext *name,
                         const oratext *publicId, const oratext *systemId);
   sword (*)(void *ctx, const oratext *name, const oratext *publicId,
                               const oratext *systemId, const oratext
*notationName);
   sword (*)(void *ctx, const oratext *qname,
                           const oratext *local, const oratext *namespace);
} xmlsaxcb;
```

表 16-12 C++ SAX のメソッドの概要

メソッド	説明
startDocument() (16-39 ページ)	ドキュメントの処理を開始します。
endDocument() (16-39 ページ)	ドキュメントの処理を終了します。
startElement() (16-40 ページ)	新しい各要素の処理を開始します。
endElement() (16-40 ページ)	新しい各要素の処理を終了します。
characters() (16-41 ページ)	リテラル・テキストのピースを処理します。
IgnorableWhitespace() (16-41 ページ)	重要でない空白のピースを処理します。

表 16-12 C++ SAX のメソッドの概要（続き）

メソッド	説明
processingInstruction()（16-42 ページ）	処理命令を処理します。
notationDecl()（16-42 ページ）	表記法を処理します。
unparsedEntityDecl()（16-43 ページ）	エンティティを解析します。
nsStartElement()（16-43 ページ）	明示的な名前空間を使用したドキュメントのドキュメント処理を開始します。

startDocument()

ドキュメントの処理が最初に開始されたときに、1 回コールされます。エラー・コード（成功の場合は 0（ゼロ）、エラーの場合はそれ以外）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()

ドキュメントの処理が終了したときに、1 回コールされます。エラー・コード（成功の場合は 0（ゼロ）、エラーの場合はそれ以外）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()

新しいドキュメント要素ごとにコールされます。エラー・コード（成功の場合は 0（ゼロ）、エラーの場合はそれ以外）を戻します。

構文

```
sword startElement(void *ctx, const oratext *name, struct xmlarray *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	ノードの名前
attrs	ノードの属性の配列

endElement()

ドキュメント要素がクローズするたびにコールされます。エラー・コード（成功の場合は 0（ゼロ）、エラーの場合はそれ以外）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	ノードの名前

characters()

リテラル・テキストのピースごとにコールされます。エラー・コード（成功の場合は 0（ゼロ）、エラーの場合はそれ以外）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	テキストの文字数

IgnorableWhitespace()

無視できる（重要でない）空白のピースごとにコールされます。エラー・コード（成功の場合は 0（ゼロ）、エラーの場合はそれ以外）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	空白の文字数

processingInstruction()

処理命令ごとにコールされます。エラー・コード（成功の場合は 0（ゼロ）、エラーの場合はそれ以外）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
target	処理命令のターゲット
data	処理命令のデータ

notationDecl()

表記法ごとにコールされます。エラー・コード（成功の場合は 0（ゼロ）、エラーの場合はそれ以外）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	表記法の名前
publicId	公開識別子
systemId	システム識別子

unparsedEntityDecl()

解析されていないエンティティ宣言ごとにコールされます。エラー・コード（成功の場合は 0（ゼロ）、エラーの場合はそれ以外）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
publicId	公開識別子
systemId	システム識別子
notationName	表記法の名前

nsStartElement()

startElement の名前空間のオプションです。要素が明示的な名前空間を使用する場合、新しいドキュメント要素ごとにコールされます。エラー・コード（成功の場合は 0（ゼロ）、エラーの場合はそれ以外）を戻します。

構文

```
sword startElement(void *ctx, const oratext *qname,
                   const oratext *local, const oratext *namespace)
```

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	ローカル名
namespace	名前空間

17

Oracle XML Class Generator for C++

この章では、XML Class Generator for C++ が各要素を定義する方法を説明します。この章の内容は次のとおりです。

XML Class Generator は、DTD 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。

この章の内容は次のとおりです。

関連項目：

- 『Oracle アプリケーション開発者ガイド - XML』

Class Generator for C++ の使用

入力は、DTD を含む XML 文書です。ドキュメント本体自体は無視されます。DTD のみが関連しますが、ドキュメントは DTD に準拠する必要があります。基礎となる XML パーサーのみが、ドキュメントのファイル名およびそれに対応付けられた外部エンティティを受け入れます。サポートされる入力ファイルのエンコーディングは、UTF-8、UTF-16、US-ASCII、ISO-10646-UCS-2、ISO-8859-1、ISO-8859-2、ISO-8859-3、ISO-8859-4、ISO-8859-5、ISO-8859-6、ISO-8859-7、ISO-8859-8、ISO-8859-9、EUC-JP、SHIFT_JIS、BIG5、GB2312、KOI8-R、EBCDIC-CP-US、EBCDIC-CP-CA、EBCDIC-CP-NL、EBCDIC-CP-WT、EBCDIC-CP-DK、EBCDIC-CP-NO、EBCDIC-CP-FI、EBCDIC-CP-SE、EBCDIC-CP-IT、EBCDIC-CP-ES、EBCDIC-CP-GB、EBCDIC-CP-FR、EBCDIC-CP-HE、EBCDIC-CP-BE、EBCDIC-CP-CH、EBCDIC-CP-ROECE、EBCDIC-CP-YU および EBCDIC-CP-IS です。

デフォルトでは、UTF-8 がエンコーディングになります。シングルバイト・キャラクタ・セット（US-ASCII、ISO-8859 キャラクタ・セットのいずれか）のみを使用している場合、明示的にデフォルトのエンコーディングを設定することをお薦めします。UTF-8 などのマルチバイト・キャラクタ・セットよりも、パフォーマンスが 25% 向上します。

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。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。

関連する標準には、eXtensible Markup Language（XML）1.0、DOM レベル 1 1.0、XML 名前空間および Simple API for XML（SAX）1.0 に関する W3C の勧告があります。

例

スタンドアロン・パーサーは、`bin/xmlcg like xmlcg [flags] <XML document>` を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。表 17-1 に、オプション・フラグを示します。

表 17-1 オプション・フラグ

フラグ	説明
`-d`	出力ディレクトリを指定します。デフォルトは現在のディレクトリです。
`-e`	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`	ヘルプの手順を表示します。

クラス XMLClassGenerator

DTD に基づいてクラスを生成します。

generate()

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 後に名前付けされた DTDname.h と DTDname.cpp の 2 つのファイルが作成されます。DTD の定義済要素ごとに、1 つのクラスが生成されます。uword エラー・コード（成功の場合は 0（ゼロ））を戻します。

構文

```
uword generate(DocumentType *dtd, char *outdir)
```

パラメータ	説明
dtd	クラスを生成するために使用される DTD ソース
outdir	生成されたファイルの出力ディレクトリ

クラス generated

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。

空の要素を作成したり、子やデータの初期セットを含む要素を作成するためのコンストラクタが提供されます。メソッドは、構成後に子またはデータを追加したり、属性を設定するために提供されます。要素を作成するには、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。たとえば、要素宣言 <!ELEMENT B (#PCDATA | F)*> が指定されると、次のコンストラクタが提供されます。

```
B(Document *doc);
B(Document *doc, String data);
B(Document *doc, F *theF);
```

最初のコンストラクタは、子を含まない空の要素のみを作成します。2 つ目のコンストラクタは、PCDATA を使用して要素を初期化し、3 つ目のコンストラクタは、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、構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 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");
```

と

```
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 = new F(...);
b = new B(f);
```

と

```
f = new F(...);
b = new B();
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、要素 <!ELEMENT Sample (A* | (B, (C? | (D, E)*)) | F)+> の場合、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、最初に空の要素を作成し、必要に応じて前述の 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が提供されます。たとえば、要素宣言 <!ELEMENT D (#PCDATA)> ... <!ATTLIST D foo CDATA #REQUIRED> の場合、クラス D にはメソッド Attr* setfoo(String value) が提供されます。構成された要素は、作成中に妥当性がテストされません。ユーザーは、結果の要素に対して、XMLParser の validate メソッドを明示的にコールする必要があります。

表 17-2 generated のメソッドの概要

メソッド	説明
Constructor()（17-5 ページ）	ドキュメントに属する要素を構成します。
addData()（17-6 ページ）	要素にデータを追加します。
addNode()（17-6 ページ）	要素に子ノードを追加します。
setattribute()（17-6 ページ）	要素の属性値を設定します。

Constructor()

ドキュメントに属する要素を構成します。フォーム 1 は、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移入します。フォーム 2 は、要素の定義に応じて、初期のデータまたは子を提供します。

構文

```
class(Document *doc)
class(Document *doc, ...)
```

パラメータ	説明
doc	要素が属するドキュメント。
...	引数のリスト。これらは要素の定義によって異なります。

addData()

要素にデータを追加します。要素に、任意の値を含む PCDATA サブノードを追加します。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、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()

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 thenode)
```

パラメータ	説明
node	追加されるノード

setattribute()

要素の属性値を設定します。各属性には、「set 属性名」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。

構文

```
Attr* setattribute(String value)
```

パラメータ	説明
value	属性の値

索引

X

い

く

は