Oracle9*i*

Java パッケージ・プロシージャ・リファレンス

リリース 2（9.2）

2002 年 7 月

部品番号：J06294-01

ORACLE®

Oracle9*i* Java パッケージ・プロシージャ・リファレンス , リリース 2（9.2）

部品番号：J06294-01

原本名：Oracle9*i* Supplied Java Packages Reference, Release 2 (9.2)

原本部品番号：A96610-01（Vol.1）、A96611-01（Vol.2）

原本著者：Cathy Shea

原本協力者：K Karun, Bhushan Khaladkar, Roza Leyderman, Vivek Maganty, Mehta Megna, Jack Melnick, Bhagat Nainani, Denis Raphaely, Jim Warner

Printed in Japan.

目次

3 パッケージ oracle.AQ.xml

4 パッケージ oracle.jms

5 パッケージ oracle.ODCI

7 パッケージ oracle.xml.parser.schema

8 パッケージ oracle.xml.sql.dml

9 パッケージ oracle.xml.sql.query

10 パッケージ oracle.xml.util

11 パッケージ oracle.xml.parser.v2

第 III 部　Oracle9*i* XDK for JavaBeans 用の Java パッケージ

12　パッケージ oracle.xml.async

13 パッケージ oracle.xml.dbviewer

14 パッケージ oracle.xml.differ

15 パッケージ oracle.xml.srcviewer

16 パッケージ oracle.xml.transviewer

17 パッケージ oracle.xml.treeviewer

第 IV 部　Oracle SOAP 用の Java パッケージ

18　パッケージ oracle.soap.server

19　パッケージ oracle.soap.transport

20　パッケージ oracle.soap.transport.http

21　パッケージ oracle.soap.util.xml

第 V 部　Oracle XML DB 用の Java パッケージ

22　パッケージ oracle.xdb.dom

23　パッケージ oracle.xdb.spi

索引

はじめに

ここでは、次の項目について説明します。

このマニュアルの内容

このマニュアルに含まれるほとんどの情報は、Java API 用のドキュメントの正確性と完全性を確認するために、Java ソース・パッケージから解析されています。ただし、マニュアルの完成間際での更新データは、個々のクラス（.java files）から解析され、クラス階層内に挿入されている場合があります。変更が小規模な場合は、パッケージ全体または個々のクラスが手動で更新されています。Java パッケージは様々に異なる要件を伴う機能を実装しているため、API のドキュメントもそれに応じて多様化しています。また、Java API を作成する開発者が意図する情報に比較して、フォーマットは二次的な情報となります。このため、API の提示方法に若干の違いがある場合があります。Java API ドキュメントの自動作成については、`http://java.sun.com/` を参照してください。

対象読者

このマニュアルは、Java プログラマおよび Oracle9*i* リリース 2（9.2）用のデータベース・アプリケーションの開発者を対象としています。このマニュアルは、読者にクライアント / サーバーを採用している企業でのアプリケーション・プログラミングの実践経験があり、リレーショナル・データベース・システムの情報にアクセスして操作するための Java と SQL の使用経験が十分にあることを前提としています。また、XML 機能と Oracle XML DB を実装する Java クラスを活用するには、World Wide Web Consortium（W3C）が規定する XML 標準についての十分な理解も重要です。さらに、Web 開発とオブジェクト・リレーショナル・データベース・システムについての知識も役立ちます。

このマニュアルの構成

このリファレンス・マニュアルは 5 部に分かれています。各部および関連する章は次のとおりです。

第 I 部「Oracle9i RDBMS 用の Java パッケージ」

第 I 部は、Oracle RDBMS 用の Java API を実装する Java パッケージに関する章で構成されています。

第 II 部「Oracle9i XDK for Java 用の Java パッケージ」

第 II 部では、Oracle XDK for Java に含まれている Java パッケージについて説明します。

第 III 部「Oracle9i XDK for JavaBeans 用の Java パッケージ」

第 III 部では、Oracle XDK for JavaBeans に含まれている Java パッケージについて説明します。

第 IV 部「Oracle SOAP 用の Java パッケージ」

第IV部は、XDK for Java で Oracle SOAP を実装する Java パッケージに関する章で構成されています。

第 V 部「Oracle XML DB 用の Java パッケージ」

第V部は、『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』で説明されている機能を実装する Java パッケージに関する章で構成されています。これらの Java API は、Oracle XML DB を使用して Oracle9*i* 内でネイティブに実行する XML アプリケーションの開発をサポートします。

関連文書

詳細は、次の Oracle リソースを参照してください。

- 『Oracle9*i* JDBC 開発者ガイドおよびリファレンス』
- 『Oracle9*i* アプリケーション開発者ガイド - 基礎編』
- 『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』
- 『Oracle9*i* Data Cartridge Developer's Guide』
- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』

このマニュアルの多くの例で、Oracle のインストール時にデフォルトでインストールされるシード・データベースのサンプル・スキーマを使用しています。これらのスキーマがどのように作成されているか、およびその使用方法については、『Oracle9*i* サンプル・スキーマ』を参照してください。

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

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

OTN-J のユーザー名とパスワードを取得済みであれば、次の OTN-J Web サイトの文書セクションに直接接続できます。

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

その他の情報は、次の URL にアクセスしてください。

- http://www.w3c.org
- http://java.sun.com

表記規則

このマニュアル・セットの本文とコード例に使用されている表記規則について説明します。

本文の表記規則

本文中には、特別な用語が一目でわかるように様々な表記規則が使用されています。次の表は、本文の表記規則と使用例を示しています。

規則	意味	例
太字	太字は、本文中に定義されている用語または用語集に含まれている用語、あるいはその両方を示します。	この句を指定する場合は、**索引構成表**を作成します。
固定幅フォントの大文字	固定幅フォントの大文字は、システムにより指定される要素を示します。この要素には、パラメータ、権限、データ型、Recovery Manager キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージとメソッド、システム指定の列名、データベース・オブジェクトと構造体、ユーザー名、およびロールがあります。	この句は、NUMBER 列に対してのみ指定できます。 BACKUP コマンドを使用すると、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サンプルのユーザー指定要素を示します。この要素には、コンピュータ名とデータベース名、ネット・サービス名、接続識別子の他、ユーザー指定のデータベース・オブジェクトと構造体、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があります。 **注意：**一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。	sqlplus と入力して SQL*Plus をオープンします。 パスワードは orapwd ファイルに指定されています。 データ・ファイルと制御ファイルのバックアップを /disk1/oracle/dbs ディレクトリに作成します。 department_id、department_name および location_id の各列は、hr.departments 表にあります。 初期化パラメータ QUERY_REWRITE_ENABLED を true に設定します。 oe ユーザーで接続します。 これらのメソッドは JRepUtil クラスに実装されます。

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。*old_release* は、アップグレード前にインストールしたリリースです。

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus またはその他のコマンドラインを示します。次のように、固定幅フォントで、通常の本文とは区別して記載しています。

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

次の表に、コード例の記載上の表記規則と使用例を示します。

規則	意味	例
[ ]	大カッコで囲まれている項目は、1 つ以上のオプション項目を示します。大カッコ自体は入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコで囲まれている項目は、そのうちの 1 つのみが必要であることを示します。中カッコ自体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肢を区切るために使用します。オプションのうち 1 つを入力します。縦線自体は入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	水平の省略記号は、次のどちらかを示します。	
	■ 例に直接関係のないコード部分が省略されいること。	CREATE TABLE ... AS *subquery*;
	■ コードの一部が繰返し可能であること。	SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	垂直の省略記号は、例に直接関係のない数行のコードが省略されていることを示します。	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.

規則	意味	例
その他の表記	大カッコ、中カッコ、縦線および省略記号以外の記号は、示されているとおりに入力してください。	`acctbal NUMBER(11,2);` `acct    CONSTANT NUMBER(4) := 3;`
イタリック	イタリックの文字は、特定の値を指定する必要のあるプレースホルダまたは変数を示します。	`CONNECT SYSTEM/`*`system_password`* `DB_NAME = `*`database_name`*

大文字	大文字は、システムにより指定される要素を示します。これらの用語は、ユーザー定義用語と区別するために大文字で記載されています。大カッコで囲まれている場合を除き、記載されているとおりの順序とスペルで入力してください。ただし、この種の用語は大 / 小文字区別がないため、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を示します。
	注意 : 一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Java パッケージ・プロシージャの新機能

この章では、Java パッケージ・プロシージャに導入された新機能について説明します。

- Oracle9i リリース 2（9.2）の Java パッケージ・プロシージャで導入された新機能
- Oracle JVM 用 Java クラスを更新するスクリプトの変更
- Oracle9i リリース 1（9.0.1）の Java パッケージ・プロシージャで導入された新機能

Oracle9*i* リリース 2（9.2）の Java パッケージ・プロシージャで導入された新機能

この項では、Oracle9*i* リリース 2（9.2）の Java パッケージ・プロシージャで導入された機能をリストします。

Oracle RDBMS 用 Java パッケージの新機能

Oracle データベースにおける次の機能に関して Java API が更新されています。

- **Global Context Manager**
- **Advanced Queuing**
- **Java Message Service**
- **Oracle Data Cartridge**

関連項目： 第 I 部「Oracle9i RDBMS 用の Java パッケージ」の各章

XDK for Java 用 Java パッケージの新機能

この項では、XDK for Java 用のパッケージとクラスの新機能および追加機能をリストします。

- **XML Schema Processor for Java**
  - XML Schema に関する World Wide Web Consortium（W3C）の最新の推奨事項をサポート。
- **XSQL Servlet**
  - 新規パフォーマンス改善オプション。
  - `XPath` 属性のサポート。
  - `CLOB` 列と `VARCHAR2` 列からの簡素化された XML のインクルード。
  - 転送した XML をインクルードするための新規アクション・ハンドラ。
  - Apache FOP を使用した PDF 出力のサポート。XSQL Pages を Apache FOP プロセッサと組み合せると、XML コンテンツから Adobe PDF 出力を生成できます。（FOP は、XSL フォーマット・オブジェクトによって動作する Apache の出力フォーマッタです。フォーマット・オブジェクト・ツリーを読み込み、指定された出力に結果のページをレンダリングします。）
  - Cookie として設定された値の即時読込みのサポート。
  - 単一の SQL 文による複数パラメータ値の設定のサポート。

- **Class Generator for Java**
  - DTD Class Generator への新規データ・バインド機能の追加。
  - インスタンス・データの生成済みクラスへのロードで、XML インスタンス文書を入力として提供可能。
  - SAX 2.0 に対する XSU サポートと SQL 問合せに関する XML Schema の生成。
  - DOM 圧縮に対するサポート。
  - Java XML Parser での SAX2 Extension に対するサポートの追加。
  - Java XML Parser での XML 圧縮に対するサポートの追加。
  - JAXP 1.1 に対するサポート。
  - データとテキストをロードするための Oracle TransX Utility。
  - XML Schema Processor for Java による `LAX` と `STRICT` の両モードのサポート。

    関連項目： 第 II 部「Oracle9i XDK for Java 用の Java パッケージ」の各章

XDK for JavaBeans 用 Java パッケージの新機能

この項では、XDK for JavaBeans 用のパッケージにおける新しい機能とサポートをリストします。

- **新規 XMLDiff Bean**
- **SourceViewer Bean に対する内部 DTD サポート**

  関連項目： 第 III 部「Oracle9i XDK for JavaBeans 用の Java パッケージ」の各章

XDK for Java における Oracle SOAP の新機能

このリリースでは、Oracle SOAP に対して次の更新と追加が行われています。

- **新規 Oracle SOAP API**
- **SOAP サービスに対する新規サポート**

  関連項目： 第 IV 部「Oracle SOAP 用の Java パッケージ」の各章

新規 Oracle XML DB に導入された新規 Java パッケージ

Oracle XML DB 用 XMLType の拡張

XMLType データ型が Oracle9*i* に初めて導入されました。このデータ型は、新規 Oracle XML DB をサポートするために、Oracle9*i* リリース 2（9.2）で大幅に拡張されています。

- **XMLType 表**
  - データ型 XMLType を使用して XMLType の表を作成できるようになりました。
- **XMLType コンストラクタ**
  - 追加の XMLType コンストラクタ・メソッドが追加されました。
- **W3C XML Schema サポート**
  - Oracle XML DB に対する拡張 XML Schema サポートがこのリリースで追加されました。

新規 Oracle XML DB リポジトリ

新規 Oracle XML DB リポジトリによって、すべてのデータベース・データに対して、ファイル・システムと Web アクセスが提供されます。

- **Oracle XML DB リソース API（JNDI）**
  - JNDI（Java Naming and Directory Interface）を使用して、リソースの検索およびコレクションの管理を行います。
  - JNDI Service Provider Interface（SPI）をサポートします。このインタフェースは、Oracle JVM プラットフォーム上のデータベース・サーバー内でのみ動作します。

    関連項目： 第 V 部「Oracle XML DB 用の Java パッケージ」の各章

Oracle JVM 用 Java クラスを更新するスクリプトの変更

この項は、独自のカスタム・スクリプトを作成するために Oracle スクリプトをテンプレートまたは例として使用する開発者を対象としています。このリリースでは、アップグレード・プロセスを統合するために新規スクリプトがいくつか追加されています。新規スクリプトの 1 つは rdbms/admin/catjava.sql スクリプトです。catjava.sql スクリプトは、Oracle JVM がデータベース内にある場合、Oracle9*i* リリース 2（9.2）へのアップグレード時に自動的に実行されます。

catjava.sql スクリプトは、ここにリストされているスクリプトを rdbms/admin から実行して個々のスクリプトをコールし、関連する Java クラスを次のようにロードします。

- **initapcx.sql**
  - oracle/security/rdbms/server/AppCtx/
- **initjms.sql**
  - javax/jms
    - oracle/jms
    - oracle/AQ
- **initsjty.sql**
  - oracle/aurora/sqljtype
- **initsoxx.sql**
  - oracle/CartridgeServices
    - oracle/ODCI

補足情報ですが、catjava.sql スクリプトは、サーバー機能を実装する Java クラスをロードする追加のスクリプトを他に 2 つコールします （これらは、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』に記述されているクラスとは関係ありません）。

- **initcdc.sql**
  - oracle/CDC（チェンジ・データ・キャプチャ）
- **initqsma.sql**
  - oracle/qsma（サマリー・アドバイザ）

Oracle9*i* リリース 1（9.0.1）の Java パッケージ・プロシージャで導入された新機能

この項では、Oracle9*i* リリース 1（9.0.1）の Java パッケージ・プロシージャで導入された機能をリストします。

XDK for Java

- XML Schema Processor for Java。
- DOM 2.0 と SAX 2.0 をサポートする XML Parser for Java。
- XSLT パフォーマンスの改善。
- Class Generator for Java。XML Schema ベースと DTD ベースの Class Generator を含みます。

- **XSQL Servlet と XSQL Pages**
  - データベース・バインド変数。パフォーマンス向上のために、字句置換とデータベース・バインド変数がサポートされています。
  - Apache FOP を使用した PDF 出力。
  - XSLT スタイルシート用のトラステッド・ホスト・サポート。スタイルシートは、非トラステッド・ホストからは実行できません。
  - Oracle 以外の JDBC Drivers に対する全面サポート。問合せ、挿入、更新および削除のすべての操作で、Oracle と Oracle 以外の JDBC Drivers の両方がサポートされます。
  - 動的に構成された XSQL Pages。XSQLRequest API によって、プログラムで構成された XSQL Pages が処理されます。
  - カスタム Connection Manager。独自の Connection Manager を実装し、任意の方法でデータベース接続を処理できます。
  - インライン XML Schema。XML Query 結果の構造を記述するインライン XML Schema を必要に応じて生成できます。
  - 問合せ用のデフォルトの日付書式。日付書式マスクを指定して、日付データの書式を設定するデフォルトの方法を変更できます。
  - カスタム・シリアライザ。XSQL ページ・プロセッサがクライアントに戻す内容とその方法を制御するカスタム・シリアライザを作成し、使用します。
  - スタイルシートの動的割当て。パラメータまたは SQL 問合せの結果に基づいて、スタイルシートを動的に割り当てます。
  - 転送された XML の更新または削除。XML を挿入、更新および削除します。

- ターゲット列のみの挿入または更新。挿入または更新要求の対象となる列を明示的にリストします。
- ページ要求範囲付きオブジェクト。アクション・ハンドラでは、ページ要求コンテキスト内のオブジェクトを get/set（取得 / 設定）し、アクション間の状態をページ内で共有できます。
- ServletContext へのアクセス。HttpRequest オブジェクトと HttpResponse オブジェクトにアクセスする以外に ServletContext にもアクセスできます。

- **XDK for JavaBeans**
  - DBViewer Bean。XSL スタイルシートを適用し、HTML をスクロール可能なパネルで視覚化することで、データベース問合せや任意の XML を表示します。
  - DBAccess Bean。DBAccess Bean は、複数の XML 文書とテキスト・ドキュメントが含まれた CLOB 表を保持します。

XML SQL Utility（XSU）の機能

- SQL 問合せを指定して XML Schema を生成する機能。
- XMLType と URI 参照に対するサポート。
- XML を SAX2 コールバックのストリームとして生成する機能。
- データベースから XML を生成するときの XML 属性サポート。特定の列または列グループを、XML 要素ではなく XML 属性にマップするように指定する簡単な方法を提供します。

第 I 部

Oracle9*i* RDBMS 用の Java パッケージ

第 I 部には、Oracle RDBMS 用の Java API を実装する Java パッケージに関する参照情報が記載されています。次の章で説明するパッケージは、パブリックの Java クラスとその標準に対して Oracle 固有の拡張機能を提供します。

第 I 部は、次の章で構成されています。

1

パッケージ oracle.security.rdbms.appctx の AppCtxManager

この章では、パッケージ oracle.security.rdbms.appctx で公開されるパブリックの Java クラス AppCtxManager について説明します。AppCtxManager とその関連クラスは、グローバルにアクセス（ACCESSED GLOBALLY）されるように作成された CONTEXT に対してのみ機能し、他のタイプの CONTEXT（例 : LDAP（Lightweight Directory Access Protocol）を介してグローバルに初期化（INITIALIZED GLOBALLY）されるタイプ）に対しては機能しません。

この API では、開発者のアプリケーション・コンテキストを格納するための集中化された場所が提供され、アプリケーションでユーザーのコンテキストを設定できます。

この章は、次の項で構成されています。

AppCtxManager の説明

AppCtxManager クラスはアプリケーション・コンテキストを管理します。このクラス内のメソッドのコールはすべて、グローバルにアクセスされるアプリケーション・コンテキストを管理するアプリケーション指定のクラスによって行われる必要があります。AppCtxManager クラスはインスタンス化できません。

具体的には、AppCtxManager は、グローバルにアクセス可能なアプリケーション・コンテキストを処理するための Oracle Java API を提供します。この API は、グローバルにアクセス可能なアプリケーション・コンテキスト・ネームスペースを管理できる、ユーザー定義の Java クラスを指定します。AppCtxManager は、Oracle Label Security ラベルをサポートします。この機能によって、管理者は、企業内の多数のユーザーとデータベースに対するコンテキストを管理できます。

『Oracle9*i* アプリケーション開発者ガイド - 基礎編』には、グローバルにアクセスされるアプリケーション・コンテキストの使用方法とその機能に関する詳細な情報が記載されています。

> **関連項目：** この機能の詳細は、『Oracle9*i* アプリケーション開発者ガイド - 基礎編』を参照してください。また、関連する PL/SQL パッケージ・プロシージャ DBMS_APPCTX の詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

クラス階層

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

java.lang.Object
  |
  +--oracle.security.rdbms.server.AppCtx.AppCtxManager
```

AppCtxManager メソッド

表 1-1　AppCtxManager メソッドの概要

メソッド	説明
clearContext(AppCtxPermit, String, String, String)	AppCtxPermit オブジェクトをチェックして、ユーザーがコンテキストを消去できるようにします。
createAppCtxPermit()	AppCtxPermit オブジェクトを戻します。
setContext(AppCtxPermit, String, String, String, String, String)	AppCtxPermit オブジェクトをチェックして、ユーザーがコンテキストを設定できるようにします。

clearContext(AppCtxPermit, String, String, String)

説明

このメソッドは、AppCtxPermit オブジェクトをチェックして、ユーザーがコンテキストを消去できるようにします。

構文

```
public static void clearContext(AppCtxPermit permit, java.lang.String namespace,
java.lang.String client_id, java.lang.String attribute)
```

パラメータ

`permit` ― アプリケーション・コンテキストを管理するように指定されたクラスに関する情報を格納する AppCtx オブジェクト

`namespace` ― 名前空間

`client_id` ― セッションのクライアント識別子

`attribute` ― 属性

createAppCtxPermit()

説明

このメソッドは、AppCtxPermit オブジェクトを戻します。

ユーザーは次のようにグローバル・アクセス・コンテキストを作成できます。

```
CREATE CONTEXT hr using HR.initclass ACCESSED GLOBALLY;
```

Java API を使用して HR アプリケーション・コンテキストを管理する場合、ユーザーは AppCtxPermit オブジェクトを使用する必要があります。有効な AppCtxPermit オブジェクトを作成できる唯一のクラスは、前述の CREATE CONTEXT 構文で指定した HR アプリケーション・スキーマ内の HR.initclass クラスです。AppCtxPermit オブジェクトは、HR コンテキストを管理するためのトラスト・ポイントになります。

構文

```
public static AppCtxPermit createAppCtxPermit()
```

パラメータ

なし

戻り値

AppCtxPermit オブジェクト

setContext(AppCtxPermit, String, String, String, String, String)

説明

このメソッドは、AppCtxPermit オブジェクトをチェックして、ユーザーがコンテキストを設定できるようにします。

構文

```
public static void setContext(AppCtxPermit permit, java.lang.String namespace,
java.lang.String attribute, java.lang.String value, java.lang.String username,
java.lang.String client_id)
```

パラメータ

permit ― アプリケーション・コンテキストを管理するように指定されたクラスについての情報を格納する AppCtx オブジェクト

namespace ― 名前空間

attribute ― 属性

value ― 属性の値

username ― クライアントを表示できるユーザーのユーザー名

client_id ― セッションのクライアント識別子

戻り値

なし

AppCtxManager の例

次に示す例はサンプルの Java クラスです。loadjava ツールを使用してデータベースにロードできます。

```
SQL> CREATE CONTEXT ctx1  using ctxj.employee ACCESSED GLOBALLY;

/* The java class */
import java.sql.*;
import oracle.sql.*;
import oracle.jdbc2.*;
import oracle.jdbc.driver.*;
import oracle.security.rdbms.server.AppCtx.*;
import java.util.ResourceBundle;

class Employee
{
  public static void setctx9() throws Exception
  {
   try
   {
    AppCtxPermit appCtxPermit = AppCtxManager.createAppCtxPermit() ;
    AppCtxManager.setContext(appCtxPermit, "Ctx1","Attr1", "9","ctxj","10");
   }
   catch(Exception e)
   {
    e.printStackTrace() ;
     throw new Exception(e.toString());
   }
}
public static void clrctx4() throws Exception
{
 try
  {
    AppCtxPermit appCtxPermit = AppCtxManager.createAppCtxPermit() ;
    AppCtxManager.clearContext(appCtxPermit, "Ctx1", "10", "Attr1") ;
   }
   catch(Exception e)
   {
    e.printStackTrace() ;
     throw new Exception(e.toString());
    }
 }
}
```

```
/* load the java class into the database */
loadjava -resolve -resolver "((* CTXJ) (* SYS))" -v -u ctxj/ctxj Employee.java
```

2

パッケージ oracle.AQ

この章では、パッケージ oracle.AQ に含まれる Oracle Java インタフェースとクラスについて説明します。これらは、Oracle Advanced Queuing（AQ）の現行の PL/SQL インタフェースに基づいています。

この章は、次の項で構成されています。

パッケージ oracle.AQ の説明

Java AQ API は、Oracle Advanced Queuing の管理機能と操作機能の両方をサポートします。メッセージ・アプリケーション用の Java プログラムの開発では、JDBC を使用してデータベースへの接続をオープンしてから、メッセージ・キューイング用の Java AQ API を含んでいる oracle.AQ のインタフェースをオープンします。PL/SQL インタフェースのみを使用する必要はありません。

注意： Java クラスが事前ロードされていない場合は、SYS で接続して $ORACLE_HOME/rdbms/admin/initjms.sql スクリプトをロードすることによって、Java クラスをロードできます。

Java AQ クラスへのアクセス

Java AQ クラスは、`$ORACLE_HOME/rdbms/jlib/aqapi.jar` にあります。Oracle9*i* リリース 2（9.2）の `rdbms/jlib/*.jar` は、Sun 社が公開している JMS 1.0.2b 標準に準拠しています。このクラスは、任意の Oracle8*i* または Oracle9*i* の JDBC ドライバで使用できます。

JDK 1.3 の場合は、次のクラスを CLASSPATH に挿入する必要があります。

```
$ORACLE_HOME/rdbms/jlib/aqapi13.jar
$ORACLE_HOME/lib/jndi.jar
$ORACLE_HOME/jdbc/lib/classes12.zip
```

JDK 1.2 の場合は、次のクラスを CLASSPATH に挿入する必要があります。

```
$ORACLE_HOME/rdbms/jlib/aqapi12.jar
$ORACLE_HOME/lib/jndi.jar
$ORACLE_HOME/jdbc/lib/classes12.zip
```

JDK 1.1 の場合は、次のクラスを CLASSPATH に挿入する必要があります。

```
$ORACLE_HOME/rdbms/jlib/aqapi11.jar
$ORACLE_HOME/lib/jndi.jar
$ORACLE_HOME/jdbc/lib/classes111.zip
```

『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』の付録 A には、この章の例以外の例が紹介されています。

oracle.AQ の設定例

1. oracle.AQ ユーザー aqjava の作成

aqjava ユーザーの設定方法は、次のとおりです。

```
CONNECT sys/change_on_install AS sysdba

DROP USER aqjava CASCADE;
GRANT CONNECT, RESOURCE, AQ_ADMINISTRATOR_ROLE TO aqjava
   IDENTIFIED BY aqjava;
GRANT EXECUTE ON SYS.DBMS_AQADM TO aqjava;
GRANT EXECUTE ON SYS.DBMS_AQ TO aqjava;
GRANT EXECUTE ON SYS.DBMS_AQIN TO aqjava;
CONNECT aqjava/aqjava
```

2. メイン・クラスの設定

次に、メイン・クラスを設定します。このクラスから以降の例をコールし、例外を処理します。この例でのメイン・クラスの名前は test_aqjava です。

```
import java.sql.*;
import oracle.AQ.*;

public class test_aqjava
{
   public static void main(String args[])
   {
      AQSession  aq_sess = null;

      try
      {
         aq_sess = createSession(args);

        /* now run the test: */
        runTest(aq_sess);
      }
      catch (Exception ex)
      {
         System.out.println("Exception-1: " + ex);
         ex.printStackTrace();
      }
   }
}
```

3. AQ セッションの作成

次に、後述の AQDriverManager の手順に従って、aqjava ユーザー用に AQ セッションを作成します。

```
public static AQSession createSession(String args[])
   {
      Connection db_conn;
      AQSession  aq_sess = null;

      try
      {

         Class.forName("oracle.jdbc.driver.OracleDriver");
         /* your actual hostname, port number, and SID will
         vary from what follows. Here we use 'dlsun736,' '5521,'
         and 'test,' respectively: */

         db_conn =
                  DriverManager.getConnection(
                  "jdbc:oracle:thin:@dlsun736:5521:test",
                  "aqjava", "aqjava");

         System.out.println("JDBC Connection opened ");
         db_conn.setAutoCommit(false);

         /* Load the Oracle9i AQ driver: */
         Class.forName("oracle.AQ.AQOracleDriver");
         /* Create an AQ Session: */
         aq_sess = AQDriverManager.createAQSession(db_conn);
         System.out.println("Successfully created AQSession ");
      }
      catch (Exception ex)
      {
         System.out.println("Exception: " + ex);
         ex.printStackTrace();
      }
      return aq_sess;
   }
```

パッケージ oracle.AQ の概要

表 2-1　パッケージ oracle.AQ メンバーの概要

メンバー	説明
クラス、共通	-
AQConstants	AQ 操作で使用される定数
AQAgent	AQ エージェント
AQDriverManager	様々な AQ ドライバに対するドライバ・マネージャ
AQEnqueueOption	AQ エンキュー・オプション
AQDequeueOption	AQ デキュー・オプション
AQMessageProperty	AQ メッセージ・プロパティ
AQQueueProperty	AQ キュー・プロパティ
AQQueueTableProperty	AQ キュー表プロパティ
クラス、Oracle8*i*（このマニュアルでは説明しません）	-
AQOracleSession	Oracle サーバーによる AQSession の実装
AQOracleMessage	Oracle サーバーによる AQMessage の実装
AQOracleDriver	Oracle サーバーによる AQDriver の実装
AQOracleQueue	Oracle サーバーによる AQQueue の実装
AQOracleQueueTable	Oracle サーバーによる AQQueueTable の実装
AQOracleRawPayload	Oracle サーバーによる AQRawPayload の実装
AQOracleObjectPayload	Oracle サーバーによる AQObjectPayload の実装
例外	-
AQException	Java AQ API 使用時にエラーが検出された場合に発生
AQOracleSQLException	SQL 実行時に発生するすべてのエラーに対して発生

AQDriverManager

ドライバ・マネージャ・インタフェース AQDriverManager を介して、Java AQ API の様々な実装が管理されます。Oracle Lite および Oracle9*i* には、AQDriverManager に登録されている AQDriver があります。ドライバ・マネージャは、メッセージ交換タスクの実行に使用できる AQSession の作成に使用します。

AQDriverManager.createAQSession() メソッドをコールすると、createAQSession() コールに渡したパラメータに従って、（登録してあるドライバの中から）適切な AQDriver がコールされます。

Oracle9*i* の AQDriver では、AQSession を作成するパラメータとして、有効な JDBC 接続が渡される必要があります。AQ Java インタフェースを使用するには、DBMS_AQIN パッケージの実行権限が必要です。この権限は AQ_USER_ROLE または AQ_ADMINSTRATOR_ROLE を使用して取得することもできます。また、Oracle9*i* 形式のキュー表に対する適切なシステム権限とキュー権限も必要です。

メソッド

getDrivers

```
public static java.util.Vector getDrivers()
```

このメソッドは、ドライバ・マネージャに登録されているドライバのリストを戻します。登録されているドライバの名前を含んだ文字列のベクトルを戻します。

getAQSession

```
public static AQSession getAQSession (java.lang.Object conn)
   throws AQException
```

このメソッドは、AQSession を作成します。

パラメータ

conn

ユーザーが AQOracleDriver を使用している場合は、オブジェクトとして有効な JDBC 接続を渡す必要があります。

マルチスレッド・プログラム・サポート

現在、Java AQ オブジェクトはスレッド・セーフではありません。したがって、AQSession、AQQueueTable、AQQueue および他の AQ オブジェクトのメソッドを、異なるスレッドから同時にコールしないでください。これらのオブジェクトをスレッド間で渡す

ことはできますが、これらの AQ オブジェクトのメソッドは、プログラムで同時にコールしないでください。

マルチスレッド・プログラムで、各スレッドに（同じまたは異なる JDBC 接続を使用して）異なる AQSession を作成し、AQSession の getQueueTable メソッドと getQueue メソッドを使用して、新しいキュー表とキュー・ハンドルを取得することをお薦めします。

Java AQ ドライバのロード

AQSession を作成するには、最初に JDBC 接続をオープンする必要があります。次に、アプリケーションで使用する必要がある AQDriver をロードします。Oracle9*i* を使用している場合、このドライバは Class.forName("oracle.AQ.AQOracleDriver") コマンドを使用してロードされます。

ドライバのロードが必要となるのは、（最初の createAQSession コール前）1 回のみであることに注意してください。ドライバを複数回ロードしても効果はありません。詳細は、2-3 ページの「oracle.AQ の設定例」を参照してください。

例

```
Connection db_conn;       /* JDBC connection */
AQSession  aq_sess;       /* AQSession */

/* JDBC setup and connection creation: */
Class.forName("oracle.jdbc.driver.OracleDriver");
db_conn = DriverManager.getConnection (
   "jdbc:oracle:oci8:@", "aquser", "aquser");
db_conn.setAutoCommit(false);

/* Load the Oracle9i AQ driver: */
Class.forName("oracle.AQ.AQOracleDriver");
/* Create an AQ Session: */
aq_sess = AQDriverManager.createAQSession(db_conn);
```

通常は、両方の実装に共通のインタフェースとクラスのみ使用してください。この結果、Oracle9*i* による AQ の実装と Oracle Lite による AQ の実装間で、アプリケーションを移植できることが保証されます。

また、oracle.AQ クラスは、共通のインタフェースで使用不能なメソッドが必要な場合にのみ使用してください。AQQueue インタフェースは AQQueueAdmin を拡張したインタフェースであるため、キューの管理と操作のすべての機能性は AQQueue を通して使用できます。

AQSession

メソッド

createQueueTable

```
public AQQueueTable createQueueTable(java.lang.String owner,
                                     java.lang.String name,
                     AQQueueTableProperty property) throws AQException
```

このメソッドは、渡された AQQueueTableProperty オブジェクトに指定されているプロパティに従って、特定のユーザーのスキーマに新しいキュー表を作成します。

パラメータ	説明
owner	キュー表を作成するスキーマ（ユーザー）
name	キュー表の名前
property	キュー表のプロパティ

戻り値

AQQueueTable オブジェクト

getQueueTable

```
public AQQueueTable getQueueTable(java.lang.String owner,
                                  java.lang.String name)
```

このメソッドは、既存のキュー表に対するハンドルを取得するために使用します。

パラメータ	説明
owner	キュー表が常駐しているスキーマ（ユーザー）
name	キュー表の名前

戻り値

AQQueueTable オブジェクト

createQueue

```
public AQQueue createQueue(AQQueueTable q_table,
                           java.lang.String q_name,
                     AQQueueProperty q_property) throws AQException
```

このメソッドは、指定したキュー・プロパティで queue_table にキューを作成します。キュー表の作成に使用された同じスキーマ名を使用します。

パラメータ	説明
q_table	キューを作成するキュー表
q_name	作成するキューの名前
q_property	キューのプロパティ

戻り値

AQQueue オブジェクト

getQueue

```
public AQQueue getQueue(java.lang.String owner,
                        java.lang.String name)
```

このメソッドは、既存のキューに対するハンドルを取得するために使用します。

パラメータ	説明
owner	キュー表が常駐しているスキーマ（ユーザー）
name	キューの名前

戻り値

AQQueue オブジェクト

getDBConnection

```
public java.sql.Connection getDBConnection()
```
このメソッドは、AQ セッション・オブジェクトから実際の JDBC 接続を取得するために使用します。

このメソッドは、Oracle サーバーによる AQSession の実装でのみ使用可能です。したがって、このメソッドをコールする前に、AQSession オブジェクトを AQOracleSession にキャストする必要があります。

例

```
AQSession aq_sess;
Connection db_conn =((AQOracleSession)aq_sess).getDBConnection();
```

listen

```
public AQAgent listen(AQAgent[] agent_list,
                      int wait_time)
```
このメソッドは、メッセージ用の複数のキューをリスニングするために使用します。

パラメータ	説明
agent_list	リスニング対象のエージェントのリスト * コンシューマ・キューが 1 つの場合、AQAgent の名前フィールドは null に設定し、アドレス・フィールドには、[schema].[queue_name] を含める必要があります。 * コンシューマ・キューが複数の場合、AQAgent の名前フィールドにはコンシューマ名を含め、アドレス・フィールドには、[schema].[queue_name] を含める必要があります。
wait_time	リスニング・コールのタイムアウト（秒）。

戻り値

消費可能なメッセージを持つエージェント

例外

タイムアウト（ORA-25254）または別のエラーのためにリスニングが失敗した場合の AQException

例

1. キュー表とキューの作成

AQDriverManager メイン・クラスからコールした runTest クラスを使用して、aqjava ユーザー用のキュー表とキューを作成します。

```
public static void runTest(AQSession aq_sess) throws AQException
{
    AQQueueTableProperty     qtable_prop;
    AQQueueProperty          queue_prop;
    AQQueueTable             q_table;
    AQQueue                  queue;

    /* Create a AQQueueTableProperty object (payload type - RAW): */
    qtable_prop = new AQQueueTableProperty("RAW");

    /* Create a queue table called aq_table1 in aqjava schema: */
    q_table = aq_sess.createQueueTable ("aqjava", "aq_table1",
        qtable_prop);
    System.out.println("Successfully created aq_table1 in aqjava
        schema");

   /* Create a new AQQueueProperty object: */
   queue_prop = new AQQueueProperty();

   /* Create a queue called aq_queue1 in aq_table1: */
   queue = aq_sess.createQueue (q_table, "aq_queue1", queue_prop);
  System.out.println("Successfully created aq_queue1 in aq_table1");
}
```

2. 既存のキュー表とキューに対するハンドルの取得

```
public static void runTest(AQSession aq_sess) throws AQException
{
    AQQueueTable            q_table;
    AQQueue                 queue;

    /* Get a handle to queue table - aq_table1 in aqjava schema: */
    q_table = aq_sess.getQueueTable ("aqjava", "aq_table1");
    System.out.println("Successful getQueueTable");

    /* Get a handle to a queue - aq_queue1 in aqjava schema: */
    queue = aq_sess.getQueue ("aqjava", "aq_queue1");
    System.out.println("Successful getQueue");
}
```

AQConstants

このクラスには、Java AQ API で使用される定数が含まれています。

可視性定数

```
VISIBILITY_IMMEDIATE
public static final int VISIBILITY_IMMEDIATE

VISIBILITY_ONCOMMIT
public static final int VISIBILITY_ONCOMMIT
```

ペイロード型、オブジェクト

```
OBJECT_TYPE_PAYLOAD
public static final int OBJECT_TYPE_PAYLOAD
```

ペイロード型、RAW

```
RAW_TYPE_PAYLOAD
public static final int RAW_TYPE_PAYLOAD
```

AQAgent

このオブジェクトは、メッセージのプロデューサまたはコンシューマを指定します。

コンストラクタ

```
public AQAgent(java.lang.String name,
               java.lang.String address,
               double protocol)

public AQAgent(java.lang.String name,
               java.lang.String address)
```

コンストラクタの実装方法は 2 通りあり、それぞれ指定したパラメータで新しい AQAgent を割り当てます。

パラメータ	説明
name	エージェント名
address	エージェント・アドレス
protocol	エージェントのプロトコル（最初のコンストラクタにのみ必要）。デフォルトは 0（ゼロ）です。

メソッド

getName

```
public java.lang.String getName() throws AQException
```

このメソッドは、エージェント名を取得します。

setName

```
public void setName(java.lang.String name) throws AQException
```

このメソッドは、エージェント名を設定します。

パラメータ	説明
name	エージェント名

getAddress

```
public java.lang.String getAddress() throws AQException
```
このメソッドは、エージェント・アドレスを取得します。

setAddress

```
public void setAddress(java.lang.String address) throws AQException
```
このメソッドは、エージェント・アドレスを設定します。

パラメータ	説明
address	特定の宛先にあるキュー

getProtocol

```
public int getProtocol() throws AQException
```
このメソッドは、エージェントのプロトコルを取得します。

setProtocol

```
public void setProtocol(int protocol) throws AQException
```
このメソッドは、エージェントのプロトコルを設定します。

パラメータ	説明
protocol	エージェントのプロトコル

AQQueueTableProperty

このクラスはキュー表のプロパティを示します。

定数

```
public static final int NONE
public static final int TRANSACTIONAL
```

コンストラクタ

```
public AQQueueTableProperty(java.lang.String p_type)
```

このメソッドは、デフォルトのプロパティ値と指定したペイロード型で AQQueueTableProperty オブジェクトを作成します。

パラメータ	説明
p_type	ペイロード型 : この値は、ロー・ペイロードが含まれるキュー表の場合は "RAW"、構造化ペイロードが含まれるキュー表の場合はオブジェクトのユーザー定義型名です。

メソッド

getPayloadType

```
public java.lang.String getPayloadType() throws AQException
```

このメソッドは、ロー・ペイロードの場合は "RAW"、オブジェクト・ペイロードの場合はオブジェクト型名を戻します。

setPayloadType

```
public void setPayloadType(java.lang.String p_type) throws AQException
```

このメソッドは、ペイロード型の設定に使用します。

パラメータ	説明
p_type	ペイロード型 : この値は、ロー・ペイロードが含まれるキュー表の場合は "RAW"、構造化ペイロードが含まれるキュー表の場合はオブジェクトのユーザー定義型名です。

setStorageClause

```
public void setStorageClause(java.lang.String s_clause) throws AQException
```
このメソッドは、キュー表の作成に使用される記憶域句の設定に使用します。

パラメータ	説明
s_clauses	記憶領域パラメータ : この句は、キュー表の作成時に CREATE TABLE 文で使用されます。

getSortOrder

```
public java.lang.String getSortOrder() throws AQException
```
このメソッドは、使用されるソート順を取得します。

戻り値
使用されるソート順

setSortOrder

```
public void setSortOrder(java.lang.String s_order) throws AQException
```
このメソッドは、使用されるソート順を設定します。

パラメータ	説明
s_order	昇順の sort_key に使用される列を指定します。文字列のフォーマットは、<sort_column1, sort_column2> です。許可される列名は、priority と enq_time です。

isMulticonsumerEnabled

```
public boolean isMulticonsumerEnabled() throws AQException
```
このメソッドは、1 つのメッセージに対して複数のコンシューマを、表に作成されたキューに設定できるかどうかを問い合せます。

戻り値
表に作成したキューに、1 つのメッセージに対して複数のコンシューマを設定できる場合は true

表に作成したキューに、1 つのメッセージに対して複数のコンシューマを設定できない場合は false

setMultiConsumer

```
public void setMultiConsumer(boolean enable) throws AQException
```
このメソッドは、表に作成されたキューに、1 つのメッセージに対して複数のコンシューマを設定できるかどうかを設定します。

パラメータ	説明
enable	表に作成したキューに、1 つのメッセージに対して複数のコンシューマを設定できない場合は false 表に作成したキューに、1 つのメッセージに対して複数のコンシューマを設定できる場合は true

getMessageGrouping

```
public int getMessageGrouping() throws AQException
```
このメソッドは、このキュー表内のキューのメッセージ・グループ化動作プロパティを取得するために使用します。

戻り値

NONE: 各メッセージは個々に処理されます。

TRANSACTIONAL: あるトランザクションの一部としてエンキューされた複数のメッセージは、すべて同一グループの一員とみなされ、関連するメッセージのグループとしてデキューできます。

setMessageGrouping

```
public void setMessageGrouping(int m_grouping) throws AQException
```
このメソッドは、このキュー表に作成されたキューのメッセージ・グループ化動作プロパティを設定するために使用します。

パラメータ	説明
m_grouping	NONE または TRANSACTIONAL

getComment

```
public java.lang.String getComment() throws AQException
```
このメソッドは、キュー表のコメントを取得します。

setComment

```
public void setComment(java.lang.String qt_comment) throws AQException
```
このメソッドは、コメントを設定します。

パラメータ	説明
qt_comment	コメント

getCompatible

```
public java.lang.String getCompatible() throws AQException
```
このメソッドは、Compatible プロパティを取得します。

setCompatible

```
public void setCompatible(java.lang.String qt_compatible)
   throws AQException
```
このメソッドは、Compatible プロパティを設定します。

パラメータ	説明
qt_compatible	Compatible プロパティ

getPrimaryInstance

```
public int getPrimaryInstance() throws AQException
```
このメソッドは、プライマリ・インスタンスを取得します。

setPrimaryInstance

```
public void setPrimaryInstance(int inst) throws AQException
```
このメソッドは、プライマリ・インスタンスを設定します。

パラメータ	説明
inst	プライマリ・インスタンス

getSecondaryInstance

```
public int getSecondaryInstance() throws AQException
```
このメソッドは、セカンダリ・インスタンスを取得します。

setSecondaryInstance

```
public void setSecondaryInstance(int inst) throws AQException
```
このメソッドは、セカンダリ・インスタンスを設定します。

パラメータ	説明
inst	セカンダリ・インスタンス

例

この例を実行するには、最初に、2-3 ページの「oracle.AQ の設定例」の説明に従って test_aqjava クラスを設定します。

1. ロー・ペイロード型で、キュー表のプロパティ・オブジェクトを作成します。

```
public static void runTest(AQSession aq_sess) throws AQException
 {
     AQQueueTableProperty qtable_prop;

     /* Create AQQueueTable Property object: */
     qtable_prop = new AQQueueTableProperty("RAW");
     qtable_prop.setSortOrder("PRIORITY");
}
```

2. ロー・ペイロード型で、キュー表のプロパティ・オブジェクトを作成します（8.1 形式のキューの場合）。

```
 public static void runTest(AQSession aq_sess) throws AQException
 {
     AQQueueTableProperty qtable_prop;

     /* Create AQQueueTable Property object: */
     qtable_prop = new AQQueueTableProperty("RAW");
     qtable_prop.setComment("Qtable with raw payload");
     qtable_prop.setCompatible("8.1");
}
```

3. PERSON ペイロード型（ユーザー定義型）で、キュー表のプロパティ・オブジェクトを作成します。

```
 public static void runTest(AQSession aq_sess) throws AQException
{
     AQQueueTableProperty qtable_prop;
     qtable_prop = new AQQueueTableProperty("PERSON");
     qtable_prop.setComment("Qtable with Person ADT payload");
     qtable_prop.setMessageGrouping(TRANSACTIONAL);
}
```

AQQueueProperty

このクラスは、キューのプロパティを示します。

定数

```
public static final int NORMAL_QUEUE
public static final int EXCEPTION_QUEUE
public static final int INFINITE   /* infinite retention */
```

コンストラクタ

```
public AQQueueProperty()
```

このメソッドは、デフォルトのプロパティ値で新しい AQQueueProperty オブジェクトを作成します。

メソッド

getQueueType

```
public int getQueueType() throws AQException
```

このメソッドは、キュー・タイプを取得します。

戻り値

NORMAL_QUEUE または EXCEPTION_QUEUE

setQueueType

```
public void setQueueType(int q_type) throws AQException
```

このメソッドは、キュー・タイプの設定に使用します。

パラメータ	説明
q_type	NORMAL_QUEUE または EXCEPTION_QUEUE

getMaxRetries

```
public int getMaxRetries() throws AQException
```

このメソッドは、REMOVE モードでのデキューの最大再試行回数を取得します。

setMaxRetries

```
public void setMaxRetries(int retries) throws AQException
public void setMaxRetries(Integer retries) throws AQException
```
このメソッドは、REMOVE モードでのデキューの最大再試行回数を設定します。

パラメータ	説明
retries	REMOVE モードでの最大再試行回数。null を指定すると、デフォルトが使用されます。デフォルトは、シングル・コンシューマ・キューと 8.1 互換のマルチ・コンシューマ・キューに適用されます。Max_retries は、8.0 互換のマルチ・コンシューマ・キューではサポートされていません。

setRetryInterval

```
public void setRetryInterval(double interval) throws AQException
public void setRetryInterval(Double interval) throws AQException
```
このメソッドは、再試行間隔を設定します。これは、アプリケーションのロールバック後、このメッセージの処理がスケジュールされるまでの時間です。デフォルトは 0（ゼロ）です。

パラメータ	説明
interval	再試行間隔。null を指定すると、デフォルトが使用されます。

getRetryInterval

```
public double getRetryInterval() throws AQException
```
このメソッドは、再試行間隔を取得します。

getRetentionTime

```
public double getRetentionTime() throws AQException
```
このメソッドは、保存時間を取得します。

setRetentionTime

```
public void setRetentionTime(double r_time) throws AQException
public void setRetentionTime(Double r_time) throws AQException
```
このメソッドは、保存時間を設定します。

パラメータ	
r_time	保存時間。null を指定すると、デフォルトが使用されます。

getComment

```
public java.lang.String getComment() throws AQException
```
このメソッドは、キューのコメントを取得します。

setComment

```
public void setComment(java.lang.String qt_comment) throws AQException
```
このメソッドは、キューのコメントを設定します。

パラメータ	説明
qt_comment	キューのコメント

例

この例を使用するには、最初に、**2-3 ページの「oracle.AQ の設定例」**の説明に従って test_aqjava クラスを設定します。

AQQueueProperty オブジェクトの作成

```
 {
     AQQueueProperty         q_prop;
     q_prop = new AQQueueProperty();
     q_prop.setRetentionTime(15); /* set retention time */
     q_prop.setRetryInterval(30); /* set retry interval */
}
```

AQQueueTable

AQQueueTable インタフェースには、キュー表の管理に関するメソッドが含まれています。

メソッド

getOwner

```
public java.lang.String getOwner() throws AQException
```
このメソッドは、キュー表の所有者を取得します。

getName

```
public java.lang.String getName() throws AQException
```
このメソッドは、キュー表の名前を取得します。

getProperty

```
public AQQueueTableProperty getProperty() throws AQException
```
このメソッドは、キュー表のプロパティを取得します。

戻り値

AQQueueTableProperty オブジェクト

drop

```
public void drop(boolean force) throws AQException
```
このメソッドは、現行のキュー表を削除します。

パラメータ	説明
force	false: キュー表にキューが存在している場合、この操作は失敗します（デフォルト）。 true ：キュー表内のすべてのキューが自動的に停止および削除されます。

alter

```
public void alter(java.lang.String comment,
                  int primary_instance,
                  int secondary_instance) throws AQException
public void alter(java.lang.String comment) throws AQException
```

このメソッドは、キュー表のプロパティを変更するために使用します。

パラメータ	説明
comment	新しいコメント
primary_instance	プライマリ・インスタンスの新しい値
secondary_instance	セカンダリ・インスタンスの新しい値

createQueue

```
public AQQueue createQueue(java.lang.String queue_name,
                           AQQueueProperty q_property) throws AQException
```

このメソッドは、このキュー表にキューを作成するために使用します。

パラメータ	説明
queue_name	作成するキューの名前
q_property	キューのプロパティ

戻り値

AQQueue オブジェクト

dropQueue

```
public void dropQueue(java.lang.String queue_name) throws AQException
```

このメソッドは、このキュー表のキューを削除するために使用します。

パラメータ	説明
queue_name	削除するキューの名前

例

この例を実行するには、最初に、2-3 ページの「oracle.AQ の設定例」の説明に従って test_aqjava クラスを設定します。

1. キュー表とキューの作成

```
public static void runTest(AQSession aq_sess) throws AQException
{
   AQQueueTableProperty    qtable_prop;
   AQQueueProperty         queue_prop;
   AQQueueTable            q_table;
   AQQueue                 queue;

   /* Create a AQQueueTable property object (payload type - RAW): */
   qtable_prop = new AQQueueTableProperty("RAW");

   /* Create a queue table called aq_table2 in aquser schema: */
   qtable = aq_sess.createQueueTable ("aquser", "aq_table2", qtable_prop);
   System.out.println("Successfully createQueueTable");

   /* Create a new AQQueueProperty object: */
   queue_prop = new AQQueueProperty();

   /* Create a queue called aq_queue2 in aq_table2: */
   queue = qtable.createQueue ("aq_queue2", queue_prop);
   System.out.println("Successful createQueue");
}
```

2. キュー表の変更、プロパティの取得、およびキュー表の削除

```
{
     AQQueueTableProperty    qtable_prop;
     AQQueueTable            q_table;

     /*Get a handle to the queue table called aq_table2 in aquser schema: */
     q_table = aq_sess.getQueueTable ("aqjava", "aq_table2");
     System.out.println("Successful getQueueTable");
     /* Get queue table properties: */
     qtable_prop = q_table.getProperty();

     /* Alter the queue table: */
     q_table.alter("altered queue table");

     /* Drop the queue table (and automatically drop queues inside it): */
     q_table.drop(true);
     System.out.println("Successful drop");
}
```

注意： キューは、AQSession.createQueue または AQQueueTable.createQueue インタフェースを使用して作成できます。前者には、パラメータとして queue_name とキューのプロパティの他に、AQQueueTable オブジェクトが必要です。

AQQueueAdmin

メソッド

start

```
public void start(boolean enqueue,
                  boolean dequeue) throws AQException
```
このメソッドは、このキューにおけるエンキューとデキューを使用可能にするために使用します。

パラメータ	説明
enqueue	true: このキューにおけるエンキューを使用可能にします。 false: 現行の設定のままにします。
dequeue	true: このキューにおけるデキューを使用可能にします。 false: 現行の設定のままにします。

startEnqueue

```
public void startEnqueue() throws AQException
```
このメソッドは、このキューにおけるエンキューを使用可能にするために使用します。これは、start(true, false) と同じです。

startDequeue

```
public void startEnqueue() throws AQException
```
このメソッドは、このキューにおけるデキューを使用可能にするために使用します。これは、start(false, true) と同じです。

stop

```
public void stop(boolean enqueue,
                 boolean dequeue,
                 boolean wait) throws AQException
```
このメソッドは、このキューにおけるエンキューとデキューを使用禁止にするために使用します。

パラメータ	説明
enqueue	true: このキューにおけるエンキューを使用禁止にします。 false: 現行の設定のままにします。
dequeue	true: このキューにおけるデキューを使用禁止にします。 false: 現行の設定のままにします。
wait	true: 未完了のトランザクションが完了するまで待機します。 false: 成功かエラーかを即時に戻します。

stopEnqueue

```
public void stopEnqueue(boolean wait) throws AQException
```
このメソッドは、キューにおけるエンキューを使用禁止にするために使用します。これは、stop(true, false, wait) と同じです。

パラメータ	説明
wait	true: 未完了のトランザクションが完了するまで待機します。 false: 成功かエラーかを即時に戻します。

stopDequeue

```
public void stopDequeue(boolean wait) throws AQException
```
このメソッドは、キューにおけるデキューを使用禁止にするために使用します。これは、stop(false, true, wait) と同じです。

パラメータ	説明
wait	true: 未完了のトランザクションが完了するまで待機します。 false: 成功かエラーかを即時に戻します。

drop

```
public void drop() throws AQException
```
このメソッドは、キューの削除に使用します。

alterQueue

```
public void alterQueue(AQQueueProperty property) throws AQException
```
このメソッドは、キューのプロパティを変更するために使用します。

パラメータ	説明
property	新しいプロパティ値を設定した AQQueueProperty オブジェクト。変更できるプロパティは、max_retries、retry_delay、retention_time および comment のみです。

addSubscriber

```
public void addSubscriber(AQAgent subscriber,
                          java.lang.String rule) throws AQException
```
このメソッドは、このキューにサブスクライバを追加するために使用します。

パラメータ	説明
subscriber	サブスクリプションが定義される AQAgent
rule	メッセージのプロパティおよびメッセージ・データのプロパティに基づいた条件式

removeSubscriber

```
public void removeSubscriber(AQAgent subscriber) throws AQException
```
このメソッドは、キューからサブスクライバを削除します。

パラメータ	説明
subscriber	削除する AQAgent

alterSubscriber

```
public void alterSubscriber(AQAgent subscriber,
                            java.lang.String rule) throws AQException
```

このメソッドは、キューに対するサブスクライバのプロパティを変更します。

パラメータ	説明
subscriber	サブスクリプションが変更される AQAgent
rule	メッセージのプロパティおよびメッセージ・データのプロパティに基づいた条件式

grantQueuePrivilege

```
public void grantQueuePrivilege(java.lang.String privilege,
                                java.lang.String grantee,
                              boolean grant_option) throws AQException
public void grantQueuePrivilege(java.lang.String privilege,
                            java.lang.String grantee) throws AQException
```

このメソッドは、ユーザーおよびロールにキュー権限を付与するために使用します。このメソッドはオーバーロードされています。2 番目の実装は、grant_option = false を指定して 1 番目の実装をコールした場合と同じです。

パラメータ	説明
privilege	付与する権限を指定 :ENQUEUE、DEQUEUE または ALL。
grantee	権限受領者を指定します。ユーザー、ロールまたは PUBLIC ロールを指定できます。
grant_option	true: 権限受領者は、このメソッドを使用して第三者にアクセス権限を付与することを許可されます。 false: デフォルト。

revokeQueuePrivilege

```
public void revokeQueuePrivilege(java.lang.String privilege,
                                 java.lang.String grantee) throws AQException
```

このメソッドは、キュー権限の取消しに使用します。

パラメータ	説明
privilege	取り消す権限を指定 :ENQUEUE、DEQUEUE または ALL。
grantee	権限受領者を指定します。ユーザー、ロールまたは PUBLIC ロールを指定できます。

schedulePropagation

```
public void schedulePropagation(java.lang.String destination,
                                java.util.Date start_time,
                                java.lang.Double duration,
                                java.lang.String next_time,
                               java.lang.Double latency) throws AQException
```

このメソッドは、キューから、データベース・リンクで識別される宛先への伝播をスケジュールするために使用します。

パラメータ	説明
destination	宛先のデータベース・リンクを指定します。ソース・キュー内にある宛先レシピエントに対するメッセージが伝播されます。null => 宛先はローカル・データベースで、メッセージはそのローカル・データベース内の他のすべてのキューに伝播されます。このフィールドの最大長は 128 バイトです。名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。
start_time	メッセージをこのキューから宛先に伝播する伝播枠の初期開始時間を指定します。null=> 開始時間は現在時刻です。
duration	伝播枠の継続期間を秒で指定します。null => 伝播枠は永続的か、または伝播スケジュールが取り消されるまで存続します。
next_time	現行の伝播枠の終了から次回の伝播枠の開始を計算する日付関数（例 : "SYSDATE+ 1 - duration/86400" を使用すると、伝播枠は毎日同時刻に開始されます)。null => 伝播は現行枠の終了時に停止します。
latency	エンキュー後、その伝播枠内でメッセージが伝播されるまでの最大待機時間（秒)。null => デフォルト値（60 秒）が使用されます。

unschedulePropagation

```
public void unschedulePropagation(java.lang.String destination)
   throws AQException
```

このメソッドは、現行のキューから、特定のデータベース・リンクが示す宛先へのメッセージ伝播のスケジュールを取り消すために使用します。

パラメータ	説明
destination	宛先のデータベース・リンクを指定します。null => 宛先はローカル・データベースです。

alterPropagationSchedule

```
public void alterPropagationSchedule(java.lang.String destination,
                                     java.Double duration,
                                     java.String next_time,
                          java.Double latency) throws AQException
```

このメソッドは、伝播スケジュールの変更に使用します。

パラメータ	説明
destination	宛先のデータベース・リンクを指定します。null => 宛先はローカル・データベースです。
duration	伝播枠の継続期間を秒で指定します。null => 伝播枠は永続的か、または伝播スケジュールが取り消されるまで存続します。
next_time	現行の伝播枠の終了から次回の伝播枠の開始を計算する日付関数（例："SYSDATE+ 1 - duration/86400" を使用すると、伝播枠は毎日同時刻に開始されます）。null => 伝播は現行枠の終了時に停止します。
latency	エンキュー後、その伝播枠内でメッセージが伝播されるまでの最大待機時間（秒）。null => デフォルト値（60 秒）が使用されます。

enablePropagationSchedule

```
public void enablePropagationSchedule(java.lang.String destination)
   throws AQException
```

このメソッドは、伝播スケジュールを使用可能にするために使用します。

パラメータ	説明
destination	宛先のデータベース・リンクを指定します。null => 宛先はローカル・データベースです。

disablePropagationSchedule

```
public void disablePropagationSchedule(java.lang.String destination)
   throws AQException
```

このメソッドは、伝播スケジュールを使用禁止にするために使用します。

パラメータ	説明
destination	宛先のデータベース・リンクを指定します。null => 宛先はローカル・データベースです。

例

test_aqjava クラスを設定します。詳細は、**2-3 ページの「oracle.AQ の設定例」**を参照してください。

1. キューの作成とエンキュー / デキューの開始

```
{
     AQQueueTableProperty    qtable_prop;
     AQQueueProperty         queue_prop;
     AQQueueTable            q_table;
     AQQueue                 queue;

     /* Create a AQQueueTable property object (payload type - RAW): */
     qtable_prop = new AQQueueTableProperty("RAW");
    qtable_prop.setCompatible("8.1");

     /* Create a queue table called aq_table3 in aqjava schema: */
     q_table = aq_sess.createQueueTable ("aqjava","aq_table3", qtable_prop);
     System.out.println("Successful createQueueTable");

     /* Create a new AQQueueProperty object: */
     queue_prop = new AQQueueProperty();
```

```
        /* Create a queue called aq_queue3 in aq_table3: */
        queue = aq_sess.createQueue (q_table, "aq_queue3", queue_prop);
        System.out.println("Successful createQueue");

        /* Enable enqueue/dequeue on this queue: */
        queue.start();
        System.out.println("Successful start queue");

        /* Grant enqueue_any privilege on this queue to user scott: */
        queue.grantQueuePrivilege("ENQUEUE", "scott");
        System.out.println("Successful grantQueuePrivilege");
    }
```

2. マルチ・コンシューマ・キューの作成とサブスクライバの追加

```
public static void runTest(AQSession aq_sess) throws AQException
{
    AQQueueTableProperty    qtable_prop;
    AQQueueProperty         queue_prop;
    AQQueueTable            q_table;
    AQQueue                 queue;
    AQAgent                 subs1, subs2;

    /* Create a AQQueueTable property object (payload type - RAW): */
    qtable_prop = new AQQueueTableProperty("RAW");
    System.out.println("Successful setCompatible");

    /* Set multiconsumer flag to true: */
    qtable_prop.setMultiConsumer(true);

    /* Create a queue table called aq_table4 in aqjava schema: */
    q_table = aq_sess.createQueueTable ("aqjava","aq_table4", qtable_prop);
    System.out.println("Successful createQueueTable");

    /* Create a new AQQueueProperty object: */
    queue_prop = new AQQueueProperty();

    /* Create a queue called aq_queue4 in aq_table4 */
    queue = aq_sess.createQueue (q_table, "aq_queue4", queue_prop);
    System.out.println("Successful createQueue");

    /* Enable enqueue/dequeue on this queue: */
    queue.start();
    System.out.println("Successful start queue");
```

```
        /* Add subscribers to this queue: */
        subs1 = new AQAgent("GREEN", null, 0);
        subs2 = new AQAgent("BLUE", null, 0);

        queue.addSubscriber(subs1, null); /* no rule   */
        System.out.println("Successful addSubscriber 1");

        queue.addSubscriber(subs2, "priority < 2"); /* with rule */
        System.out.println("Successful addSubscriber 2");
    }
```

AQQueue

このインタフェースは、キューの操作インタフェースをサポートします。AQQueue は AQQueueAdmin を拡張したインタフェースです。このため、このインタフェースを通して管理機能も使用できます。

メソッド

getOwner

```
public java.lang.String getOwner() throws AQException
```
このメソッドは、キューの所有者を取得します。

getName

```
public java.lang.String getName() throws AQException
```
このメソッドは、キューの名前を取得します。

getQueueTableName

```
public java.lang.String getQueueTableName() throws AQException
```
このメソッドは、キューが常駐しているキュー表の名前を取得します。

getProperty

```
public AQQueueProperty getProperty() throws AQException
```
このメソッドは、キューのプロパティを取得するために使用します。

戻り値

AQQueueProperty オブジェクト

createMessage

```
public AQMessage createMessage() throws AQException
```
このメソッドは、エンキューするデータを移入できる新しい AQMessage オブジェクトを作成するために使用します。

戻り値

AQMessage オブジェクト

enqueue

```
public byte[] enqueue(AQEnqueueOption enq_option,
                      AQMessage message) throws AQException
```
このメソッドは、キューにメッセージをエンキューするために使用します。

パラメータ	説明
enq_option	AQEnqueueOption オブジェクト
message	エンキューする AQMessage

戻り値
エンキューしたメッセージのメッセージ ID。このコールの完了後、AQMessage オブジェクトの messageId フィールドにもメッセージ ID が移入されます。

dequeue

```
public AQMessage dequeue(AQDequeueOption deq_option)
   throws AQException
```
このメソッドは、キューからメッセージをデキューするために使用します。

パラメータ	説明
deq_option	AQDequeueOption オブジェクト

戻り値
AQMessage。デキューされたメッセージ。

dequeue（Oracle オブジェクト型のペイロードを持つキュー用 ー SQL データ・バージョン）

```
public AQMessage dequeue(AQDequeueOption deq_option, java.lang.Class payload_class)
throws AQException
```
このメソッドは、Oracle オブジェクトのペイロードを含んだキューからメッセージをデキューするために使用します。プログラムで、Java クラスを Oracle オブジェクト型にマップするために SQL Data インタフェースを使用している場合は、このバージョンを使用してください。

パラメータ

deq_option - AQDequeueOption オブジェクト
payload_class ― デキューされたペイロードは、この型のオブジェクトとして変換されます。指定されるクラスは、SQLData インタフェースを実装し、キューに定義されているペイロード型に対応している必要があります。

戻り値

AQMessage。デキューされたメッセージ。

ユーザーは、JDBC 接続の typeMap 内のキューに含まれているユーザー定義型にマップされるすべての Java クラスを登録する必要もあります。

SQLData インタフェース、および型マップにクラスを登録する方法は、『Oracle9*i* JDBC 開発者ガイドおよびリファレンス』を参照してください。

dequeue（Oracle オブジェクト型のペイロードを持つキュー用 ― CustomDatum バージョン）

```
public AQMessage dequeue(AQDequeueOption deq_option,
oracle.sql.CustomDatumFactory payload_fact) throws AQException
```

このメソッドは、Oracle オブジェクトのペイロードを含んだキューからメッセージをデキューするために使用します。プログラムで、Java クラスを Oracle オブジェクト型にマップするために CustomDatum インタフェースを使用している場合は、このバージョンを使用してください。

パラメータ

deq_option - AQDequeueOption オブジェクト
payload_fact ― キュー内にあるペイロードの SQL ユーザー定義型にマップされるクラスの CustomDatum ファクトリです。たとえば、Person がデータベース内の PERSON ユーザー定義型にマップされる Java クラスの場合、このクラスの CustomDatum ファクトリは、Person.getFactory() を使用して取得できます。

戻り値

AQMessage - デキューされたメッセージ。
CustomDatum と CustomDatumFactory インタフェースの詳細、および型マップにクラスを登録する方法は、『Oracle9*i* JDBC 開発者ガイドおよびリファレンス』を参照してください。

dequeue（Oracle オブジェクト型のペイロードを持つキュー用 ー ORAData バージョン）

```
public AQMessage dequeue(AQDequeueOption deq_option,
oracle.sql.ORADataFactory payload_fact) throws AQException
```

このメソッドは、Oracle オブジェクトのペイロードを含んだキューからメッセージをデキューするために使用します。プログラムで、Java クラスを Oracle オブジェクト型にマップするために ORAData インタフェースを使用している場合は、このバージョンを使用してください。

パラメータ

`deq_option - AQDequeueOption` オブジェクト
`payload_fact` ー キュー内にあるペイロードの SQL ユーザー定義型にマップされるクラスの ORAData ファクトリです。たとえば、Person がデータベース内の PERSON ユーザー定義型にマップされる Java クラスの場合、このクラスの ORAData ファクトリは、Person.getORADataFactory() を使用して取得できます。

戻り値

`AQMessage` - デキューされたメッセージ
ORAData と ORADataFactory インタフェースの詳細、および型マップにクラスを登録する方法は、『Oracle9*i* JDBC 開発者ガイドおよびリファレンス』を参照してください。

getSubscribers

```
public AQAgent[] getSubscribers() throws AQException
```

このメソッドは、キューのサブスクライバ・リストを取得するために使用します。

戻り値

`AQAgents` の配列

AQEnqueueOption

このクラスは、エンキュー操作で使用可能なオプションを指定するために使用します。

定数

```
public static final int DEVIATION_NONE
public static final int DEVIATION_BEFORE
public static final int DEVIATION_TOP
public static final int VISIBILITY_ONCOMMIT
public static final int VISIBILITY_IMMEDIATE
```

コンストラクタ

```
public AQEnqueueOption(int visibility,
                       byte[] relative_msgid,
                       int sequence_deviation)
public AQEnqueueOption()
```

2 種類のコンストラクタを使用できます。1 つは指定したオプションでオブジェクトを作成し、もう 1 つはデフォルト・オプションでオブジェクトを作成します。

パラメータ	説明
visibility	VISIBILITY_IMMEDIATE または VISIBILITY_ONCOMMIT（デフォルト）。
relative_msgid	DEVIATION_BEFORE が使用されているときは、現行のメッセージがエンキューされた後にエンキューされるメッセージのメッセージ識別子を示します。
sequence_deviation	DEVIATION_TOP: メッセージは、他のいずれのメッセージよりも先にエンキューされます。 DEVIATION_BEFORE: メッセージは、relative_msgid で指定されたメッセージよりも先にエンキューされます。 DEVIATION_NONE: デフォルト。

getVisibility

```
public int getVisibility() throws AQException
```

このメソッドは可視性のプロパティを取得します。

戻り値

VISIBILITY_IMMEDIATE または VISIBILITY_ONCOMMIT

setVisibility

```
public void setVisibility(int visibility) throws AQException
```
このメソッドは、可視性のプロパティを設定します。

パラメータ	説明
visibility	VISIBILITY_IMMEDIATE または VISIBILITY_ONCOMMIT

getRelMessageId

```
public byte[] getRelMessageId() throws AQException
```
このメソッドは、相対メッセージ ID を取得します。

getSequenceDeviation

```
public int getSequenceDeviation() throws AQException
```
このメソッドは、順序逸脱を取得します。

setSequenceDeviation

```
public void setSequenceDeviation(int sequence_deviation,
                                 byte[] relative_msgid) throws AQException
```
このメソッドは、エンキューされるメッセージが、キュー内にすでにある他のメッセージよりも先にデキューされる必要があるかどうかを指定します。

パラメータ	説明
sequence_deviation	DEVIATION_TOP: メッセージは、他のいずれのメッセージよりも先にエンキューされます。 DEVIATION_BEFORE: メッセージは、relative_msgid で指定されたメッセージよりも先にエンキューされます。 DEVIATION_NONE: デフォルト。
relative_msgid	DEVIATION_BEFORE が使用されているときは、現行のメッセージがエンキューされた後にエンキューされるメッセージのメッセージ識別子を示します。

AQDequeueOption

このクラスは、デキュー・オプションに使用可能なオプションを指定するために使用します。

定数

```
public static final int NAVIGATION_FIRST_MESSAGE
public static final int NAVIGATION_NEXT_TRANSACTION
public static final int NAVIGATION_NEXT_MESSAGE
public static final int DEQUEUE_BROWSE
public static final int DEQUEUE_LOCKED
public static final int DEQUEUE_REMOVE
public static final int DEQUEUE_REMOVE_NODATA
public static final int WAIT_FOREVER
public static final int WAIT_NONE
public static final int VISIBILITY_ONCOMMIT
public static final int VISIBILITY_IMMEDIATE
```

コンストラクタ

```
public AQDequeueOption()
```

このメソッドは、デフォルト・オプションでオブジェクトを作成します。

メソッド

getConsumerName

```
public java.lang.String getConsumerName() throws AQException
```

このメソッドは、コンシューマ名を取得します。

setConsumerName

```
public void setConsumerName(java.lang.String consumer_name)
   throws AQException
```

このメソッドは、コンシューマ名を設定します。

パラメータ	説明
consumer_name	エージェント名

getDequeueMode

```
public int getDequeueMode() throws AQException
```
このメソッドは、デキュー・モードを取得します。

戻り値

DEQUEUE_BROWSE、DEQUEUE_LOCKED、DEQUEUE_REMOVE または DEQUEUE_REMOVE_NODATA

setDequeueMode

```
public void setDequeueMode(int dequeue_mode) throws AQException
```
このメソッドは、デキュー・モードを設定します。

パラメータ	説明
dequeue_mode	DEQUEUE_BROWSE、DEQUEUE_LOCKED、DEQUEUE_REMOVE または DEQUEUE_REMOVE_NODATA

getNavigationMode

```
public int getNavigationMode() throws AQException
```
このメソッドは、ナビゲーション・モードを取得します。

戻り値

NAVIGATION_FIRST_MESSAGE、NAVIGATION_NEXT_MESSAGE または NAVIGATION_NEXT_TRANSACTION

setNavigationMode

```
public void setNavigationMode(int navigation) throws AQException
```
このメソッドは、ナビゲーション・モードを設定します。

パラメータ	説明
navigation	NAVIGATION_FIRST_MESSAGE、NAVIGATION_NEXT_MESSAGE または NAVIGATION_NEXT_TRANSACTION

getVisibility

```
public int getVisibility() throws AQException
```
このメソッドは可視性のプロパティを取得します。

戻り値

VISIBILITY_IMMEDIATE または VISIBILITY_ONCOMMIT

setVisibility

```
public void setVisibility(int visibility) throws AQException
```
このメソッドは、可視性のプロパティを設定します。

パラメータ	説明
visibility	VISIBILITY_IMMEDIATE または VISIBILITY_ONCOMMIT

getWaitTime

```
public int getWaitTime() throws AQException
```
このメソッドは、待機時間を取得します。

戻り値

WAIT_FOREVER または WAIT_NONE、あるいは実際の時間（秒）

setWaitTime

```
public void setWaitTime(int wait_time) throws AQException
```
このメソッドは、待機時間を設定します。

パラメータ	説明
wait_time	WAIT_FOREVER または WAIT_NONE、あるいは時間（秒）

getMessageId

```
public byte[] getMessageId() throws AQException
```
このメソッドは、メッセージ ID を取得します。

setMessageId

```
public void setMessageId(byte[] message_id) throws AQException
```
このメソッドは、メッセージ ID を設定します。

パラメータ	説明
message_id	メッセージ ID

getCorrelation

```
public java.lang.String getCorrelation() throws AQException
```
このメソッドは、相関 ID を取得します。

setCorrelation

```
public void setCorrelation(java.lang.String correlation)
   throws AQException
```
このメソッドは、相関 ID を設定します。

パラメータ	説明
correlation	ユーザーが提供した情報

AQMessage

このインタフェースには、ローまたはオブジェクトのペイロードを持つ AQ メッセージ用のメソッドが含まれています。

メソッド

getMessageId

```
public byte[] getMessageId() throws AQException
```
このメソッドは、メッセージ ID を取得します。

getRawPayload

```
public AQRawPayload getRawPayload() throws AQException
```
このメソッドは、ロー・ペイロードを取得します。

戻り値
AQRawPayload オブジェクト

setRawPayload

```
public void setRawPayload(AQRawPayload message_payload)
   throws AQException
```
このメソッドは、ロー・ペイロードを設定します。オブジェクト型のキューから作成されたメッセージでコールされると、AQException が発生します。

パラメータ	説明
message_payload	ロー・ユーザー・データを含んだ AQRawPayload オブジェクト

getObjectPayload

```
public AQObjectPayload getObjectPayload() throws AQException
```
オブジェクトのペイロードを取得します。

戻り値
AQObjectPayload オブジェクト

setObjectPayload

```
public void setObjectPayload(AQObjectPayload message_payload)
   throws AQException
```
オブジェクトのペイロードを設定します。

パラメータ	説明
message_payload	オブジェクト・ユーザー・データを含んだ AQObjectPayload オブジェクト。ロー型のキューから作成されたメッセージでコールされると、AQException が発生します。

getMessageProperty

```
public AQMessageProperty getMessageProperty() throws AQException
```
このメソッドは、メッセージのプロパティを取得します。

戻り値

AQMessageProperty オブジェクト

setMessageProperty

```
public void setMessageProperty(AQMessageProperty property)
   throws AQException
```
このメソッドは、メッセージのプロパティを設定します。

パラメータ	説明
property	AQMessageProperty オブジェクト

AQMessageProperty

AQMessageProperty クラスには、個々のメッセージを管理するために AQ で使用される情報が含まれています。プロパティはエンキュー時に設定され、その値はデキュー時に戻されます。

定数

```
public static final int DELAY_NONE
public static final int EXPIRATION_NEVER
public static final int STATE_READY
public static final int STATE_WAITING
public static final int STATE_PROCESSED
public static final int STATE_EXPIRED
```

コンストラクタ

```
public AQMessageProperty()
```

このメソッドは、デフォルトのプロパティ値で AQMessageProperty オブジェクトを作成します。

メソッド

getPriority

```
public int getPriority() throws AQException
```

このメソッドは、メッセージの優先順位を取得します。

setPriority

```
public void setPriority(int priority) throws AQException
```

このメソッドは、メッセージの優先順位を設定します。

パラメータ	説明
priority	メッセージの優先順位。負数を含め、あらゆる数値を指定できます。数値が小さいほど優先順位が高いことを示します。

getDelay

```
public int getDelay() throws AQException
```

このメソッドは、遅延時間を取得します。

setDelay

```
public void setDelay(int delay) throws AQException
```
このメソッドは、遅延時間を設定します。

パラメータ	説明
delay	メッセージがデキュー可能になるまでの秒数を示します。NO_DELAY を指定すると、メッセージはすぐにデキュー可能になります。

getExpiration

```
public int getExpiration() throws AQException
```
このメソッドは、タイムアウト値を取得します。

setExpiration

```
public void setExpiration(int expiration) throws AQException
```
このメソッドは、タイムアウト値を設定します。

パラメータ	説明
expiration	メッセージをデキューできる継続期間。このパラメータは遅延からのオフセットです。NEVER を指定すると、メッセージはタイムアウトになりません。

getCorrelation

```
public java.lang.String getCorrelation() throws AQException
```
このメソッドは、相関を取得します。

setCorrelation

```
public void setCorrelation(java.lang.String correlation)
   throws AQException
```
このメソッドは、相関を設定します。

パラメータ	説明
correlation	ユーザーが提供した情報

getAttempts

```
public int getAttempts() throws AQException
```
このメソッドは、試行回数を取得します。

getRecipientList

```
public java.util.Vector getRecipientList() throws AQException
```
このメソッドは、レシピエント・リストを取得します。

戻り値

AQAgents のベクトル。このパラメータは、デキュー時にコンシューマには戻されません。

setRecipientList

```
public void setRecipientList(java.util.Vector r_list)
   throws AQException
```
このメソッドは、レシピエント・リストを設定します。

パラメータ	説明
r_list	AQAgents のベクトル。デフォルトのレシピエントはキューのサブスクライバです。

getOrigMessageId

```
public byte[] getOrigMessageId() throws AQException
```
このメソッドは、元のメッセージ ID を取得します。

getSender

```
public AQAgent getSender() throws AQException
```
このメソッドは、メッセージのセンダーを取得します。

setSender

```
public void setSender(AQAgent sender) throws AQException
```
このメソッドは、メッセージのセンダーを設定します。

パラメータ	説明
sender	AQAgent

getExceptionQueue

```
public java.lang.String getExceptionQueue() throws AQException
```
このメソッドは、例外キューの名前を取得します。

setExceptionQueue

```
public void setExceptionQueue(java.lang.String queue)
   throws AQException
```
このメソッドは、例外キューの名前を設定します。

パラメータ	説明
queue	例外キューの名前

getEnqueueTime

```
public java.util.Date getEnqueueTime() throws AQException
```
このメソッドは、エンキュー時間を取得します。

getState

```
public int getState() throws AQException
```
このメソッドは、メッセージの状態を取得します。

戻り値

STATE_READY、STATE_WAITING、STATE_PROCESSED または STATE_EXPIRED

AQRawPayload

このオブジェクトは、AQMessage に組み込まれているロー・ユーザー・データを示します。

メソッド

getStream

```
public int getStream(byte[] value, int len) throws AQException
```
このメソッドは、ロー・ペイロード・データの一部を指定したバイト配列に読み込みます。

パラメータ	説明
value	ロー・データを格納するバイト配列
len	読み込むバイト数

戻り値
読み込まれたバイト数

getBytes

```
public byte[] getBytes() throws AQException
```
このメソッドは、ロー・ペイロード・データ全体をバイト配列として取り出します。

戻り値
バイト ー バイト配列として戻されたロー・ペイロード

setStream

```
public void setStream(byte[] value,
                      int len) throws AQException
```
このメソッドは、ロー・ペイロードの値を設定します。

パラメータ	説明
value	ロー・ペイロードを格納するバイト配列
len	ロー・ストリームに書き込むバイト数

AQObjectPayload

このオブジェクトは、AQMessage に組み込まれている（オブジェクト・キュー用の）構造化されたユーザー・データを示します。

メソッド

setPayloadData

```
public void setPayloadData(java.lang.Object obj) throws AQException
```

このメソッドは、AQObjectPayload オブジェクトにペイロードを挿入するために使用します。

パラメータ	説明
obj	格納するユーザー・データ。使用している AQ ドライバによって、渡すことのできるオブジェクトの型に特定の制限があります。Oracle9*i* の AQ ドライバは、ペイロード内に SQLData、ORAData または CustomDatum インタフェースを実装しているオブジェクトを受け入れます。

SQLData、ORAData および CustomDatum の各インタフェースの詳細は、『Oracle9*i* JDBC 開発者ガイドおよびリファレンス』を参照してください。

getPayloadData

```
public java.lang.Object getPayloadData() throws AQException
```

このメソッドは、AQObjectPayload オブジェクトからメッセージ・ペイロードを取り出すために使用します。

戻り値

メッセージ内のオブジェクト・ペイロード ― デキュー時に指定された SQLData クラス、ORADataFactory または CustomDatumFactory によって異なります。

AQException

```
public class AQException extends java.lang.RuntimeException
```

この例外は、Java AQ API 使用時にエラーが検出された場合に発生します。

このクラスは、Java 例外でサポートされているすべてのメソッドと一部の追加メソッドをサポートします。

メソッド

getMessage

このメソッドはエラー・メッセージを取得します。

getErrorCode

このメソッドは、エラー番号（Oracle エラー・コード）を取得します。

getNextException

このメソッドは、連鎖内の次の例外（ある場合）を取得します。

AQOracleSQLException

AQOracleSQLException は、AQException を拡張するクラスです。

Oracle9*i* の AQ ドライバ使用時に発生するエラーには、クライアント側から発生するエラーと RDBMS から発生するエラーがあります。Oracle9*i* ドライバは、SQL 実行時に発生するすべてのエラーに対して、AQOracleSQLException を呼び出します。

2 種類の例外の違いを区別するには、このクラスが役に立つ場合があります。通常は、AQException のみ使用します。

3

パッケージ oracle.AQ.xml

この章では、Oracle9*i* Advanced Queuing（AQ）XML Servlet のクラスが含まれているパッケージ oracle.AQ.xml について説明します。このサーブレットは、Internet Data Access Presentation（iDAP）を使用して、HTTP などのオープン・プロトコルを通じて Oracle9*i* AQ にアクセスするために使用します。

この章は、次の項で構成されています。

パッケージ oracle.AQ.xml の説明

パッケージ oracle.AQ.xml には、Oracle9*i* Advanced Queuing（AQ）XML Servlet に必要なクラスが含まれています。

Oracle9*i* Advanced Queuing の Java アプリケーションを開発する方法は、『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』に記述されています。

共通のインタフェースとクラスは、現行の PL/SQL インタフェースに基づいています。

- 共通のインタフェースには接頭辞の `AQ` が付いています。
- このマニュアルでは、共通インタフェースとそれに対応する Oracle9*i* による実装（接頭辞は `AQOracle`）について説明します。

AQ XML サーブレットは、HTTP などのオープン・プロトコルおよび Internet Data Access Presentation（iDAP）と呼ばれる XML メッセージ・フォーマットを使用して、Oracle9*i* AQ にアクセスするために使用します。

AQ サーブレットを使用すると、クライアントは次の操作を実行できます。

- シングル・コンシューマ・キューへのメッセージの送信
- マルチ・コンシューマ・キュー / トピックへのメッセージの公開
- キューからのメッセージの受信
- メッセージ通知受信の登録

サーブレットは、JDBC OCI ドライバを使用して Oracle9*i* データベース・サーバーに接続するため、サーブレットのホストとなるマシンには、Oracle9*i* Client ライブラリをインストールする必要があります。LD_LIBRARY_PATH に $ORACLE_HOME/lib を含めます。

oracle.AQ.xml.AQxmlServlet または、oracle.AQ.xml.AQxmlServlet20 を拡張した Java クラスを定義してサーブレットを作成できます。これらのクラスは javax.servlet.http.HttpServlet を拡張しています。

サーブレットは、次のように、Java Servlet 2.0、Java Servlet 2.2 または Java Servlet 2.3 の各インタフェースを実装するすべての Web サーバーまたはサーブレット・コンテナに配置できます。

1. JavaSoft の Servlet 2.0 インタフェースを実装する Web サーバーに AQ サーブレットを配置するには、`oracle.AQ.xml.AQxmlServle20` クラスに拡張するように、クラスを定義します。
2. JavaSoft の Servlet 2.2 インタフェースを実装する Web サーバーに AQ サーブレットを配置するには、`oracle.AQ.xml.AQxmlServlet` クラスに拡張するように、クラスを定義します。

3. サーブレットは、次のように、JDK 1.3、JDK 1.2 または JDK 1.1 ライブラリを使用してコンパイルできます。

   - JDK 1.3 の場合は、CLASSPATH に次のものを含めてください。

     ```
     $ORACLE_HOME/jdbc/lib/classes13.zip
     $ORACLE_HOME/jdbc/lib/jta.zip
     $ORACLE_HOME/jdbc/lib/nls_charset13.zip
     $ORACLE_HOME/jdbc/lib/jndi.zip
     $ORACLE_HOME/lib/lclasses13.zip
     $ORACLE_HOME/lib/xmlparserv2.jar
     $ORACLE_HOME/lib/xschema.jar
     $ORACLE_HOME/rdbms/jlib/aqapi.jar
     $ORACLE_HOME/rdbms/jlib/jmscommon.jar
     $ORACLE_HOME/rdbms/jlib/aqxml.jar
     $ORACLE_HOME/rdbms/jlib/xsu13.jar
     $ORACLE_HOME/jis/lib/servlet.jar
     ```

   - JDK 1.2.x の場合は、CLASSPATH に次のものを含めてください。

     ```
     $ORACLE_HOME/jdbc/lib/classes12.zip
     $ORACLE_HOME/jdbc/lib/jta.zip
     $ORACLE_HOME/jdbc/lib/nls_charset12.zip
     $ORACLE_HOME/jdbc/lib/jndi.zip
     $ORACLE_HOME/lib/lclasses12.zip
     $ORACLE_HOME/lib/xmlparserv2.jar
     $ORACLE_HOME/lib/xschema.jar
     $ORACLE_HOME/rdbms/jlib/aqapi.jar
     $ORACLE_HOME/rdbms/jlib/jmscommon.jar
     $ORACLE_HOME/rdbms/jlib/aqxml.jar
     $ORACLE_HOME/rdbms/jlib/xsu12.jar
     $ORACLE_HOME/jis/lib/servlet.jar
     ```

   - JDK 1.1.x の場合は、CLASSPATH に次のものを含めてください。

     ```
     $ORACLE_HOME/jdbc/lib/classes111.zip
     $ORACLE_HOME/jdbc/lib/jta.zip
     $ORACLE_HOME/jdbc/lib/nls_charset11.zip
     $ORACLE_HOME/jdbc/lib/jndi.zip
     $ORACLE_HOME/lib/lclasses11.zip
     $ORACLE_HOME/lib/xmlparserv2.jar
     $ORACLE_HOME/lib/xschema.jar
     $ORACLE HOME/rdbms/jlib/aqapi11.jar
     $ORACLE_HOME/rdbms/jlib/jmscommon.jar
     $ORACLE_HOME/rdbms/jlib/aqxml.jar
     $ORACLE_HOME/rdbms/jlib/xsu111.jar
     $ORACLE_HOME/jis/lib/servlet.jar
     ```

サーブレットでは JDBC OCI ドライバを使用して Oracle9*i* サーバーに接続するため、サーブレットのホストとなるマシンに Oracle9*i* Client ライブラリをインストールする必要があります。LD_LIBRARY_PATH に $ORACLE_HOME/lib を含めます。

AQ へのインターネット・アクセスに関する詳細は、『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』を参照してください。

パッケージ oracle.AQ.xml の概要

次に、パッケージ oracle.AQ.xml の概要を示します。

クラスの概要	説明
インタフェース	-
AQxmlCallback	このインタフェースは、サーブレットが AQ 操作を実行する前後に起動されるコールバックを定義するために使用します。
クラス	-
AQxmlDataSource	サーブレットが AQ 操作の実行のために接続するバックエンド・データベースを指定するには、AQ データ・ソースを使用します。
AQxmlServlet	クライアントからの HTTP POST 要求を処理する AQ XML サーブレットです。Servlet 2.2 実装で使用します。
AQxmlServlet20	クライアントからの HTTP POST 要求を処理する AQ XML サーブレットです。Servlet 2.0 実装で使用します。
AQxmlCallbackContext	コールバック関数の前と後に渡されるコンテキストです。この CallbackContext には、解析された XML 文書の取出し、AQ データベースへの JDBC 接続の取得、サーブレットによって送信された応答ストリームのオーバーライドおよび応答用の XML スタイルシートの設定を行うメソッドがあります。
AQxmlDebug	AQ xml Debug クラス
例外	-
AQxmlException	AQ XML 例外

AQxmlCallback

構文

```
public interface AQxmlCallback
```

説明

このインタフェースは、サーブレットが AQ 操作を実行する前後に起動されるコールバックを定義するために使用します。setUserCallback メソッドを使用して、サーブレットの init メソッドにコールバックを定義します。コールバック・メソッドは、サーブレット要求ストリーム、サーブレット応答およびコールバック・コンテキストを取得します。CallbackContext には、解析された XML 文書の取出し、AQ データベースへの JDBC 接続の取得およびサーブレットによって送信された応答ストリームのオーバーライドを行うメソッドがあります。

メンバーの概要	説明
メソッド	-
afterAQOperation(HttpServletRequest, HttpServletResponse, AQxmlCallbackContext)	サーブレットによって AQ 操作が実行された後に起動されるコールバック
beforeAQOperation(HttpServletRequest, HttpServletResponse, AQxmlCallbackContext)	サーブレットによって AQ 操作が実行される前に起動されるコールバック

メソッド

afterAQOperation(HttpServletRequest, HttpServletResponse, AQxmlCallbackContext)

```
public void afterAQOperation(oracle.AQ.xml.HttpServletRequest request,
oracle.AQ.xml.HttpServletResponse response, AQxmlCallbackContext ctx)
```

サーブレットによって AQ 操作が実行された後に起動されるコールバック

パラメータ

`request` ― サーブレット要求

`response` ― サーブレット応答

`ctx` ― コールバック・コンテキスト

beforeAQOperation(HttpServletRequest, HttpServletResponse, AQxmlCallbackContext)

```
public void beforeAQOperation(oracle.AQ.xml.HttpServletRequest request,
oracle.AQ.xml.HttpServletResponse response, AQxmlCallbackContext ctx)
```

サーブレットによって AQ 操作が実行される前に起動されるコールバック

パラメータ

`request` — サーブレット要求

`response` — サーブレット応答

`ctx` — コールバック・コンテキスト

AQxmlDataSource

構文

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

java.lang.Object
  |
  +--oracle.AQ.xml.AQxmlDataSource
```

説明

サーブレットが AQ 操作の実行のために接続するバックエンド・データベースを指定するには、AQ データ・ソースを使用します。AQ データ・ソースには、データベース SID、ホスト名、リスナー・ポートおよび AQ サーブレット・スーパー・ユーザーのユーザー名およびパスワードが含まれています。AQ サーブレットは、JDBC OCI ドライバを使用してデータベースに接続します。このときに、接続キャッシュが作成されます。デフォルトの接続プールのサイズは 5 です。

メンバーの概要	説明
コンストラクタ	-
AQxmlDataSource(OracleOCIConnectionPool)	AQ データ・ソースを作成します。
AQxmlDataSource(String, String, String, String, String)	AQ データ・ソースを作成します。
メソッド	-
getCacheSize()	接続キャッシュのサイズを取得します。
getDBDrv()	データ・ソースが使用する JDBC ドライバを取得します。
getHost()	ホスト名を取得します。
getPort()	リスナー・ポートを取得します。
getSid()	データベース SID を取得します。
setCacheSize(int)	接続キャッシュのサイズを設定します。

継承メンバーの概要

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、wait

コンストラクタ

AQxmlDataSource(OracleOCIConnectionPool)

```
public  AQxmlDataSource(OracleOCIConnectionPool pool_ds)
```

OCI 接続プールを指定して AQ データ・ソースを作成します。

パラメータ

`pool_ds` ― OCI 接続プール

例外

AQxmlException ― データ・ソースの作成に失敗した場合

AQxmlDataSource(String, String, String, String, String)

```
public  AQxmlDataSource(java.lang.String user, java.lang.String password,
java.lang.String sid, java.lang.String host, java.lang.String port)
```

AQ データ・ソースを作成します。

パラメータ

`user` ― ユーザー名

`password` ― ユーザーのパスワード

`sid` ― データベースの SID

`host` ― ホスト名

`port` ― リスナー・ポート

例外

AQxmlException ― データ・ソースの作成に失敗した場合

メソッド

getCacheSize()

```
public int getCacheSize()
```
接続キャッシュのサイズを取得します。

getDBDrv()

```
public java.lang.String getDBDrv()
```
データ・ソースが使用する JDBC ドライバを取得します。

getHost()

```
public java.lang.String getHost()
```
ホスト名を取得します。

getPort()

```
public java.lang.String getPort()
```
リスナー・ポートを取得します。

getSid()

```
public java.lang.String getSid()
```
データベース SID を取得します。

setCacheSize(int)

```
public void setCacheSize(int csize)
```
接続キャッシュのサイズを設定します。

パラメータ

`csize` ― キャッシュ・サイズ

AQxmlCallbackContext

構文

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

java.lang.Object
  |
  +--oracle.AQ.xml.AQxmlCallbackContext
```

説明

前後のコールバック関数に渡されるコンテキストです。この CallbackContext には、解析された XML 文書の取出し、AQ データベースへの JDBC 接続の取得、サーブレットによって送信された応答ストリームのオーバーライドおよび応答用の XML スタイルシートの設定を行うメソッドがあります。

メンバーの概要	説明
メソッド	-
getDBConnection()	この要求に使用される JDBC 接続を取得します。ユーザーはこのデータベース接続を使用して SQL 操作を実行できます。
getOverrideAQResponseFlag()	AQ サーブレットから送り返される応答をオーバーライドするフラグを取得します。
getServerResponseDoc()	サーブレットから送り返される応答を表す AQxmlDocument を取得します。
getStyleSheetProcessingInstr()	XML 応答のスタイルシート処理命令を取得します。
parseRequestStream()	サーブレット要求の XML 文書を解析します。
setOverrideAQResponseFlag(boolean)	AQ サーブレットから送り返される応答をオーバーライドするフラグを設定します。
setStyleSheet(String, String)	XML 応答用のスタイルシートを設定します。
setStyleSheetProcessingInstr(String)	XML 応答用のスタイルシート処理命令を設定します。

継承メンバーの概要

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、wait

メソッド

getDBConnection()

```
public java.sql.Connection getDBConnection()
```

この要求に使用される JDBC 接続を取得します。ユーザーはこのデータベース接続を使用して SQL 操作を実行できます。操作は、AQ 操作と同じトランザクションの一部として実行されます。iDAP メッセージの AQ 操作がコミットまたは異常終了すると、この操作もコミットまたは異常終了します。ユーザーはこれらの接続でコミットおよびロールバックをコールすることはできません。コミットおよびロールバックを行うには、サーブレットに iDAP メッセージを送信する必要があります。

getOverrideAQResponseFlag()

```
public boolean getOverrideAQResponseFlag()
```

AQ サーブレットから送り返される応答をオーバーライドするフラグを取得します。

getServerResponseDoc()

```
public AQxmlDocument getServerResponseDoc()
```

サーブレットから送り返される応答を表す AQxmlDocument を取得します。これは `afterAQOperation` コールバックの中でのみ使用可能です。

getStyleSheetProcessingInstr()

```
public java.lang.String getStyleSheetProcessingInstr()
```

XML 応答のスタイルシート処理命令を取得します。

parseRequestStream()

```
public oracle.AQ.xml.Document parseRequestStream()
```

サーブレット要求の XML 文書を解析します。

setOverrideAQResponseFlag(boolean)

```
public void setOverrideAQResponseFlag(boolean value)
```

AQ サーブレットから送り返される応答をオーバーライドするフラグを設定します。AQ サーブレットはリクエスタに iDAP 応答を送信します。AQ によって送信される応答のかわりに、自分で応答を作成するには、コールバックによってこのフラグを設定できます。

setStyleSheet(String, String)

```
public void setStyleSheet(java.lang.String type, java.lang.String href)
```

XML 応答用のスタイルシートを設定します。この要求に対して送られる XML 応答用の XML スタイルシート処理命令を設定できます。

パラメータ

`type` ― スタイルシートのタイプ（例 : "text/xml"）

`href` ― スタイルシート href（例 : "http://www.aq.com/AQ/xslt.html"）

例外

AQxmlException ― 無効なパラメータが指定された場合

setStyleSheetProcessingInstr(String)

```
public void setStyleSheetProcessingInstr(java.lang.String proc_instr)
```

XML 応答用のスタイルシート処理命令を設定します。この要求に対して送られる XML 応答用の XML スタイルシート処理命令を設定できます。

パラメータ

`proc_instr` ― スタイルシート処理命令

（例 : "type=\"text/xsl\" href=\"http://www.oa.com/AQ/xslt23.html\""）

AQxmlServlet

構文

```
public class AQxmlServlet implements java.lang.Runnable
```

oracle.AQ.xml.AQxmlServlet

実装される全インタフェース

```
java.lang.Runnable
```

説明

クライアントからの HTTP POST 要求を処理する AQ XML サーブレットです。このサーブレットは、JavaSoft の Servlet2.2 標準を実装するどのサーブレット・エンジンにも配置できます。配置する前に、このサーブレットを拡張し、（データベース・インスタンスに接続するための）AQ データ・ソースを定義する必要があります。

メンバーの概要	説明
メソッド	-
doGet(HttpServletRequest, HttpServletResponse)	このメソッドは、HTTP GET 要求を処理します。
doPost(HttpServletRequest, HttpServletResponse)	このメソッドは、HTTP POST 要求を処理します。AQ XML サーブレットの主なエントリ・ポイントです。
getAQDataSource()	このサーブレットがデータベースへの接続に使用する AQ データ・ソースを取得します。
getEmailServerAddr()	電子メール・サーバーの IP アドレスを取得します。
getEmailServerHost()	電子メール・サーバーのホスト名を取得します。
getUserCallback()	ユーザーによって登録されたコールバックを取得します。
setAQDataSource(AQxmlDataSource)	サブクラスはサーブレットの init メソッド内でこのメソッドをコールして、データベース接続パラメータ（username/password、sid、portno など）を指定する必要があります。
setAQSchemaLocation(String)	AQ iDAP スキーマの場所を設定します。
setEmailServerAddr(String)	電子メール・サーバーの IP アドレスを設定します。

メンバーの概要	説明
setLdapContext(DirContext)	サーブレットの LDAP コンテキストを設定します。
setSessionMaxInactiveTime(int)	セッションがアクティブでない状態を保持できる最長時間を設定します。
setStyleSheet(String, String)	応答用のスタイルシートを設定します。
setStyleSheetProcessingInstr(String)	応答用のスタイルシート処理命令を設定します。
setUserCallback(AQxmlCallback)	コールバックを設定します。

メソッド

doGet(HttpServletRequest, HttpServletResponse)

```
protected void doGet(oracle.AQ.xml.HttpServletRequest request,
oracle.AQ.xml.HttpServletResponse response)
```

このメソッドは、HTTP GET 要求を処理します。サーブレットが正常に配置されたことをテストする目的でのみ使用します。通常、AQ 操作はすべて HTTP POST 要求として送信する必要があります。

doPost(HttpServletRequest, HttpServletResponse)

```
protected void doPost(oracle.AQ.xml.HttpServletRequest request,
oracle.AQ.xml.HttpServletResponse response)
```

このメソッドは、HTTP POST 要求を処理します。AQ XML サーブレットの主なエントリ・ポイントです。このルーチンは、iDAP スキーマに準拠した XML メッセージを含んだ text/xml タイプの着信ストリームを必要とします。

パラメータ

`request` ― http post 要求

`response` ― 応答オブジェクト。出力はこのオブジェクトより取得したストリームに書き込まれます。

例外

`ServletException` ― IOException

getAQDataSource()

```
public synchronized AQxmlDataSource getAQDataSource()
```
このサーブレットがデータベースへの接続に使用する AQ データ・ソースを取得します。

getEmailServerAddr()

```
public java.lang.String getEmailServerAddr()
```
電子メール・サーバーの IP アドレスを取得します。

getEmailServerHost()

```
public java.lang.String getEmailServerHost()
```
電子メール・サーバーのホスト名を取得します。

getUserCallback()

```
public final AQxmlCallback getUserCallback()
```
ユーザーによって登録されたコールバックを取得します。

setAQDataSource(AQxmlDataSource)

```
public final synchronized void setAQDataSource(AQxmlDataSource data_source)
```
サブクラスはサーブレットの `init` メソッド内でこのメソッドをコールして、データベース接続パラメータ（username/password、sid、portno など）を指定する必要があります。

パラメータ

`data_source` − AQ データ・ソース

setAQSchemaLocation(String)

```
public synchronized void setAQSchemaLocation(java.lang.String schema_location)
```

パラメータ

schema_location - AQ iDAP スキーマの場所を設定します。デフォルトで、aqxml.jar ファイル内の envelope.xsd、aqxml.xsd ファイルからスキーマが選択されます。

setEmailServerAddr(String)

```
public synchronized void setEmailServerAddr(java.lang.String ip_address)
```
電子メール・サーバーの IP アドレスを設定します。

パラメータ

`ip_address` ― 電子メール・サーバーの IP アドレス

setLdapContext(DirContext)

```
public final synchronized void setLdapContext(oracle.AQ.xml.DirContext ctx)
```

サーブレットの LDAP コンテキストを設定します。LDAP サーバー内で検索されるキューおよびトピックの別名が iDAP メッセージに含まれる場合は、サーブレットの init メソッドにこのコンテキストを設定する必要があります。

パラメータ

`ctx` ― LDAP ディレクトリ・コンテキスト

setSessionMaxInactiveTime(int)

```
protected synchronized void setSessionMaxInactiveTime(int secs)
```
セッションがアクティブでない状態を保持できる最長時間を設定します。セッションが指定した時間を超えてアクティブでない状態にあると、そのセッションは破壊され、まだコミットされていない操作はすべてロールバックされます。デフォルトでは 120 秒に設定されます。

パラメータ

`secs` ― 秒単位の時間。この値は 30 秒以下に設定できません。

setStyleSheet(String, String)

```
public synchronized void setStyleSheet(java.lang.String type, java.lang.String href)
```

応答用のスタイルシートを設定します。サーブレットの `init` メソッド内でコールして、サーブレットによって送信されるすべての XML 応答に XML スタイルシート処理命令を設定できます。

パラメータ

`type` ― スタイルシートのタイプ（"text/xml" など）

`href` ― スタイルシート href（"http://www.aq.com/AQ/xslt.html" など）

例外

AQxmlException ― 無効なパラメータが指定された場合

setStyleSheetProcessingInstr(String)

```
public void setStyleSheetProcessingInstr(java.lang.String proc_instr)
```

応答用のスタイルシート処理命令を設定します。サーブレットの init メソッド内でコールして、サーブレットによって送信されるすべての XML 応答に XML スタイルシート処理命令を設定できます。

パラメータ

proc_instr ― スタイルシート処理命令（"type=\"text/xsl\" href=\"http://www.oa.com/AQ/xslt23.html\"" など）

setUserCallback(AQxmlCallback)

```
public final void setUserCallback(AQxmlCallback callback)
```

コールバックを設定します。コールバック・メソッドは、AQ 操作の前後にコールされます。

パラメータ

callback ― コールバック

AQxmlServlet20

構文

```
public class AQxmlServlet20 implements java.lang.Runnable
```

oracle.AQ.xml.AQxmlServlet20

実装される全インタフェース

```
java.lang.Runnable
```

説明

クライアントからの HTTP POST 要求を処理する AQ XML サーブレットです。このサーブレットは、JavaSoft の Servlet2.0 標準を実装するどのサーブレット・エンジンにも配置できます。配置する前に、このサーブレットを拡張し、（データベース・インスタンスに接続するための）AQ データ・ソースを定義する必要があります。

メンバーの概要	説明
メソッド	-
doGet(HttpServletRequest, HttpServletResponse)	このメソッドは、HTTP GET 要求を処理します。
doPost(HttpServletRequest, HttpServletResponse)	このメソッドは、HTTP POST 要求を処理します。AQ XML サーブレットの主なエントリ・ポイントです。
getAQDataSource()	このサーブレットがデータベースへの接続に使用する AQ データ・ソースを取得します。
getEmailServerAddr()	電子メール・サーバーの IP アドレスを取得します。
getEmailServerHost()	電子メール・サーバーのホスト名を取得します。
getUserCallback()	ユーザーによって登録されたコールバックを取得します。
setAQDataSource(AQxmlDataSource)	サブクラスは init メソッド内でこのメソッドをコールして、データベース接続パラメータ（username/password、sid、portno など）を指定する必要があります。
setAQSchemaLocation(String)	AQ iDAP スキーマの場所を設定します。
setEmailServerAddr(String)	電子メール・サーバーの IP アドレスを設定します。

メンバーの概要	説明
setLdapContext(DirContext)	サーブレットの LDAP コンテキストを設定します。
setManualInvalidation(boolean)	手動によるセッションの無効化をオン / オフにするフラグを設定します。Servlet2.0 実装の場合、最長非アクティブ時間を超えてアクティブでない状態にあるセッションを自動的に無効化するスレッドが起動されます。
setSessionMaxInactiveTime(int)	セッションがアクティブでない状態を保持できる最長時間を設定します。
setStyleSheet(String, String)	応答用のスタイルシートを設定します。
setStyleSheetProcessingInstr(String)	応答用のスタイルシート処理命令を設定します。
setUserCallback(AQxmlCallback)	コールバックを設定します。

メソッド

doGet(HttpServletRequest, HttpServletResponse)

```
protected void doGet(oracle.AQ.xml.HttpServletRequest request,
oracle.AQ.xml.HttpServletResponse response)
```

このメソッドは、HTTP GET 要求を処理します。サーブレットが正常に配置されたことをテストする目的でのみ使用します。通常、AQ 操作はすべて HTTP POST 要求として送信する必要があります。

doPost(HttpServletRequest, HttpServletResponse)

```
protected void doPost(oracle.AQ.xml.HttpServletRequest request,
oracle.AQ.xml.HttpServletResponse response)
```

このメソッドは、HTTP POST 要求を処理します。AQ XML サーブレットの主なエントリ・ポイントです。このルーチンは、iDAP スキーマに準拠した XML メッセージを含んだ text/xml タイプの着信ストリームを必要とします。

パラメータ

`request` ― HTTP POST 要求

`response` ― 応答オブジェクト。出力はこのオブジェクトより取得したストリームに書き込まれます。

例外

ServletException — IOException

getAQDataSource()

```
public synchronized AQxmlDataSource getAQDataSource()
```

このサーブレットがデータベースへの接続に使用する AQ データ・ソースを取得します。

getEmailServerAddr()

```
public java.lang.String getEmailServerAddr()
```

電子メール・サーバーの IP アドレスを取得します。

getEmailServerHost()

```
public java.lang.String getEmailServerHost()
```

電子メール・サーバーのホスト名を取得します。

getUserCallback()

```
public final AQxmlCallback getUserCallback()
```

ユーザーによって登録されたコールバックを取得します。

setAQDataSource(AQxmlDataSource)

```
public final synchronized void setAQDataSource(AQxmlDataSource data_source)
```

サブクラスはサーブレットの init メソッド内でこのメソッドをコールして、データベース接続パラメータ（username/password、sid、portno など）を指定する必要があります。

パラメータ

data_source — AQ データ・ソース

setAQSchemaLocation(String)

```
public synchronized void setAQSchemaLocation(java.lang.String schema_location)
```

AQ iDAP スキーマの場所を設定します。デフォルトで、aqxml.jar ファイル内の envelope.xsd、aqxml.xsd ファイルからスキーマが選択されます。

setEmailServerAddr(String)

```
public synchronized void setEmailServerAddr(java.lang.String ip_address)
```
電子メール・サーバーの IP アドレスを設定します。

パラメータ

`ip_address` − 電子メール・サーバーの IP アドレス

setLdapContext(DirContext)

```
public final synchronized void setLdapContext(oracle.AQ.xml.DirContext ctx)
```
サーブレットの LDAP コンテキストを設定します。LDAP サーバー内で検索されるキューおよびトピックの別名が iDAP メッセージに含まれる場合は、サーブレットの `init` メソッドにこのコンテキストを設定する必要があります。

パラメータ

`ctx` − LDAP ディレクトリ・コンテキスト

setManualInvalidation(boolean)

```
protected synchronized void setManualInvalidation(boolean flag)
```
手動によるセッションの無効化をオン / オフにするフラグを設定します。Servlet2.0 実装の場合、最長非アクティブ時間を超えてアクティブでない状態にあるセッションを自動的に無効化するスレッドが起動されます。サーブレット・コンテナでセッションの無効化を実行する場合には、このフラグを false に設定できます。

パラメータ

`flag` − true の場合は、手動によるセッションの無効化がオンに設定されていることを示し、false の場合は、手動によるセッションの無効化がオフに設定されていることを示します。

setSessionMaxInactiveTime(int)

```
protected synchronized void setSessionMaxInactiveTime(int secs)
```
セッションがアクティブでない状態を保持できる最長時間を設定します。セッションが指定した時間を超えてアクティブでない状態にあると、そのセッションは破壊され、まだコミットされていない操作はすべてロールバックされます。デフォルトでは 120 秒に設定されます。

パラメータ

`secs` − 秒単位の時間。この値は 30 秒以下に設定できません。

setStyleSheet(String, String)

```
public synchronized void setStyleSheet(java.lang.String type, java.lang.String href)
```
応答用のスタイルシートを設定します。サーブレットの init メソッド内でコールして、サーブレットによって送信されるすべての XML 応答に XML スタイルシート処理命令を設定できます。

パラメータ

type ― スタイルシートのタイプ（"text/xml" など）

href ― スタイルシート href（"http://www.aq.com/AQ/xslt.html" など）

例外

AQxmlException ― 無効なパラメータが指定された場合

setStyleSheetProcessingInstr(String)

```
public void setStyleSheetProcessingInstr(java.lang.String proc_instr)
```
応答用のスタイルシート処理命令を設定します。サーブレットの init メソッド内でコールして、サーブレットによって送信されるすべての XML 応答に XML スタイルシート処理命令を設定できます。

パラメータ

proc_instr ― スタイルシート処理命令（"type=\"text/xsl\" href=\"http://www.oa.com/AQ/xslt23.html\"" など）

setUserCallback(AQxmlCallback)

```
public final void setUserCallback(AQxmlCallback callback)
```
コールバックを設定します。コールバック・メソッドは、AQ 操作の前後にコールされます。

パラメータ

callback ― コールバック

AQxmlDebug

構文

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

java.lang.Object
  |
  +--oracle.AQ.xml.AQxmlDebug
```

説明

このクラスには、AQ サーブレットのトレース・レベルを設定するための静的メソッドがあります。オラクル社カスタマ・サポート・センターから指示がないかぎり、このクラスは使用しないでください。

メンバーの概要	説明
メソッド	-
getLogStream()	トレース情報を書き込むログ・ストリームを取得します。
getPrintWriter()	印刷ストリームを取得します。
getTraceLevel()	トレース・レベルを取得します。
setDebug(boolean)	デバッグ・フラグを設定します。
setLogStream(OutputStream)	トレース情報を書き込むログ・ストリームを設定します。
setTraceLevel(int)	トレース・レベルー AQ_ORA_TR1..5 を設定します。

継承メンバーの概要
クラス java.lang.Object から継承されるメソッド
clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、wait

メソッド

getLogStream()

```
public static java.io.OutputStream getLogStream()
```
トレース情報を書き込むログ・ストリームを取得します。

getPrintWriter()

```
public static java.io.PrintWriter getPrintWriter()
```
印刷ストリームを取得します。

getTraceLevel()

```
public static int getTraceLevel()
```
トレース・レベルを取得します。

setDebug(boolean)

```
public static void setDebug(boolean val)
```
デバッグ・フラグを設定します。

setLogStream(OutputStream)

```
public static void setLogStream(java.io.OutputStream output_stream)
```
トレース情報を書き込むログ・ストリームを設定します。

パラメータ

`output_stream` ― ログ・ストリーム

setTraceLevel(int)

```
public static void setTraceLevel(int level)
```
トレース・レベルを設定します。

0 ― トレースなし（デフォルト）

1 ― 致命的エラー

2 ― その他のエラー、imp メッセージ

3 ― 例外トレース、その他のトレース情報

4 ― メソッドの開始 / 終了

5 ― スタック・トレース、変数情報の出力

AQxmlException

構文

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

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--oracle.AQ.xml.AQxmlException
```

実装される全インタフェース

```
java.io.Serializable
```

説明

AQ XML 例外

メンバーの概要	説明
メソッド	-
getErrorCode()	例外に対応する Oracle エラー・コードを取得します。
getNextException()	リンクされている例外を取得します。
setNextException(Exception)	リンクされている例外を設定します。

継承メンバーの概要

クラス java.lang.Throwable から継承されるメソッド

fillInStackTrace、getLocalizedMessage、getMessage、printStackTrace、toString

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、wait

メソッド

getErrorCode()

```
public int getErrorCode()
```
例外に対応する Oracle エラー・コードを取得します。

getNextException()

```
public java.lang.Exception getNextException()
```
リンクされている例外を取得します。

setNextException(Exception)

```
protected void setNextException(java.lang.Exception exc)
```
リンクされている例外を設定します。

パラメータ

exc ― リンクされている例外

4

パッケージ oracle.jms

この章では、パッケージ oracle.jms に含まれる Oracle Java Message Service（OJMS）のインタフェースとクラスについて説明します。Oracle JMS インタフェースは、標準の JMS インタフェースを拡張して、Oracle9*i* Advanced Queuing（AQ）管理操作や、javax.jms パッケージに含まれるパブリック標準に組み込まれていない他の AQ 機能をサポートします。

この章は、次の項で構成されています。

パッケージ oracle.jms の説明

Oracle パッケージ oracle.jms は、Java Message Service（JMS）標準に基づいた一連のインタフェースと関連するセマンティクスを提供します。これらのインタフェースは、JMS クライアントが Oracle9*i* Advanced Queuing などのエンタープライズ・メッセージング製品の機能にアクセスする方法を定義します。Advanced Queuing（AQ）は、Oracle9*i* データベース固有のデータベース統合型メッセージ・キューイング機能です。Oracle は、標準の JMS インタフェースをサポートし、AQ 管理操作や、パブリック標準に組み込まれていない他の AQ 機能をサポートする拡張機能を備えています。

> **関連項目：**『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』

標準 JMS および Oracle JMS パッケージへのアクセス

Oracle JMS は、JDBC を使用してデータベースに接続します。したがって、そのアプリケーションを次のように実行できます。

- OCI9 または Thin JDBC ドライバを使用してデータベースの外部で実行する。
- Oracle サーバーのドライバを使用して Oracle8*i* または Oracle9*i* の Oracle JVM 内で実行する。

標準の JMS インタフェースは、javax.jms パッケージに含まれています（詳細は、Sun 社の J2EE のマニュアルを参照してください）。Oracle JMS インタフェースは javax.jms を拡張するインタフェースで、oracle.jms パッケージに含まれています。

OCI9 または Thin JDBC ドライバの使用

データベースの外部で実行しているクライアントで JMS インタフェースを使用するには、適切な JDBC ドライバ、JNDI ファイルおよび次のような AQ jar ファイルを CLASSPATH に含める必要があります （CLASSPATH は、アプリケーションの実行に必要なクラスを検索するために、JVM が使用するオペレーティング・システム環境変数です）。

JDK 1.3 の場合は、次のファイルを含めてください。

```
$ORACLE_HOME/rdbms/jlib/jmscommon.jar
$ORACLE_HOME/rdbms/jlib/aqapi13.jar
$ORACLE_HOME/jdbc/lib/jndi.zip
$ORACLE_HOME/jdbc/lib/classes13.zip
```

JDK 1.2 の場合は、次のファイルを含めてください。

```
$ORACLE_HOME/rdbms/jlib/jmscommon.jar
$ORACLE_HOME/rdbms/jlib/aqapi12.jar
$ORACLE_HOME/jdbc/lib/jndi.zip
$ORACLE_HOME/jdbc/lib/classes12.zip
```

JDK 1.1 の場合は、次のファイルを含めてください。

```
$ORACLE_HOME/rdbms/jlib/jmscommon.jar
$ORACLE_HOME/rdbms/jlib/aqapi11.jar
$ORACLE_HOME/jdbc/lib/jndi.zip
$ORACLE_HOME/jdbc/lib/classes11.zip
```

Oracle JVM での Oracle サーバーのドライバの使用

Oracle JVM 内でアプリケーションを実行している場合は、Oracle JVM のインストール時に自動的にロードされている Oracle JMS クラスにアクセスできます。Oracle JMS クラスにアクセスできない場合は、loadjava ユーティリティを使用して、まず jmscommon.jar をロードしてから、aqapi.jar をロードする必要があります。

必要な権限

Oracle JMS インタフェースを使用するには、DBMS_AQIN パッケージと DBMS_AQJMS パッケージの実行権限が必要です。この権限は AQ_USER_ROLE または AQ_ADMINSTRATOR_ROLE を使用して取得することもできます。

また、メッセージの送信または受信に対する適切なシステム権限とキューまたはトピック権限も必要です。

パッケージ oracle.jms の概要

表 4-1　oracle.jms のインタフェースの概要

インタフェース	説明
AdtMessage	このインタフェースは Message インタフェースを拡張し、Oracle のオブジェクト型ペイロードを含んだメッセージを表します。JMS に対する AQ 拡張インタフェースです。
AQjmsQueueReceiver	このインタフェースは javax.jms.QueueReceiver を拡張し、JMS に対する AQ 拡張を定義します。クライアントは、QueueReceiver を使用して、キューに配信されたメッセージを受信します。
AQjmsQueueSender	このインタフェースは QueueSender を拡張し、JMS に対する AQ 拡張を定義します。クライアントは、QueueSender を使用してキューにメッセージを送信します。
AQjmsTopicPublisher	このインタフェースは TopicPublisher を拡張し、JMS に対する AQ 拡張を定義します。クライアントは、TopicPublisher を使用して、トピックにメッセージを公開します。
AQjmsTopicReceiver	このインタフェースは TopicReceiver インタフェースを拡張し、（Point-to-Multipoint 通信において）リモートのサブスクライバと明示的に指定したレシピエントに対する AQ 拡張を定義します。TopicReceiver を使用して、トピックからメッセージを受信します。
AQjmsTopicSubscriber	このインタフェースは TopicSubscriber を拡張し、JMS に対する AQ 拡張を定義します。クライアントは、TopicSubscriber を使用して、トピックに公開されているメッセージを受信します。
TopicBrowser	このインタフェースは MessageConsumer を拡張し、リモートのサブスクライバがトピック上のメッセージを削除せずに参照できるようにします。
TopicReceiver	このインタフェースは MessageConsumer を拡張します。（Point-to-Multipoint 通信において）リモートのサブスクライバと明示的に指定したレシピエントは、このインタフェースを使用してメッセージを受信します。

表 4-2 oracle.jms のクラスの概要

クラス	説明
AQjmsAdtMessage	このクラスは、AdtMessage インタフェースを実装します。AdtMessage は、Oracle のオブジェクト型ペイロードを含んだメッセージを送信するために使用します。
AQjmsAgent	このクラスは、Destination インタフェースを実装します。リモートのサブスクライバおよび返信先を定義するために使用します。
AQjmsBytesMessage	このクラスは、BytesMessage インタフェースを実装します。BytesMessage は、未解釈のバイト・ストリームを含んだメッセージを送信するために使用します。
AQjmsConnection	このクラスは、Connection インタフェースを実装します。これは、JMS プロバイダへのアクティブ接続です。
AQjmsConnectionMetaData	クラス AQjmsConnectionMetaData は、JMS Connection に使用可能なメタデータ情報を表します。
AQjmsConstants	このクラスは、oracle.jms パッケージで使用される定数を定義します。
AQjmsConsumer	このクラスは、MessageConsumer インタフェースを実装します。
AQjmsDestination	このクラスは、管理オブジェクト、キューおよびトピックを実装します。
AQjmsDestinationProperty	このクラスは、宛先のプロパティを定義します。
AQjmsFactory	このクラスは、Oracle による JMS の実装で、管理される ConnectionFactory オブジェクトにアクセスするために使用します。
AQjmsMapMessage	このクラスは、MapMessage インタフェースを実装します。MapMessage は、名前 - 値のペアを送信するために使用します。名前は String 型、値は Java プリミティブ型です。
AQjmsMessage	このクラスは、Message インタフェースを実装します。これは、すべての JMS メッセージのスーパークラスです。
AQjmsObjectMessage	このクラスは、ObjectMessage インタフェースを実装します。ObjectMessage は、シリアル化可能な Java オブジェクトを含んだメッセージを送信するために使用します。
AQjmsOracleDebug	AQ Oracle デバッグ・クラス。オラクル社カスタマ・サポート・センターから指示がないかぎり、このクラスは使用しないでください。

表 4-2 oracle.jms のクラスの概要（続き）

クラス	説明
AQjmsProducer	このクラスは、MessageProducer インタフェースを実装します。MessageProducer は、宛先にメッセージを送信するために使用します。
AQjmsQueueBrowser	このクラスは、QueueBrowser インタフェースを実装します。QueueBrowser は、キュー内のメッセージを、そのメッセージを削除せずに参照するために使用します。
AQjmsQueueConnectionFactory	このクラスは、QueueConnectionFactory インタフェースを実装します。QueueConnectionFactory は、QueueConnections を作成するために使用します。
AQjmsSession	このクラスは、javax.jms.Session インタフェースを実装します。JMS セッションは、メッセージを生成および使用するための単一スレッドのコンテキストです。
AQjmsStreamMessage	このクラスは、StreamMessage インタフェースを実装します。StreamMessage は、Java プリミティブ型のストリームを送信するために使用します。
AQjmsTextMessage	このクラスは、TextMessage インタフェースを実装します。TextMessage は、java.lang.StringBuffer を含んだメッセージを送信するために使用します。
AQjmsTopicBrowser	このクラスは、TopicBrowser インタフェースを実装します。TopicBrowser は、トピック内のメッセージを、そのメッセージを削除せずに参照するために使用します。
AQjmsTopicConnectionFactory	このクラスは、TopicConnectionFactory インタフェースを実装します。TopicConnectionFactory は、トピック・コネクションを作成するために使用します。

表 4-3 oracle.jms の例外

例外	説明
AQjmsException	この例外は JMSException を拡張し、Oracle エラー・コードを追加します。これは、すべての JMS 例外の親クラスです。
AQjmsIllegalStateException	この例外は、IllegalStateException を拡張します。OJMS メソッドが無効または不適切な時間に起動された場合、または要求された操作に対して OJMS が適切な状態ではない場合に発生します。

表 4-3 oracle.jms の例外（続き）

例外	説明
AQjmsInvalidDestinationException	この例外は、InvalidDestinationException を拡張します。宛先が無効な場合に発生します。
AQjmsInvalidSelectorException	この例外は、InvalidSelectorException を拡張します。指定した MessageSelector が無効な場合に発生します。
AQjmsMessageEOFException	この例外は、MessageEOFException を拡張します。StreamMessage または BytesMessage の読込み時に、予期しないストリームの終了に到達した場合に発生します。
AQjmsMessageFormatException	この例外は、MessageFormatException を拡張します。クライアントが、メッセージでサポートされていないデータ型を使用しようとしたとき、またはメッセージ内のデータを異なる型で読み込もうとしたときに発生します。
AQjmsMessageNotReadableException	この例外は、MessageNotReadableException を拡張します。クライアントが、書込み専用メッセージを読み込もうとしたときに発生します。
AQjmsMessageNotWriteableException	この例外は、MessageNotWriteableException を拡張します。クライアントが、読取り専用メッセージを書き込もうとしたときに発生します。

AdtMessage

構文

```
public interface AdtMessage extends javax.jms.Message
```

全スーパーインタフェース

```
javax.jms.Message
```

既知の全実装クラス

```
AQjmsAdtMessage
```

説明

このインタフェースは Message インタフェースを拡張し、Oracle のオブジェクト型ペイロードを含んだメッセージを表します。

メンバーの概要	説明
メソッド	-
getAdtPayload()	このユーザー定義型メッセージのデータを含んだ CustomDatum オブジェクトを取得します。
setAdtPayload(CustomDatum)	このユーザー定義型メッセージのデータを含んだ CustomDatum オブジェクトを設定します。

継承メンバーの概要

インタフェース javax.jms.Message から継承されるフィールド

DEFAULT_DELIVERY_MODE、DEFAULT_PRIORITY、DEFAULT_TIME_TO_LIVE

インタフェース javax.jms.Message から継承されるメソッド

継承メンバーの概要（続き）

clearBody、clearProperties、getBooleanProperty、getByteProperty、getDoubleProperty、getFloatProperty、getIntProperty、getJMSCorrelationID、getJMSCorrelationIDAsBytes、getJMSDeliveryMode、getJMSDestination、getJMSExpiration、getJMSMessageID、getJMSPriority、getJMSRedelivered、getJMSReplyTo、getJMSTimestamp、getJMSType、getLongProperty、getObjectProperty、getPropertyNames、getShortProperty、getStringProperty、propertyExists、setBooleanProperty、setByteProperty、setDoubleProperty、setFloatProperty、setIntProperty、setJMSCorrelationID、setJMSCorrelationIDAsBytes、setJMSDeliveryMode、setJMSDestination、setJMSExpiration、setJMSMessageID、setJMSPriority、setJMSRedelivered、setJMSReplyTo、setJMSTimestamp、setJMSType、setLongProperty、setObjectProperty、setShortProperty、setStringProperty

メソッド

getAdtPayload()

```
public oracle.sql.CustomDatum getAdtPayload()
```

このユーザー定義型メッセージのデータを含んだ CustomDatum オブジェクトを取得します。

戻り値

このメッセージのデータを含んだオブジェクト

例外

JMSException － 内部 JMS エラーのために、JMS がオブジェクトの取得に失敗した場合

setAdtPayload(CustomDatum)

```
public void setAdtPayload(oracle.sql.CustomDatum payload)
```

このユーザー定義型メッセージのデータを含んだ CustomDatum オブジェクトを設定します。

ユーザー定義型は、非同期データ転送のためにあります。

パラメータ

payload － メッセージのデータ（オブジェクトが、CustomDatum インタフェースを実装している必要があります)。このペイロードは、キューまたはトピックのペイロード型として定義されているユーザー定義型を表す Java オブジェクトであることが必要です。

例外

`JMSException` ― JMS がユーザー定義型ペイロードの設定に失敗した場合

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

AQjmsAdtMessage

構文

```
public class AQjmsAdtMessage extends AQjmsMessage implements AdtMessage

java.lang.Object
  |
  +--AQjmsMessage
        |
        +--oracle.jms.AQjmsAdtMessage
```

実装される全インタフェース

AdtMessage, javax.jms.Message

説明

このクラスは、AdtMessage インタフェースを実装します。AdtMessage は、Oracle のオブジェクト型ペイロードを含んだメッセージを送信するために使用します。

メンバーの概要	説明
メソッド	-
clearBody()	メッセージ本体を消去します。
getAdtPayload()	このユーザー定義型メッセージのデータを含んだ CustomDatum オブジェクトを取得します。
getBooleanProperty(String)	指定した名前の boolean 型プロパティの値を戻します。
getByteProperty(String)	指定した名前の byte 型プロパティの値を戻します。
getDoubleProperty(String)	指定した名前の double 型プロパティの値を戻します。
getFloatProperty(String)	指定した名前の float 型プロパティの値を戻します。
getIntProperty(String)	指定した名前の int 型プロパティの値を戻します。
getJMSReplyTo()	このメッセージの返信先を取得します。
getJMSType()	メッセージ・タイプを取得します。
getLongProperty(String)	指定した名前の long 型プロパティの値を戻します。
getObjectProperty(String)	指定した名前の Java オブジェクト型プロパティの値を戻します。

メンバーの概要（続き）	説明
getPropertyNames()	すべてのプロパティ名の一覧を戻します。
getShortProperty(String)	指定した名前の short 型プロパティの値を戻します。
getStringProperty(String)	指定した名前の String 型プロパティの値を戻します。
propertyExists(String)	プロパティの値が存在するかどうかをチェックします。
setAdtPayload(CustomDatum)	このユーザー定義型メッセージのデータを含んだ CustomDatum オブジェクトを設定します。
setBooleanProperty(String, boolean)	指定した名前の boolean 型プロパティの値をメッセージに設定します。
setByteProperty(String, byte)	指定した名前の byte 型プロパティの値をメッセージに設定します。
setDoubleProperty(String, double)	指定した名前の double 型プロパティの値をメッセージに設定します。
setFloatProperty(String, float)	指定した名前の float 型プロパティの値をメッセージに設定します。
setIntProperty(String, int)	指定した名前の int 型プロパティの値をメッセージに設定します。
setJMSReplyTo(Destination)	このメッセージの返信先を設定します。
setJMSType(String)	メッセージ・タイプを設定します。
setLongProperty(String, long)	指定した名前の long 型プロパティの値をメッセージに設定します。
setObjectProperty(String, Object)	指定した名前の Java オブジェクト・プロパティの値をメッセージに設定します。
setShortProperty(String, short)	指定した名前の short 型プロパティの値をメッセージに設定します。
setStringProperty(String, String)	指定した名前の String 型プロパティの値をメッセージに設定します。

継承メンバーの概要

インタフェース javax.jms.Message から継承されるフィールド

DEFAULT_DELIVERY_MODE、DEFAULT_PRIORITY、DEFAULT_TIME_TO_LIVE

クラス AQjmsMessage から継承されるメソッド

継承メンバーの概要（続き）

clearProperties()、getJMSCorrelationID()、getJMSCorrelationIDAsBytes()、getJMSDeliveryMode()、getJMSDestination()、getJMSExpiration()、getJMSMessageID()、getJMSMessageIDAsBytes()、getJMSPriority()、getJMSRedelivered()、getJMSTimestamp()、getSenderID()、setJMSCorrelationID(String)、setJMSCorrelationIDAsBytes(byte[])、setJMSDestination(Destination)、setJMSExpiration(long)、setJMSMessageID(String)、setJMSPriority(int)、setJMSRedelivered(boolean)、setJMSTimestamp(long)、setSenderID(AQjmsAgent)

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、wait

インタフェース javax.jms.Message から継承されるメソッド

clearProperties、getJMSCorrelationID、getJMSCorrelationIDAsBytes、getJMSDeliveryMode、getJMSDestination、getJMSExpiration、getJMSMessageID、getJMSPriority、getJMSRedelivered、getJMSTimestamp、setJMSCorrelationID、setJMSCorrelationIDAsBytes、setJMSDeliveryMode、setJMSDestination、setJMSExpiration、setJMSMessageID、setJMSPriority、setJMSRedelivered、setJMSTimestamp

メソッド

clearBody()

```
public void clearBody()
```
メッセージ本体を消去します。メッセージの他の部分はすべてそのまま残ります。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearBody()

オーバーライド
クラス AQjmsMessage 内の clearBody()

例外

JMSException ー 内部 JMS エラーのために、JMS がメッセージ本体の消去に失敗した場合

getAdtPayload()

```
public oracle.sql.CustomDatum getAdtPayload()
```

このユーザー定義型メッセージのデータを含んだ CustomDatum オブジェクトを取得します。

指定方法

インタフェース AdtMessage 内の getAdtPayload()

戻り値

このメッセージのデータを含んだオブジェクト

例外

JMSException ー 内部 JMS エラーのために、JMS がオブジェクトの取得に失敗した場合

getBooleanProperty(String)

```
public boolean getBooleanProperty(java.lang.String name)
```

指定した名前の boolean 型プロパティの値を戻します。

指定方法

インタフェース javax.jms.Message 内の
javax.jms.Message.getBooleanProperty(java.lang.String)

オーバーライド

クラス AQjmsMessage 内の getBooleanProperty(String)

パラメータ

name ー boolean 型プロパティの名前

戻り値

指定した名前の boolean 型プロパティの値

例外

JMSException ー 内部 JMS エラーのために、JMS がプロパティの取得に失敗した場合

MessageFormatException ー この型変換が無効な場合

getByteProperty(String)

```
public byte getByteProperty(java.lang.String name)
```
指定した名前の byte 型プロパティの値を戻します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getByteProperty(java.lang.String)

オーバーライド
クラス AQjmsMessage 内の getByteProperty(String)

パラメータ
`name` ― byte 型プロパティの名前

戻り値
指定した名前の byte 型プロパティの値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がプロパティの取得に失敗した場合

`MessageFormatException` ― この型変換が無効な場合

getDoubleProperty(String)

```
public double getDoubleProperty(java.lang.String name)
```
指定した名前の double 型プロパティの値を戻します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.getDoubleProperty(java.lang.String)

オーバーライド
クラス AQjmsMessage 内の getDoubleProperty(String)

パラメータ
`name` ― double 型プロパティの名前

戻り値
指定した名前の double 型プロパティの値

例外

JMSException ー 内部 JMS エラーのために、JMS がプロパティの取得に失敗した場合

MessageFormatException ー この型変換が無効な場合

getFloatProperty(String)

```
public float getFloatProperty(java.lang.String name)
```
指定した名前の float 型プロパティの値を戻します。

指定方法

インタフェース javax.jms.Message 内の javax.jms.Message.getFloatProperty(java.lang.String)

オーバーライド

クラス AQjmsMessage 内の getFloatProperty(String)

パラメータ

name ー float 型プロパティの名前

戻り値

指定した名前の float 型プロパティの値

例外

JMSException ー 内部 JMS エラーのために、JMS がプロパティの取得に失敗した場合

MessageFormatException ー この型変換が無効な場合

getIntProperty(String)

```
public int getIntProperty(java.lang.String name)
```
指定した名前の int 型プロパティの値を戻します。

指定方法

インタフェース javax.jms.Message 内の javax.jms.Message.getIntProperty(java.lang.String)

オーバーライド

クラス AQjmsMessage 内の getIntProperty(String)

パラメータ

name ー int 型プロパティの名前

戻り値
指定した名前の int 型プロパティの値

例外
`JMSException` － 内部 JMS エラーのために、JMS がプロパティの取得に失敗した場合

`MessageFormatException` － この型変換が無効な場合

getJMSReplyTo()

```
public javax.jms.Destination getJMSReplyTo()
```
このメッセージの返信先を取得します。このメソッドは、このリリースの AdtMessages ではサポートされていません。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSReplyTo()

オーバーライド
クラス AQjmsMessage 内の getJMSReplyTo()

例外
`JMSException` － AdtMessage ではサポートされていません。

getJMSType()

```
public java.lang.String getJMSType()
```
メッセージ・タイプを取得します。このメソッドは、このリリースの AdtMessages ではサポートされていません。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSType()

オーバーライド
クラス AQjmsMessage 内の getJMSType()

戻り値
メッセージ・タイプ

例外
`JMSException` － AdtMessage ではサポートされていません。

getLongProperty(String)

```
public long getLongProperty(java.lang.String name)
```
指定した名前の long 型プロパティの値を戻します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.getLongProperty(java.lang.String)

オーバーライド
クラス AQjmsMessage 内の getLongProperty(String)

パラメータ
`name` ― long 型プロパティの名前

戻り値
指定した名前の long 型プロパティの値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がプロパティの取得に失敗した場合

`MessageFormatException` ― この型変換が無効な場合

getObjectProperty(String)

```
public java.lang.Object getObjectProperty(java.lang.String name)
```
指定した名前のプロパティの値を Java オブジェクト型で戻します。

このメソッドは、`setObject` メソッド・コールまたは基本型設定メソッドでプロパティとしてメッセージ内に格納されていたオブジェクトを、オブジェクト化された形式で戻すために使用できることに注意してください。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.getObjectProperty(java.lang.String)

オーバーライド
クラス AQjmsMessage 内の getObjectProperty(String)

パラメータ
`name` ― Java オブジェクト・プロパティの名前

戻り値
指定された名前の Java オブジェクト・プロパティの値をオブジェクト化された形式で戻します（つまり、int で設定された場合は、Integer が戻されます）。この名前のプロパティがない場合は、null が戻されます。

例外
`JMSException` － 内部 JMS エラーのために、JMS がプロパティの取得に失敗した場合

getPropertyNames()

```
public synchronized java.util.Enumeration getPropertyNames()
```
すべてのプロパティ名の一覧を戻します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getPropertyNames()

オーバーライド
クラス AQjmsMessage 内の getPropertyNames(String)

戻り値
プロパティ値のすべての名前の一覧

例外
`JMSException` － 内部 JMS エラーのために、JMS がプロパティ名の取得に失敗した場合

getShortProperty(String)

```
public short getShortProperty(java.lang.String name)
```
指定した名前の short 型プロパティの値を戻します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.getShortProperty(java.lang.String)

オーバーライド
クラス AQjmsMessage 内の getShortProperty(String)

パラメータ
`name` － short 型プロパティの名前

戻り値
指定した名前の short 型プロパティの値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がプロパティの取得に失敗した場合

`MessageFormatException` ― この型変換が無効な場合

getStringProperty(String)

```
public java.lang.String getStringProperty(java.lang.String name)
```
指定した名前の String 型プロパティの値を戻します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.getStringProperty(java.lang.String)

オーバーライド
クラス AQjmsMessage 内の getStringProperty(String)

パラメータ
`name` ― String 型プロパティの名前

戻り値
指定した名前の String 型プロパティの値。この名前のプロパティがない場合は、null が戻されます。

例外
`JMSException` ― 内部 JMS エラーのために、JMS がプロパティの取得に失敗した場合

`MessageFormatException` ― この型変換が無効な場合

propertyExists(String)

```
public boolean propertyExists(java.lang.String name)
```
プロパティの値が存在するかどうかをチェックします。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.propertyExists(java.lang.String)

オーバーライド
クラス AQjmsMessage 内の propertyExists(String)

パラメータ
name ― テストするプロパティの名前

戻り値
プロパティが存在している場合は true

例外
JMSException ― 内部 JMS エラーのために、JMS がプロパティの存在のチェックに失敗した場合

setAdtPayload(CustomDatum)

```
public void setAdtPayload(oracle.sql.CustomDatum payload)
```
このユーザー定義型メッセージのデータを含んだ CustomDatum オブジェクトを設定します。

指定方法
インタフェース AdtMessage 内の setAdtPayload(CustomDatum)

パラメータ
payload ― メッセージのデータ（オブジェクトが、CustomDatum インタフェースを実装している必要があります）。このペイロードは、キューまたはトピックのペイロード型として定義されているユーザー定義型を表す Java オブジェクトであることが必要です。

例外
JMSException ― JMS がユーザー定義型ペイロードの設定に失敗した場合

MessageNotWriteableException ― メッセージが読取り専用モードの場合

setBooleanProperty(String, boolean)

```
public void setBooleanProperty(java.lang.String name, boolean value)
```
指定した名前の boolean 型プロパティの値をメッセージに設定します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.setBooleanProperty(java.lang.String, boolean)

オーバーライド
クラス AQjmsMessage 内の setBooleanProperty(String, boolean)

パラメータ
name ― boolean 型プロパティの名前

value ― メッセージに設定する boolean 型プロパティの値

例外
JMSException ― 内部 JMS エラーのために、JMS がプロパティの設定に失敗した場合

MessageNotWriteableException ― プロパティが読取り専用の場合

setByteProperty(String, byte)

```
public void setByteProperty(java.lang.String name, byte value)
```
指定した名前の byte 型プロパティの値をメッセージに設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setByteProperty(java.lang.String, byte)

オーバーライド
クラス AQjmsMessage 内の setByteProperty(String, byte)

パラメータ
name ― byte 型プロパティの名前

value ― メッセージに設定する byte 型プロパティの値

例外
JMSException ― 内部 JMS エラーのために、JMS がプロパティの設定に失敗した場合

MessageNotWriteableException ― プロパティが読取り専用の場合

setDoubleProperty(String, double)

```
public void setDoubleProperty(java.lang.String name, double value)
```
指定した名前の double 型プロパティの値をメッセージに設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setDoubleProperty(java.lang.String, double)

オーバーライド
クラス AQjmsMessage 内の setDoubleProperty(String, double)

パラメータ
`name` ― double 型プロパティの名前

`value` ― メッセージに設定する double 型プロパティの値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がプロパティの設定に失敗した場合

`MessageNotWriteableException` ― プロパティが読取り専用の場合

setFloatProperty(String, float)

```
public void setFloatProperty(java.lang.String name, float value)
```
指定した名前の float 型プロパティの値をメッセージに設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setFloatProperty(java.lang.String, float)

オーバーライド
クラス AQjmsMessage 内の setFloatProperty(String, float)

パラメータ
`name` ― float 型プロパティの名前

`value` ― メッセージに設定する float 型プロパティの値

例外

JMSException ー 内部 JMS エラーのために、JMS がプロパティの設定に失敗した場合

MessageNotWriteableException ー プロパティが読取り専用の場合

setIntProperty(String, int)

```
public void setIntProperty(java.lang.String name, int value)
```
指定した名前の int 型プロパティの値をメッセージに設定します。

指定方法

インタフェース javax.jms.Message 内の javax.jms.Message.setIntProperty(java.lang.String, int)

オーバーライド

クラス AQjmsMessage 内の setIntProperty(String, int)

パラメータ

name ー int 型プロパティの名前

value ー メッセージに設定する int 型プロパティの値

例外

JMSException ー 内部 JMS エラーのために、JMS がプロパティの設定に失敗した場合

MessageNotWriteableException ー プロパティが読取り専用の場合

setJMSReplyTo(Destination)

```
public void setJMSReplyTo(javax.jms.Destination replyTo)
```
このメッセージの返信先を設定します。このメソッドは、このリリースの AdtMessage ではサポートされていません。

指定方法

インタフェース javax.jms.Message 内の javax.jms.Message.setJMSReplyTo(javax.jms.Destination)

オーバーライド

クラス AQjmsMessage 内の setJMSReplyTo(Destination)

例外

JMSException ー AdtMessage ではサポートされていません。

setJMSType(String)

```
public void setJMSType(java.lang.String type)
```
メッセージ・タイプを設定します。このメソッドは、このリリースの AdtMessages ではサポートされていません。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setJMSType(java.lang.String)

オーバーライド
クラス AQjmsMessage 内の setJMSType(String)

パラメータ
`type` ― メッセージのタイプ

例外
`JMSException` ― AdtMessage ではサポートされていません。

setLongProperty(String, long)

```
public void setLongProperty(java.lang.String name, long value)
```
指定した名前の long 型プロパティの値をメッセージに設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setLongProperty(java.lang.String, long)

オーバーライド
クラス AQjmsMessage 内の setLongProperty(String, long)

パラメータ
`name` ― long 型プロパティの名前

`value` ― メッセージに設定する long 型プロパティの値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がプロパティの設定に失敗した場合

`MessageNotWriteableException` ― プロパティが読取り専用の場合

setObjectProperty(String, Object)

```
public void setObjectProperty(java.lang.String name, java.lang.Object value)
```
指定した名前の Java オブジェクト・プロパティの値をメッセージに設定します。

このメソッドは、オブジェクト化された基本オブジェクト型（Integer、Double、Long など）と String 型に対してのみ機能することに注意してください。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setObjectProperty(java.lang.String, java.lang.Object)

オーバーライド
クラス AQjmsMessage 内の setObjectProperty(String, Object)

パラメータ
`name` ― Java オブジェクト・プロパティの名前

`value` ― メッセージに設定する Java オブジェクトのプロパティ値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がプロパティの設定に失敗した場合

`MessageFormatException` ― オブジェクトが無効な場合

`MessageNotWriteableException` ― プロパティが読取り専用の場合

setShortProperty(String, short)

```
public void setShortProperty(java.lang.String name, short value)
```
指定した名前の short 型プロパティの値をメッセージに設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setShortProperty(java.lang.String, short)

オーバーライド
クラス AQjmsMessage 内の setShortProperty(String, short)

パラメータ
`name` ― short 型プロパティの名前

`value` ― メッセージに設定する short 型プロパティの値

例外

`JMSException` － 内部 JMS エラーのために、JMS がプロパティの設定に失敗した場合

`MessageNotWriteableException` － プロパティが読取り専用の場合

setStringProperty(String, String)

```
public void setStringProperty(java.lang.String name, java.lang.String value)
```
指定した名前の String 型プロパティの値をメッセージに設定します。

指定方法

インタフェース javax.jms.Message 内の
javax.jms.Message.setStringProperty(java.lang.String, java.lang.String)

オーバーライド

クラス AQjmsMessage 内の setStringProperty(String, String)

パラメータ

`name` － String 型プロパティの名前

`value` － メッセージに設定する String 型プロパティの値

例外

`JMSException` － 内部 JMS エラーのために、JMS がプロパティの設定に失敗した場合

`MessageNotWriteableException` － プロパティが読取り専用の場合

AQjmsAgent

構文

```
public class AQjmsAgent implements javax.jms.Destination
```

oracle.jms.AQjmsAgent

実装される全インタフェース

javax.jms.Destination

説明

このクラスは、Destination インタフェースを実装します。リモートのサブスクライバおよび返信先を定義するために使用します。

メンバーの概要	説明
コンストラクタ	-
AQjmsAgent(String, String)	コンストラクタ
AQjmsAgent(String, String, int)	コンストラクタ
メソッド	-
getAddress()	エージェントのアドレスを取得します。
getName()	エージェントの名前を取得します。
getProtocol()	エージェントのプロトコルを取得します。
setAddress(String)	エージェントのアドレスを設定します。
setName(String)	エージェントの名前を設定します。
setProtocol(int)	エージェントのプロトコルを設定します。
toString()	エージェントを "[AQjmsAgent] \n name :NAME \n address: ADDRESS \n protocol: PROTOCOL" という形式の文字列表現に変換します。

コンストラクタ

AQjmsAgent(String, String)

```
public  AQjmsAgent(java.lang.String name, java.lang.String address)
```
コンストラクタ

パラメータ

name ― エージェントの名前

address ― エージェントのアドレス

例外

SQLException ― エージェントの作成に失敗した場合

AQjmsAgent(String, String, int)

```
public  AQjmsAgent(java.lang.String name, java.lang.String address, int protocol)
```
コンストラクタ

パラメータ

name ― エージェントの名前

address ― エージェントのアドレス

protocol ― エージェントのプロトコル

例外

SQLException ― エージェントの作成に失敗した場合

メソッド

getAddress()

```
public java.lang.String getAddress()
```
エージェントのアドレスを取得します。

戻り値

エージェントのアドレス

例外

SQLException ― アドレスの取得時にエラーが発生した場合

getName()

```
public java.lang.String getName()
```

エージェントの名前を取得します。

戻り値

エージェントの名前

例外

SQLException ― 名前の取得時にエラーが発生した場合

getProtocol()

```
public int getProtocol()
```

エージェントのプロトコルを取得します。

戻り値

エージェントのプロトコル

例外

SQLException ― プロトコルの取得時にエラーが発生した場合

setAddress(String)

```
public void setAddress(java.lang.String address)
```

エージェントのアドレスを設定します。

パラメータ

address ― エージェントのアドレス

例外

SQLException ― プロトコルの設定時にエラーが発生した場合

setName(String)

```
public void setName(java.lang.String name)
```
エージェントの名前を設定します。

パラメータ
name ― エージェントの名前

例外
SQLException ― 名前の設定時にエラーが発生した場合

setProtocol(int)

```
public void setProtocol(int protocol)
```
エージェントのプロトコルを設定します。

パラメータ
protocol ― エージェントのプロトコル

例外
SQLException ― プロトコルの設定時にエラーが発生した場合

toString()

```
public java.lang.String toString()
```
エージェントを "[AQjmsAgent] \n name :NAME \n address: ADDRESS \n protocol: PROTOCOL" という形式の文字列表現に変換します。

戻り値
エージェントの文字列表現

例外
SQLException ― プロトコルの設定時にエラーが発生した場合

AQjmsBytesMessage

構文

```
public class AQjmsBytesMessage extends AQjmsMessage
   implements javax.jms.BytesMessage

java.lang.Object
  |
  +--AQjmsMessage
        |
        +--oracle.jms.AQjmsBytesMessage
```

実装される全インタフェース

javax.jms.BytesMessage, javax.jms.Message

説明

このクラスは、BytesMessage インタフェースを実装します。BytesMessage は、未解釈のバイト・ストリームを含んだメッセージを送信するために使用します。

メンバーの概要	説明
メソッド	-
clearBody()	メッセージ本体を消去します。
clearProperties()	メッセージのプロパティを消去します。
readBoolean()	ストリーム・メッセージから boolean 型の値を読み込みます。
readByte()	ストリーム・メッセージから 8 ビットの符号付きの値を読み込みます。
readBytes(byte[])	ストリーム・メッセージからバイト配列を読み込みます。
readBytes(byte[], int)	バイト・メッセージの一部を読み込みます。
readChar()	ストリーム・メッセージから Unicode 文字の値を読み込みます。
readDouble()	ストリーム・メッセージから double 型の値を読み込みます。
readFloat()	ストリーム・メッセージから float 型の値を読み込みます。

メンバーの概要	説明
readInt()	ストリーム・メッセージから 32 ビットの符号付き整数を読み込みます。
readLong()	ストリーム・メッセージから 64 ビットの符号付き整数を読み込みます。
readShort()	ストリーム・メッセージから 16 ビットの符号付き整数を読み込みます。
readUnsignedByte()	ストリーム・メッセージから 8 ビットの符号なし数値を読み込みます。
readUnsignedShort()	ストリーム・メッセージから 16 ビットの符号なし数値を読み込みます。
readUTF()	ストリーム・メッセージから、変更済みの UTF-8 形式でエンコーディングされた文字列を読み込みます。
reset()	メッセージを読取り専用モードにして、読み出し位置をストリームの先頭に設定し直します。
writeBoolean(boolean)	boolean 型の値を、1 バイトの値でストリーム・メッセージに書き込みます。
writeByte(byte)	byte 型の値を、1 バイトの値でストリーム・メッセージに書き込みます。
writeBytes(byte[])	ストリーム・メッセージにバイト配列を書き込みます。
writeBytes(byte, int, int)	ストリーム・メッセージにバイト配列の一部を書き込みます。
writeChar(char)	char 型の値を、2 バイトの値で上位バイトから先にストリーム・メッセージに書き込みます。
writeDouble(double)	クラス Double 内の doubleToLongBits メソッドを使用して double 型の引数を long 型に変換し、その long 型の値を、8 バイトの値で上位バイトから先にストリーム・メッセージに書き込みます。
writeFloat(float)	クラス Float 内の floatToIntBits メソッドを使用して、float 型の引数を int 型に変換し、その int 型の値を 4 バイトの値で上位バイトから先にストリーム・メッセージに書き込みます。
writeInt(int)	int 型の値を、4 バイトの値で上位バイトから先にストリーム・メッセージに書き込みます。
writeLong(long)	long 型の値を、8 バイトで上位バイトから先にストリーム・メッセージに書き込みます。

メンバーの概要	説明
writeObject(Object)	ストリーム・メッセージに Java オブジェクトを書き込みます。
writeShort(short)	short 型の値を、2 バイトで上位バイトから先にストリーム・メッセージに書き込みます。
writeUTF(String)	UTF-8 エンコーディングを使用し、マシンに依存しない方法でストリーム・メッセージに文字列を書き込みます。

継承メンバーの概要

インタフェース javax.jms.Message から継承されるフィールド

DEFAULT_DELIVERY_MODE、DEFAULT_PRIORITY、DEFAULT_TIME_TO_LIVE

クラス AQjmsMessage から継承されるメソッド

getBooleanProperty(String)、getByteProperty(String)、
getDoubleProperty(String)、getFloatProperty(String)、
getIntProperty(String)、getJMSCorrelationID()、
getJMSCorrelationIDAsBytes()、getJMSDeliveryMode()、getJMSDestination()、
getJMSExpiration()、getJMSMessageID()、getJMSMessageIDAsBytes()、
getJMSPriority()、getJMSRedelivered()、getJMSReplyTo()、
getJMSTimestamp()、getJMSType()、getLongProperty(String)、
getObjectProperty(String)、getPropertyNames()、getSenderID()、
getShortProperty(String)、getStringProperty(String)、
propertyExists(String)、setBooleanProperty(String, boolean)、
setByteProperty(String, byte)、setDoubleProperty(String, double)、
setFloatProperty(String, float)、setIntProperty(String, int)、
setJMSCorrelationID(String)、setJMSCorrelationIDAsBytes(byte[])、
setJMSDestination(Destination)、setJMSExpiration(long)、
setJMSMessageID(String)、setJMSPriority(int)、
setJMSRedelivered(boolean)、setJMSReplyTo(Destination)、
setJMSTimestamp(long)、setJMSType(String)、setLongProperty(String,
long)、setObjectProperty(String, Object)、setSenderID(AQjmsAgent)、
setShortProperty(String, short)、setStringProperty(String, String)

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、
wait

インタフェース javax.jms.Message から継承されるメソッド

継承メンバーの概要

getBooleanProperty、getByteProperty、getDoubleProperty、
getFloatProperty、getIntProperty、getJMSCorrelationID、
getJMSCorrelationIDAsBytes、getJMSDeliveryMode、getJMSDestination、
getJMSExpiration、getJMSMessageID、getJMSPriority、getJMSRedelivered、
getJMSReplyTo、getJMSTimestamp、getJMSType、getLongProperty、
getObjectProperty、getPropertyNames、getShortProperty、
getStringProperty、propertyExists、setBooleanProperty、setByteProperty、
setDoubleProperty、setFloatProperty、setIntProperty、
setJMSCorrelationID、setJMSCorrelationIDAsBytes、setJMSDeliveryMode、
setJMSDestination、setJMSExpiration、setJMSMessageID、setJMSPriority、
setJMSRedelivered、setJMSReplyTo、setJMSTimestamp、setJMSType、
setLongProperty、setObjectProperty、setShortProperty、setStringProperty

メソッド

clearBody()

```
public void clearBody()
```
メッセージ本体を消去します。メッセージの他の部分はすべてそのまま残ります。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearBody()

オーバーライド
クラス AQjmsMessage 内の clearBody()

例外
`JMSException` ー 内部 JMS エラーのために、JMS がメッセージ本体の消去に失敗した場合

clearProperties()

```
public void clearProperties()
```
メッセージのプロパティを消去します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearProperties()

オーバーライド
クラス AQjmsMessage 内の clearProperties()

例外

JMSException － 内部 JMS エラーのために、JMS が JMS メッセージ・プロパティの消去に失敗した場合

readBoolean()

```
public boolean readBoolean()
```

ストリーム・メッセージから boolean 型の値を読み込みます。

指定方法

インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readBoolean()

戻り値

読み込まれた boolean 型の値

例外

MessageNotReadableException － メッセージが書込み専用モードの場合

JMSException － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

MessageEOFException － メッセージ・ストリームの終わりに達した場合

readByte()

```
public byte readByte()
```

ストリーム・メッセージから 8 ビットの符号付きの値を読み込みます。

指定方法

インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readByte()

戻り値

ストリーム・メッセージから、8 ビットの符号付きの byte 型として次の 1 バイト

例外

MessageNotReadableException － メッセージが書込み専用モードの場合

MessageEOFException － メッセージ・ストリームの終わりに達した場合

JMSException － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

readBytes(byte[])

```
public int readBytes(byte[] value)
```
ストリーム・メッセージからバイト配列を読み込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readBytes(byte[])

パラメータ
`value` ― データが読み込まれるバッファ

戻り値
バッファに読み込まれた合計バイト数、またはストリームの終わりに到達しているためにデータがそれ以上ない場合は -1

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

readBytes(byte[], int)

```
public int readBytes(byte[] value, int length)
```
バイト・メッセージの一部を読み込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readBytes(byte[], int)

パラメータ
`value` ― データが読み込まれるバッファ

`length` ― 読み込むバイト数

戻り値
バッファに読み込まれた合計バイト数、またはストリームの終わりに到達しているためにデータがそれ以上ない場合は -1

例外

MessageNotReadableException － メッセージが書込み専用モードの場合

MessageEOFException － メッセージ・ストリームの終わりに達した場合

JMSException － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

readChar()

```
public char readChar()
```
ストリーム・メッセージから Unicode 文字の値を読み込みます。

指定方法

インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readChar()

戻り値

ストリーム・メッセージから、Unicode 文字として次の 2 バイト

例外

MessageNotReadableException － メッセージが書込み専用モードの場合

MessageEOFException － メッセージ・ストリームの終わりに達した場合

JMSException － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

readDouble()

```
public double readDouble()
```
ストリーム・メッセージから double 型の値を読み込みます。

指定方法

インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readDouble()

戻り値

ストリーム・メッセージから、double 型として解釈された次の 8 バイト

例外

MessageNotReadableException － メッセージが書込み専用モードの場合

MessageEOFException － メッセージ・ストリームの終わりに達した場合

JMSException － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

readFloat()

```
public float readFloat()
```
ストリーム・メッセージから float 型の値を読み込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readFloat()

戻り値
ストリーム・メッセージから、`float` 型として解釈された次の 4 バイト

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

readInt()

```
public int readInt()
```
ストリーム・メッセージから 32 ビットの符号付き整数を読み込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readInt()

戻り値
ストリーム・メッセージから、`int` 型として解釈された次の 4 バイト

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

readLong()

```
public long readLong()
```
ストリーム・メッセージから 64 ビットの符号付き整数を読み込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readLong()

戻り値
ストリーム・メッセージから、long 型として解釈された次の 8 バイト

例外
MessageNotReadableException － メッセージが書込み専用モードの場合

MessageEOFException － メッセージ・ストリームの終わりに達した場合

JMSException － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

readShort()

```
public short readShort()
```
ストリーム・メッセージから 16 ビットの符号付き整数を読み込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readShort()

戻り値
ストリーム・メッセージから、short 型として解釈された次の 2 バイト

例外
MessageNotReadableException － メッセージが書込み専用モードの場合

MessageEOFException － メッセージ・ストリームの終わりに達した場合

JMSException － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

readUnsignedByte()

```
public int readUnsignedByte()
```
ストリーム・メッセージから 8 ビットの符号なし数値を読み込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readUnsignedByte()

戻り値
ストリーム・メッセージから、8 ビットの符号なし数値として解釈された次の 1 バイト

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

readUnsignedShort()

```
public int readUnsignedShort()
```
ストリーム・メッセージから 16 ビットの符号なし数値を読み込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readUnsignedShort()

戻り値
ストリーム・メッセージから、16 ビットの符号なし整数として解釈された次の 2 バイト

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

readUTF()

```
public java.lang.String readUTF()
```
ストリーム・メッセージから、変更済みの UTF-8 形式でエンコーディングされた文字列を読み込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.readUTF()

戻り値
ストリーム・メッセージからの Unicode 文字列

例外
`MessageNotReadableException` － メッセージが書込み専用モードの場合

`MessageEOFException` － メッセージ・ストリームの終わりに達した場合

`JMSException` － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

reset()

```
public void reset()
```
メッセージを読取り専用モードにして、読み出し位置をストリームの先頭に設定し直します。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.reset()

例外
`JMSException` － 内部 JMS エラーのために、JMS がメッセージのリセットに失敗した場合

`MessageFormatException` － メッセージの形式が無効な場合

writeBoolean(boolean)

```
public void writeBoolean(boolean value)
```
boolean 型の値を、1 バイトの値でストリーム・メッセージに書き込みます。値 `true` は値 `(byte)1` として書き込まれ、値 `false` は値 `(byte)0` として書き込まれます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.writeBoolean(boolean)

パラメータ

value ― 書き込まれる boolean 型の値

例外

MessageNotWriteableException ― メッセージが読取り専用モードの場合

JMSException ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

writeByte(byte)

```
public void writeByte(byte value)
```
byte 型の値を、1 バイトの値でストリーム・メッセージに書き込みます。

指定方法

インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.writeByte(byte)

パラメータ

value ― 書き込まれる byte 型の値

例外

MessageNotWriteableException ― メッセージが読取り専用モードの場合

JMSException ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

writeBytes(byte[])

```
public void writeBytes(byte[] value)
```
ストリーム・メッセージにバイト配列を書き込みます。

指定方法

インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.writeBytes(byte[])

パラメータ

value ― 書き込まれるバイト配列

例外

MessageNotWriteableException ― メッセージが読取り専用モードの場合

JMSException ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

writeBytes(byte, int, int)

```
public void writeBytes(byte[] value, int offset, int length)
```
ストリーム・メッセージにバイト配列の一部を書き込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.writeBytes(byte[], int, int)

パラメータ
`value` ― 書き込まれるバイト配列の値

`offset` ― バイト配列内の初期オフセット

`length` ― 使用するバイト数

例外
`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

writeChar(char)

```
public void writeChar(char value)
```
char 型の値を、2 バイトの値で上位バイトから先にストリーム・メッセージに書き込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.writeChar(char)

パラメータ
`value` ― 書き込まれる `char` 型の値

例外
`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

writeDouble(double)

```
public void writeDouble(double value)
```
クラス Double 内の doubleToLongBits メソッドを使用して double 型の引数を long 型に変換し、その long 型の値を、8 バイトの値で上位バイトから先にストリーム・メッセージに書き込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.writeDouble(double)

パラメータ
value ― 書き込まれる double 型の値

例外
MessageNotWriteableException ― メッセージが読取り専用モードの場合

JMSException ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

writeFloat(float)

```
public void writeFloat(float value)
```
クラス Float 内の floatToIntBits メソッドを使用して、float 型の引数を int 型に変換し、その int 型の値を 4 バイトの値で上位バイトから先にストリーム・メッセージに書き込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.writeFloat(float)

パラメータ
value ― 書き込まれる float 型の値

例外
MessageNotWriteableException ― メッセージが読取り専用モードの場合

JMSException ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

writeInt(int)

```
public void writeInt(int value)
```
int 型の値を、4 バイトの値で上位バイトから先にストリーム・メッセージに書き込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.writeInt(int)

パラメータ
value ― 書き込まれる int 型の値

例外
MessageNotWriteableException ― メッセージが読取り専用モードの場合

JMSException ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

writeLong(long)

```
public void writeLong(long value)
```
long 型の値を、8 バイトで上位バイトから先にストリーム・メッセージに書き込みます。

指定方法
インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.writeLong(long)

パラメータ
value ― 書き込まれる long 型の値

例外
MessageNotWriteableException ― メッセージが読取り専用モードの場合

JMSException ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

writeObject(Object)

```
public void writeObject(java.lang.Object value)
```
ストリーム・メッセージに Java オブジェクトを書き込みます。

このメソッドは、オブジェクト化された基本オブジェクト型（Integer、Double、Long など）、String 型およびバイト配列に対してのみ機能することに注意してください。

指定方法

インタフェース javax.jms.BytesMessage 内の
javax.jms.BytesMessage.writeObject(java.lang.Object)

パラメータ

`value` ― 書き込まれる Java オブジェクト

例外

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

`MessageFormatException` ― オブジェクトの型が無効な場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

writeShort(short)

```
public void writeShort(short value)
```

`short` 型の値を、2 バイトで上位バイトから先にストリーム・メッセージに書き込みます。

指定方法

インタフェース javax.jms.BytesMessage 内の javax.jms.BytesMessage.writeShort(short)

パラメータ

`value` ― 書き込まれる `short` 型の値

例外

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

writeUTF(String)

```
public void writeUTF(java.lang.String value)
```

UTF-8 エンコーディングを使用し、マシンに依存しない方法でストリーム・メッセージに文字列を書き込みます。

指定方法

インタフェース javax.jms.BytesMessage 内の
javax.jms.BytesMessage.writeUTF(java.lang.String)

パラメータ

`value` ― 書き込まれる `String` 型の値

例外

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

AQjmsConnection

構文

```
public class AQjmsConnection extends java.lang.Object
   implements javax.jms.QueueConnection, javax.jms.TopicConnection

java.lang.Object
  |
  +--oracle.jms.AQjmsConnection
```

実装される全インタフェース

javax.jms.Connection, javax.jms.QueueConnection, javax.jms.TopicConnection

説明

このクラスは、Connection インタフェースを実装します。これは、JMS プロバイダへのアクティブ接続です。

メンバーの概要	説明
メソッド	-
close()	プロバイダは通常、JVM の外部で大量のリソースを接続のために割り当てている場合があります。このため、クライアントでは、不要なリソースをクローズしておく必要があります。
createQueueSession(boolean, int)	キュー・セッションを作成します。
createTopicSession(boolean, int)	トピック・セッションを作成します。
getClientID()	この接続に対するクライアント識別子を取得します。
getCurrentJmsSession()	現行のセッションを取得します。
getMetaData()	この接続に対するメタデータを取得します。
setClientID(String)	この接続に対するクライアント識別子を設定します。
start()	接続による着信メッセージの配信を開始（または再開始）します。
stop()	接続による着信メッセージの配信を一時的に停止するために使用します。

メンバーの概要	説明
setExceptionListener(ExceptionListener)	この接続に対する例外リスナーを設定します。
getExceptionListener()	この接続に対する例外リスナーを取得します。
setPingPeriod(long)	例外リスナーを ping する間のスリープ周期を設定します。
getPingPeriod()	例外リスナーを ping する間のスリープ周期を取得します。

継承メンバーの概要
クラス java.lang.Object から継承されるメソッド
`clone`、`equals`、`getClass`、`hashCode`、`notify`、`notifyAll`、`toString`、`wait`

メソッド

close()

```
public void close()
```

プロバイダは通常、JVM の外部で大量のリソースを接続のために割り当てている場合があります。このため、クライアントでは、不要なリソースをクローズしておく必要があります。これらのリソースは最終的にガベージ・コレクションで再生されますが、適切な時期に行われない可能性があります。

指定方法

インタフェース javax.jms.Connection 内の javax.jms.Connection.close()

例外

`JMSException` ― 内部エラーのために、JMS 実装が接続のクローズに失敗した場合。たとえば、リソースの解放やソケット接続のクローズに失敗すると、この例外が発生する場合があります。

createQueueSession(boolean, int)

```
public javax.jms.QueueSession createQueueSession(boolean transacted, int ack_mode)
```
キュー・セッションを作成します。

指定方法
インタフェース javax.jms.QueueConnection 内の
javax.jms.QueueConnection.createQueueSession(boolean, int)

パラメータ
`transacted` ― セッションが処理されるかどうかを指定します。

`ack_mode` ― 応答モード

戻り値
QueueSession。QueueSession は、QueueReceiver、QueueSender、QueueBrowser の各内容を作成するためのメソッドを提供します。

例外
`JMSException` ― JMS がキュー・セッションの作成に失敗した場合

createTopicSession(boolean, int)

```
public javax.jms.TopicSession createTopicSession(boolean transacted, int ack_mode)
```
トピック・セッションを作成します。

指定方法
インタフェース javax.jms.TopicConnection 内の
javax.jms.TopicConnection.createTopicSession(boolean, int)

パラメータ
`transacted` ― true の場合、セッションは処理されます。

`ack_mode` ― コンシューマまたはクライアントが、受信したメッセージに応答するかどうかを指定します。このパラメータは、セッションが処理されている場合は無視されます。

戻り値
新しく作成されたトピック・セッション

例外

JMSException ー 内部エラーまたは特定のトランザクションと応答モードをサポートしていないために、JMS Connection がセッションの作成に失敗した場合

getClientID()

```
public java.lang.String getClientID()
```
この接続に対するクライアント識別子を取得します。

指定方法

インタフェース javax.jms.Connection 内の javax.jms.Connection.getClientID()

戻り値

一意クライアント識別子

例外

JMSException ー 内部エラーのために、JMS 実装がこの接続に対するクライアント ID を戻すことができなかった場合

getCurrentJmsSession()

```
public javax.jms.Session getCurrentJmsSession()
```
現行のセッションを取得します。

戻り値

セッション。現行の JMS セッション。

getMetaData()

```
public javax.jms.ConnectionMetaData getMetaData()
```
この接続に対するメタデータを取得します。

指定方法

インタフェース javax.jms.Connection 内の javax.jms.Connection.getMetaData()

戻り値

接続メタデータ

例外

`JMSException` ― JMS 実装が、この接続に関する接続メタデータの取得に失敗した場合の一般的な例外

関連項目

`javax.jms.ConnectionMetaData`

setClientID(String)

```
public void setClientID(java.lang.String clientID)
```

この接続に対するクライアント識別子を設定します。

クライアントのクライアント識別子を割り当てるには、識別子をクライアント固有の ConnectionFactory で構成し、そのクライアントが作成する接続に透過的に割り当てる方法をお薦めします。かわりに、接続のクライアント識別子は、クライアントがプロバイダ固有の値を使用して設定できます。

クライアント識別子の目的は、セッションとそのオブジェクトを、プロバイダがメンテナンスしているクライアントの状態に関連付けることにあります。永続的なサブスクリプションのサポートには、JMS によって識別されるような状態のみが必要です。

指定方法

インタフェース javax.jms.Connection 内の
javax.jms.Connection.setClientID(java.lang.String)

パラメータ

`clientID` ― 一意クライアント識別子

例外

`JMSException` ― 内部エラーのために、JMS 実装がこの接続に対するクライアント ID の設定に失敗した場合の一般的な例外

`InvalidClientIDException` ― JMS クライアントが指定したクライアント ID が無効または重複している場合

start()

```
public void start()
```
接続による着信メッセージの配信を開始（または再開始）します。再度開始すると、最も古い無応答メッセージから始まります。開始済みのセッションの開始は無視されます。

指定方法
インタフェース javax.jms.Connection 内の javax.jms.Connection.start()

例外
`JMSException` ― 内部エラーのために、JMS 実装がメッセージ配信の開始に失敗した場合

関連項目
`javax.jms.Connection.stop()`

stop()

```
public void stop()
```
接続による着信メッセージの配信を一時的に停止するために使用します。`start` メソッドを使用することにより再開できます。停止すると、接続のメッセージ・コンシューマに対する配信はすべて抑制されます。同時期に受信したブロックとメッセージは、メッセージ・リスナーに配信されません。

stop がコールされた後、いくつかのメッセージが配信される場合があります。

セッションを停止しても、メッセージの送信機能に影響はありません。停止済みのセッションの停止は無視されます。

指定方法
インタフェース javax.jms.Connection 内の javax.jms.Connection.stop()

例外
`JMSException` ― 内部エラーのために、JMS 実装がメッセージ配信の停止に失敗した場合

関連項目
`javax.jms.Connection.start()`

setExceptionListener(ExceptionListener)

```
public void setExceptionListener(javax.jms.ExceptionListener listener)
```
この接続に対する例外リスナーを設定します。

JMS のプロバイダによって接続に重大な問題が検出されると、その接続に登録されている ExceptionListener に通知されます。JMS プロバイダは、リスナーの onException() メソッドをコールし、問題について説明する JMSException を渡します。

これにより、クライアントに問題を非同期的に通知できます。メッセージを使用するのみの接続では、障害が起きたことを通知する方法はこれ以外にありません。

接続では、その ExceptionListener の実行がシリアル化されます。

指定方法

インタフェース javax.jms.Connection 内の javax.jms.Connection.setExceptionListener(javax.jms.ExceptionListener listener)

パラメータ

`listener` ― 例外リスナー

例外

`JMSException` ― JMS 実装が、この接続に対する例外リスナーの設定に失敗した場合の一般的な例外

getExceptionListener()

```
public javax.jms.ExceptionListener getExceptionListener()
```
この接続に対する ExceptionListener を取得します。

指定方法

インタフェース javax.jms.Connection 内の javax.jms.Connection.getExceptionListener()

戻り値

登録されている場合は、この接続に対する ExceptionListener。登録されていない場合は、null。

例外

`JMSException` ― JMS 実装が、この接続に対する例外リスナーの取得に失敗した場合の一般的な例外

setPingPeriod(long)

```
public void setPingPeriod(long period)
```
この接続の例外リスナーを ping する間のスリープ周期（ミリ秒単位）を設定します。

例外リスナーが登録されている場合、接続はサーバーを定期的に ping して、サーバーに障害が起きていないことを確認します。この ping により、パフォーマンスが低下する場合があります。適切な ping 周期値を選択するには、バランスを考える必要があります。ping 周期値を大きくすると、非同期クライアントが重大な例外について知らされるまでの時間が長くなります。ping 周期値を小さくすると、ping によるオーバーヘッドが大きくなります。この接続に例外リスナーが登録されていない場合は、ping 間隔を設定する必要はありません。ping 周期のデフォルト値は 2 分です。

パラメータ

`period` − ping を実行するまでの間のスリープ周期（ミリ秒）

getPingPeriod()

```
public long getPingPeriod()
```
この接続の例外リスナーを ping する間のスリープ周期（ミリ秒単位）を取得します。

このメソッドは、直前の `setPingPeriod()` に対するコールによって設定された値、または setPingPeriod がコールされていない場合は、デフォルト値（2 分）を戻します。

戻り値

ping を実行する間のスリープ周期（ミリ秒）。

AQjmsConnectionMetaData

構文

```
public class AQjmsConnectionMetaData extends java.lang.Object
   implements javax.jms.ConnectionMetaData

java.lang.Object
  |
  +--oracle.jms.AQjmsConnectionMetaData
```

実装される全インタフェース

javax.jms.ConnectionMetaData

説明

このクラスは、JMS Connection に使用可能なメタデータ情報を示します。

メンバーの概要	説明
コンストラクタ	-
AQjmsConnectionMetaData()	-
メソッド	-
getJMSMajorVersion()	JMS のバージョン番号を取得します。
getJMSMinorVersion()	JMS のリリース番号を取得します。
getJMSProviderName()	JMS のプロバイダ名を取得します。
getJMSVersion()	JMS のバージョンを取得します。
getProviderMajorVersion()	JMS のプロバイダのバージョン番号を取得します。
getProviderMinorVersion()	JMS のプロバイダのリリース番号を取得します。
getProviderVersion()	JMS のプロバイダのバージョンを取得します。

継承メンバーの概要

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、wait

コンストラクタ

AQjmsConnectionMetaData()

```
public  AQjmsConnectionMetaData()
```

メソッド

getJMSMajorVersion()

```
public int getJMSMajorVersion()
```
JMS のバージョン番号を取得します。

指定方法
インタフェース javax.jms.ConnectionMetaData 内の
javax.jms.ConnectionMetaData.getJMSMajorVersion()

戻り値
JMS のバージョン番号

例外
`JMSException` ― メタデータの取出し時に、JMS の実装で内部エラーが発生した場合

getJMSMinorVersion()

```
public int getJMSMinorVersion()
```
JMS のリリース番号を取得します。

指定方法
インタフェース javax.jms.ConnectionMetaData 内の
javax.jms.ConnectionMetaData.getJMSMinorVersion()

戻り値
JMS のリリース番号

例外
`JMSException` ― メタデータの取出し時に、JMS の実装で内部エラーが発生した場合

getJMSProviderName()

```
public java.lang.String getJMSProviderName()
```
JMS のプロバイダ名を取得します。

指定方法
インタフェース javax.jms.ConnectionMetaData 内の
javax.jms.ConnectionMetaData.getJMSProviderName()

戻り値
JMS のプロバイダ名

例外
`JMSException` ― メタデータの取出し時に、JMS の実装で内部エラーが発生した場合

getJMSVersion()

```
public java.lang.String getJMSVersion()
```
JMS のバージョンを取得します。

指定方法
インタフェース javax.jms.ConnectionMetaData 内の
javax.jms.ConnectionMetaData.getJMSVersion()

戻り値
JMS のバージョン

例外
`JMSException` ― メタデータの取出し時に、JMS の実装で内部エラーが発生した場合

getProviderMajorVersion()

```
public int getProviderMajorVersion()
```
JMS のプロバイダのバージョン番号を取得します。

指定方法
インタフェース javax.jms.ConnectionMetaData 内の
javax.jms.ConnectionMetaData.getProviderMajorVersion()

戻り値
JMS のプロバイダのバージョン番号

例外
`JMSException` ー メタデータの取出し時に、JMS の実装で内部エラーが発生した場合

getProviderMinorVersion()

```
public int getProviderMinorVersion()
```
JMS のプロバイダのリリース番号を取得します。

指定方法
インタフェース javax.jms.ConnectionMetaData 内の
javax.jms.ConnectionMetaData.getProviderMinorVersion()

戻り値
JMS のプロバイダのリリース番号

例外
`JMSException` ー メタデータの取出し時に、JMS の実装で内部エラーが発生した場合

getProviderVersion()

```
public java.lang.String getProviderVersion()
```
JMS のプロバイダのバージョンを取得します。

指定方法
インタフェース javax.jms.ConnectionMetaData 内の
javax.jms.ConnectionMetaData.getProviderVersion()

戻り値
JMS のプロバイダのバージョン

例外
`JMSException` ー メタデータの取出し時に、JMS の実装で内部エラーが発生した場合

AQjmsConstants

構文

```
public class AQjmsConstants
```

oracle.jms.AQjmsConstants

説明

このクラスは、oracle.jms パッケージで使用される定数を定義します。

メンバー
フィールド
EXCEPTION
NONE
NORMAL
STATE_EXPIRED
STATE_PROCESSED
STATE_READY
STATE_WAITING
TRANSACTIONAL
WAIT_FOREVER
WAIT_NONE
コンストラクタ
AQjmsConstants()
メソッド
isJ2eeCompliant()

フィールド

EXCEPTION

```
public static final int EXCEPTION
```

NONE

```
public static final int NONE
```

NORMAL

```
public static final int NORMAL
```

STATE_EXPIRED

```
public static final int STATE_EXPIRED
```

STATE_PROCESSED

```
public static final int STATE_PROCESSED
```

STATE_READY

```
public static final int STATE_READY
```

STATE_WAITING

```
public static final int STATE_WAITING
```

TRANSACTIONAL

```
public static final int TRANSACTIONAL
```

WAIT_FOREVER

```
public static final int WAIT_FOREVER
```

WAIT_NONE

```
public static final int WAIT_NONE
```

コンストラクタ

AQjmsConstants()

```
public  AQjmsConstants()
```

メソッド

isJ2eeCompliant()

JMS クライアントが J2EE/JMS 1.3 準拠モードで実行されている場合は `true` を、それ以外のモードで実行されている場合は `false` を戻します。

クライアントは、実行時に Java プロパティ oracle.jms.j2eeCompliant を true または false のいずれかに設定することによって、OJMS で使用される J2EE 準拠モードを定義できます。j2eeCompliant フラグを false に設定して実行すると、OJMS は、優先順位、期限切れおよび非永続サブスクライバ・セマンティクスについて、以前の（非 J2EE 準拠の）OJMS 動作をサポートします。この結果、以前のクライアントはコードの変更なしに実行できます。

AQjmsConsumer

構文

```
public class AQjmsConsumer extends java.lang.Object
   implements AQjmsQueueReceiver, AQjmsTopicSubscriber, AQjmsTopicReceiver

java.lang.Object
  |
  +--oracle.jms.AQjmsConsumer
```

実装される全インタフェース

AQjmsQueueReceiver, AQjmsTopicReceiver, AQjmsTopicSubscriber, javax.jms.MessageConsumer, javax.jms.QueueReceiver, TopicReceiver, javax.jms.TopicSubscriber

説明

このクラスは、MessageConsumer インタフェースを実装します。

メンバーの概要	説明
メソッド	-
close()	プロバイダは、JVM の外部で MessageConsumer 用に一部のリソースを割り当てている場合があります。このため、クライアントでは、不要なリソースをクローズしておく必要があります。
getMessageListener()	メッセージ・コンシューマの MessageListener を取得します。
getMessageSelector()	メッセージ・コンシューマのメッセージの選択指定式を取得します。
getNavigationMode()	コンシューマのナビゲーション・モードを取得します。
getNoLocal()	この TopicSubscriber の NoLocal 属性を取得します。
getQueue()	このキュー・レシーバに関連付けられているキューを取得します。
getTopic()	このサブスクライバに関連付けられているトピックを取得します。
receive()	このメッセージ・コンシューマ用に生成された次のメッセージを受信します。

メンバーの概要	説明
receive(long)	指定したタイムアウト間隔以内に到着した次のメッセージを受信します。
receiveNoData()	メッセージを、ユーザーに戻さずに使用します。
receiveNoData(long)	メッセージを、ユーザーに戻さずに使用します。
receiveNoWait()	メッセージが即時使用可能な場合に、次のメッセージを受信します。
setMessageListener(MessageListener)	メッセージ・コンシューマの MessageListener を設定します。
setNavigationMode(int)	コンシューマのナビゲーション・モードを設定します。

継承メンバーの概要

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、wait

メソッド

close()

```
public void close()
```

プロバイダは、JVM の外部で MessageConsumer 用に一部のリソースを割り当てている場合があります。このため、クライアントでは、不要なリソースをクローズしておく必要があります。これらのリソースは最終的にガベージ・コレクションで再生されますが、適切な時期に行われない可能性があります。

指定方法

インタフェース javax.jms.MessageConsumer 内の javax.jms.MessageConsumer.close()

例外

JMSException ー エラーのために、JMS がコンシューマのクローズに失敗した場合

getMessageListener()

```
public synchronized javax.jms.MessageListener getMessageListener()
```
メッセージ・コンシューマの MessageListener を取得します。

指定方法
インタフェース javax.jms.MessageConsumer 内の
javax.jms.MessageConsumer.getMessageListener()

戻り値
メッセージ・コンシューマのリスナー、1 セットではない場合は null

例外
`JMSException` − JMS エラーのために、JMS がメッセージ・リスナーの取得に失敗した場合

getMessageSelector()

```
public synchronized java.lang.String getMessageSelector()
```
メッセージ・コンシューマのメッセージの選択指定式を取得します。

指定方法
インタフェース javax.jms.MessageConsumer 内の
javax.jms.MessageConsumer.getMessageSelector()

戻り値
このメッセージ・コンシューマのメッセージ・セレクタ

例外
`JMSException` − JMS エラーのために、JMS がメッセージ・セレクタの取得に失敗した場合

getNavigationMode()

```
public synchronized int getNavigationMode()
```
コンシューマのナビゲーション・モードを取得します。

指定方法
インタフェース AQjmsTopicSubscriber 内の getNavigationMode()

インタフェース AQjmsTopicReceiver 内の getNavigationMode()

戻り値
コンシューマのナビゲーション・モード

例外
`JMSException` ― ナビゲーション・モードが取得できなかった場合

getNoLocal()

```
public synchronized boolean getNoLocal()
```
この TopicSubscriber の NoLocal 属性を取得します。この属性のデフォルト値は false です。

指定方法
インタフェース javax.jms.TopicSubscriber 内の javax.jms.TopicSubscriber.getNoLocal()

戻り値
ローカルに公開されたメッセージが抑制されている場合は true

例外
`JMSException` ― 内部エラーのために、JMS がこのトピック・サブスクライバに対する NoLocal 属性の取得に失敗した場合

getQueue()

```
public synchronized javax.jms.Queue getQueue()
```
このキュー・レシーバに関連付けられているキューを取得します。

指定方法
インタフェース javax.jms.QueueReceiver 内の javax.jms.QueueReceiver.getQueue()

戻り値
レシーバに関連付けられているキュー

例外
JMSException ― 内部エラーのために、JMS がこのキュー・レシーバに対するキューの取得に失敗した場合

getTopic()

```
public synchronized javax.jms.Topic getTopic()
```
このサブスクライバに関連付けられているトピックを取得します。

指定方法
インタフェース javax.jms.TopicSubscriber 内の javax.jms.TopicSubscriber.getTopic()

インタフェース TopicReceiver 内の getTopic()

戻り値
このサブスクライバのトピック

例外
JMSException ― 内部エラーのために、JMS がこのトピック・サブスクライバに対するトピックの取得に失敗した場合

receive()

```
public synchronized javax.jms.Message receive()
```
このメッセージ・コンシューマ用に生成された次のメッセージを受信します。

このコールは、メッセージが生成されるまで無期限にブロックします。

指定方法
インタフェース javax.jms.MessageConsumer 内の javax.jms.MessageConsumer.receive()

戻り値
このメッセージ・コンシューマ用に生成された次のメッセージ

例外
JMSException ― エラーのために、JMS が次のメッセージの受信に失敗した場合

receive(long)

```
public synchronized javax.jms.Message receive(long timeout)
```
指定したタイムアウト間隔以内に到着した次のメッセージを受信します。

このコールは、メッセージが到着するまで、またはタイムアウトになるまでブロックします。

指定方法
インタフェース javax.jms.MessageConsumer 内の javax.jms.MessageConsumer.receive(long)

パラメータ
`timeout` ― タイムアウト値（ミリ秒）

戻り値
このメッセージ・コンシューマ用に生成された次のメッセージ、メッセージが使用可能でない場合は null

例外
`JMSException` ― エラーのために、JMS が次のメッセージの受信に失敗した場合

receiveNoData()

```
public synchronized void receiveNoData()
```
メッセージを、ユーザーに戻さずに使用します。このコールは、データベースからメッセージをフェッチするときのオーバーヘッドを回避します。JMS クライアントが、キュー・ブラウザを使用するなどしてメッセージをすでに取得した場合に、最適化するために使用できます。

指定方法
インタフェース AQjmsQueueReceiver 内の receiveNoData()

例外
`JMSException` ― エラーのためにメッセージの受信ができなかった場合

receiveNoData(long)

```
public synchronized void receiveNoData(long timeout)
```
メッセージを、ユーザーに戻さずに使用します。このコールは、データベースからメッセージをフェッチするときのオーバーヘッドを回避します。JMS クライアントが、キュー・ブラウザを使用するなどしてメッセージをすでに取得した場合に、最適化するために使用できます。このコールは、メッセージが到着するまで、またはタイムアウトになるまでブロックします。

指定方法
インタフェース AQjmsQueueReceiver 内の receiveNoData(long)

パラメータ
`timeout` ― タイムアウト値（ミリ秒）

例外
`JMSException` ― エラーのためにメッセージの受信ができなかった場合

receiveNoWait()

```
public synchronized javax.jms.Message receiveNoWait()
```
メッセージが即時使用可能な場合に、次のメッセージを受信します。

指定方法
インタフェース javax.jms.MessageConsumer 内の
javax.jms.MessageConsumer.receiveNoWait()

戻り値
このメッセージ・コンシューマ用に生成された次のメッセージ、メッセージが使用可能でない場合は null

例外
`JMSException` ― エラーのために、JMS が次のメッセージの受信に失敗した場合

setMessageListener(MessageListener)

```
public synchronized void setMessageListener(javax.jms.MessageListener myListener)
```
メッセージ・コンシューマの MessageListener を設定します。このオブジェクトの onMessage メソッドは、このコンシューマに対するメッセージがある場合にコールされます。

指定方法
インタフェース javax.jms.MessageConsumer 内の javax.jms.MessageConsumer.setMessageListener(javax.jms.MessageListener)

パラメータ
`myListener` ― コンシューマのメッセージ・リスナーを設定します。

例外
`JMSException` ― JMS エラーのために、JMS がメッセージ・リスナーの取得に失敗した場合

setNavigationMode(int)

```
public synchronized void setNavigationMode(int mode)
```
コンシューマのナビゲーション・モードを設定します。

指定方法
インタフェース AQjmsQueueReceiver 内の setNavigationMode(int)

パラメータ
`mode` ― コンシューマのナビゲーション・モード

例外
`JMSException` ― JMS エラーのために、ナビゲーション・モードが設定できなかった場合

AQjmsDestination

構文

```
public class AQjmsDestination extends java.lang.Object
   implements javax.jms.Queue, javax.jms.Topic

java.lang.Object
  |
  +--oracle.jms.AQjmsDestination
```

実装される全インタフェース

```
javax.jms.Destination, javax.jms.Queue, javax.jms.Topic
```

説明

このクラスは、管理オブジェクト、キューおよびトピックを実装します。

メンバーの概要	説明
メソッド	-
alter(Session, AQjmsDestinationProperty)	キューまたはトピックのプロパティを変更します。
alterPropagationSchedule(Session, String, Double, String, Double)	トピックと宛先データベース間の伝播スケジュールを変更します。
delete()	一時宛先を削除し、今後の操作で使用できないようにします。
disablePropagationSchedule(Session, String)	伝播スケジュールを使用禁止にします。
drop(Session)	キューまたはトピックを削除します。
enablePropagationSchedule(Session, String)	伝播スケジュールを使用可能にします。
getCompleteName()	[schema].name の形式で、キューまたはトピックの完全名を取得します。
getCompleteTableName()	[schema].name の形式で、キューまたはトピックのキュー表の完全名を取得します。
getQueueName()	キューの名前を取得します。
getQueueOwner()	キューの所有者を取得します。

メンバーの概要（続き）	説明
getTopicName()	トピックの名前を取得します。
getTopicOwner()	トピックのスキーマを取得します。
grantQueuePrivilege(Session, String, String, boolean)	データベース・ユーザーに対して、キューのエンキュー権限またはデキュー権限を付与します。
grantTopicPrivilege(Session, String, String, boolean)	トピック権限を付与します。
revokeQueuePrivilege(Session, String, String)	キュー権限を取り消します。
revokeTopicPrivilege(Session, String, String)	トピック権限を取り消します。
schedulePropagation(Session, String, Date, Double, String, Double)	指定した宛先データベースのトピックからの伝播をスケジュールします。
start(Session, boolean, boolean)	エンキューまたはデキュー、あるいは両方のためにキューまたはトピックを開始します。
stop(Session, boolean, boolean, boolean)	エンキューまたはデキュー、あるいは両方のためにキューまたはトピックを停止します。
toString()	[schema].name の形式で、キューまたはトピックを文字列として取得します。
unschedulePropagation(Session, String)	トピックと指定した宛先との間の伝播スケジュールを取り消します。

継承メンバーの概要

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、wait

メソッド

alter(Session, AQjmsDestinationProperty)

```
public void alter(javax.jms.Session session, AQjmsDestinationProperty dest_property)
```
キューまたはトピックのプロパティを変更します。

パラメータ

`session` ― JMS セッション

`dest_property` ― キューまたはトピックの新しいプロパティ

alterPropagationSchedule(Session, String, Double, String, Double)

```
public void alterPropagationSchedule(javax.jms.Session session, java.lang.String
destination, java.lang.Double duration, java.lang.String next_time, java.lang.Double
latency)
```
トピックと宛先データベース間の伝播スケジュールを変更します。

パラメータ

`session` ― JMS セッション

`destination` ― 宛先データベースのデータベース・リンク

`duration` ― 継続期間

`next_time` ― 次回の伝播時間

`latency` ― 待ち時間

delete()

```
public void delete()
```
一時宛先を削除し、今後の操作で使用できないようにします。

指定方法

インタフェース javax.jms.TemporaryQueue 内の javax.jms.TemporaryQueue.delete()

およびインタフェース javax.jms.TemporaryTopic 内の javax.jms.TemporaryTopic.delete()

例外

`JMSException` ― 内部エラーのために、OJMS が一時キューまたはトピックの削除に失敗した場合、または一時宛先を使用しているレシーバがまだ存在している場合

disablePropagationSchedule(Session, String)

```
public void disablePropagationSchedule(javax.jms.Session session, java.lang.String
destination)
```
伝播スケジュールを使用禁止にします。

パラメータ

`session` ― JMS セッション

`destination` ― 宛先データベースへのデータベース・リンク

例外

`JMSException` ― 伝播スケジュールを使用禁止にできなかった場合

drop(Session)

```
public void drop(javax.jms.Session session)
```
キューまたはトピックを削除します。

パラメータ

`session` ― JMS セッション

例外

`JMSException` ― キューまたはトピックを削除できなかった場合

enablePropagationSchedule(Session, String)

```
public void enablePropagationSchedule(javax.jms.Session session, java.lang.String
destination)
```
伝播スケジュールを使用可能にします。

パラメータ

`session` ― JMS セッション

`destination` ― 宛先データベースのデータベース・リンク

例外

`JMSException` ― 伝播を使用可能にできなかった場合

getCompleteName()

```
public java.lang.String getCompleteName()
```
[schema].name の形式で、キューまたはトピックの完全名を取得します。

戻り値
キューまたはトピックの完全名

getCompleteTableName()

```
public java.lang.String getCompleteTableName()
```
[schema].name の形式で、キューまたはトピックのキュー表の完全名を取得します。

戻り値
キューまたはトピックのキュー表の完全名

getQueueName()

```
public java.lang.String getQueueName()
```
キューの名前を取得します。

指定方法
インタフェース javax.jms.Queue 内の javax.jms.Queue.getQueueName()

戻り値
キューの名前

例外
`JMSException` ― キューがシングル・コンシューマ・キューではない場合

getQueueOwner()

```
public java.lang.String getQueueOwner()
```
キューの所有者を取得します。

戻り値
キューのスキーマ

例外
`JMSException` ― スキーマを取り出すことができなかった場合

getTopicName()

```
public java.lang.String getTopicName()
```
トピックの名前を取得します。

指定方法
インタフェース javax.jms.Topic 内の javax.jms.Topic.getTopicName()

戻り値
トピックの名前

例外
`JMSException` － キューがマルチ・コンシューマ・キュー（トピック）ではない場合

getTopicOwner()

```
public java.lang.String getTopicOwner()
```
トピックのスキーマを取得します。

戻り値
トピックのスキーマ

例外
`JMSException` － スキーマを取り出すことができなかった場合

grantQueuePrivilege(Session, String, String, boolean)

```
public void grantQueuePrivilege(javax.jms.Session session, java.lang.String
privilege, java.lang.String grantee, boolean grant_option)
```
データベース・ユーザーに対して、キューのエンキュー権限またはデキュー権限を付与します。

パラメータ
`session` － JMS セッション

`privilege` － 権限（ENQUEUE または DEQUEUE）

`grantee` － 権限を付与されるユーザー

`grant_option` － 権限受領者が他のユーザーに権限を付与できるかどうかの指定

例外

JMSException ― 権限を付与できなかった場合

grantTopicPrivilege(Session, String, String, boolean)

```
public void grantTopicPrivilege(javax.jms.Session session, java.lang.String
privilege, java.lang.String grantee, boolean grant_option)
```

トピック権限を付与します。

パラメータ

session ― JMS セッション

privilege ― 付与する権限（ENQUEUE または DEQUEUE）

grantee ― 権限を付与されるデータベース・ユーザー

grant_option ― 権限受領者が他のユーザーに権限を付与できるかどうかの指定

例外

JMSException ― 権限を付与できなかった場合

revokeQueuePrivilege(Session, String, String)

```
public void revokeQueuePrivilege(javax.jms.Session session, java.lang.String
privilege, java.lang.String grantee)
```

キュー権限を取り消します。

パラメータ

session ― JMS セッション

privilege ― 取り消す権限（ENQUEUE または DEQUEUE）

grantee ― 権限を取り消されるデータベース・ユーザー

例外

JMSException ― 権限を取り消すことができなかった場合

revokeTopicPrivilege(Session, String, String)

```
public void revokeTopicPrivilege(javax.jms.Session session, java.lang.String
privilege, java.lang.String grantee)
```
トピック権限を取り消します。

パラメータ

session ― JMS セッション

privilege ― 取り消す権限（ENQUEUE または DEQUEUE）

grantee ― 権限を取り消されるデータベース・ユーザー

例外

JMSException ― 権限を取り消すことができなかった場合

schedulePropagation(Session, String, Date, Double, String, Double)

```
public void schedulePropagation(javax.jms.Session session, java.lang.String
destination, java.util.Date start_time, java.lang.Double duration, java.lang.String
next_time, java.lang.Double latency)
```
指定した宛先データベースのトピックからの伝播をスケジュールします。

パラメータ

session ― JMS セッション

destination ― 伝播のスケジュール先のリモート・データベースのデータベース・リンク。文字列が null の場合は、トピックのデータベース内のすべてのサブスクライバに対して伝播をスケジュールします。

start_time ― 伝播の開始予定時間。

duration ― 伝播の継続期間。

next_time ― 次回の伝播予定時間。

latency ― 遅延が許容される待ち時間（秒）。待ち時間は、メッセージがエンキューされてから伝播されるまでの時間です。

例外

JMSException ― 伝播をスケジュールできなかった場合

start(Session, boolean, boolean)

```
public void start(javax.jms.Session session, boolean enqueue, boolean dequeue)
```
エンキューまたはデキュー、あるいは両方のためにキューまたはトピックを開始します。

パラメータ

`session` ― JMS セッション

`enqueue` ― エンキューを使用可能にするかどうかの指定

`dequeue` ― デキューを使用可能にするかどうかの指定

例外

`JMSException` ― キューまたはトピックの開始に失敗した場合

stop(Session, boolean, boolean, boolean)

```
public void stop(javax.jms.Session session, boolean enqueue, boolean dequeue,
boolean wait)
```
エンキューまたはデキュー、あるいは両方のためにキューまたはトピックを停止します。

パラメータ

`session` ― JMS セッション

`enqueue` ― エンキューを使用禁止にするかどうかの指定

`dequeue` ― デキューを使用禁止にするかどうかの指定

`wait` ― キューまたはトピックにおける保留トランザクションの完了を待機するかどうかの指定

例外

`JMSException` ― キューまたはトピックの停止に失敗した場合

toString()

```
public java.lang.String toString()
```
[schema].name の形式で、キューまたはトピックを文字列として取得します。

指定方法

インタフェース javax.jms.Queue 内の javax.jms.Queue.toString()

オーバーライド

クラス java.lang.Object 内の java.lang.Object.toString()

戻り値

文字列としてのキューまたはトピック

unschedulePropagation(Session, String)

```
public void unschedulePropagation(javax.jms.Session session, java.lang.String
destination)
```

トピックと指定した宛先との間の伝播スケジュールを取り消します。

パラメータ

`session` ― JMS セッション

`destination` ― 伝播スケジュールを取り消す宛先データベースのデータベース・リンク

例外

`JMSException` ― 伝播スケジュールを取り消せなかった場合

AQjmsDestinationProperty

public class AQjmsDestinationProperty
oracle.jms.AQjmsDestinationProperty
このクラスは、宛先のプロパティを定義します。

メンバーの概要	説明
フィールド	-
NORMAL_QUEUE	-
EXCEPTION_QUEUE	-
INFINITE	無限保存
コンストラクタ	-
AQjmsDestinationProperty()	コンストラクタ ー デフォルトの宛先のプロパティでオブジェクトを初期化します。
メソッド	-
getQueueType	このメソッドは、キュー・タイプを取得します。
setQueueType	このメソッドは、キュー・タイプの設定に使用します。
getMaxRetries	このメソッドは、REMOVE モードでのデキューの最大再試行回数を取得します。
setMaxRetries	このメソッドは、REMOVE モードでのデキューの最大再試行回数を設定します。
setRetryInterval	このメソッドは、再試行間隔を設定します。これは、アプリケーションのロールバック後、このメッセージの処理がスケジュールされるまでの時間です。デフォルトは 0（ゼロ）です。
getRetryInterval	このメソッドは、再試行間隔を取得します。
getRetentionTime	このメソッドは、保存時間を取得します。
setRetentionTime	このメソッドは、保存時間を設定します。
getComment	このメソッドは、キューのコメントを取得します。
setComment	このメソッドは、キューのコメントを設定します。

定数

```
public static final int NORMAL_QUEUE
public static final int EXCEPTION_QUEUE
public static final int INFINITE   /* infinite retention */
```

コンストラクタ

AQjmsDestinationProperty()

```
public  AQjmsDestinationProperty()
```
コンストラクタ ー デフォルトの宛先のプロパティでオブジェクトを初期化します。

メソッド

getQueueType

```
public int getQueueType() throws AQException
```
このメソッドは、キュー・タイプを取得します。

戻り値

NORMAL_QUEUE または EXCEPTION_QUEUE

setQueueType

```
public void setQueueType(int q_type) throws AQException
```
このメソッドは、キュー・タイプの設定に使用します。

パラメータ	意味
q_type	NORMAL_QUEUE または EXCEPTION_QUEUE

getMaxRetries

```
public int getMaxRetries() throws AQException
```
このメソッドは、REMOVE モードでのデキューの最大再試行回数を取得します。

setMaxRetries

```
public void setMaxRetries(int retries) throws AQException
public void setMaxRetries(Integer retries) throws AQException
```
このメソッドは、REMOVE モードでのデキューの最大再試行回数を設定します。

パラメータ	意味
retries	REMOVE モードでの最大再試行回数。null を指定すると、デフォルトが使用されます。デフォルトは、シングル・コンシューマ・キューと 8.1 互換のマルチ・コンシューマ・キューに適用されます。Max_retries は、8.0 互換のマルチ・コンシューマ・キューではサポートされていません。

setRetryInterval

```
public void setRetryInterval(double interval) throws AQException
public void setRetryInterval(Double interval) throws AQException
```
このメソッドは、再試行間隔を設定します。これは、アプリケーションのロールバック後、このメッセージの処理がスケジュールされるまでの時間です。デフォルトは 0（ゼロ）です。

パラメータ	意味
interval	再試行間隔。null を指定すると、デフォルトが使用されます。

getRetryInterval

```
public double getRetryInterval() throws AQException
```
このメソッドは、再試行間隔を取得します。

getRetentionTime

```
public double getRetentionTime() throws AQException
```
このメソッドは、保存時間を取得します。

setRetentionTime

```
public void setRetentionTime(double r_time) throws AQException
public void setRetentionTime(Double r_time) throws AQException
```
このメソッドは、保存時間を設定します。

パラメータ	意味
r_time	保存時間。null を指定すると、デフォルトが使用されます。

getComment

```
public java.lang.String getComment() throws AQException
```
このメソッドは、キューのコメントを取得します。

setComment

```
public void setComment(java.lang.String qt_comment) throws AQException
```
このメソッドは、キューのコメントを設定します。

パラメータ	意味
qt_comment	キューのコメント

AQjmsException

構文

```
public class AQjmsException extends javax.jms.JMSException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--javax.jms.JMSException
                    |
                    +--oracle.jms.AQjmsException
```

実装される全インタフェース

```
java.io.Serializable
```

説明

この例外は JMSException を拡張し、Oracle エラー・コードを追加します。これは、すべての JMS 例外の親クラスです。

メンバーの概要	説明
メソッド	-
getErrorNumber()	例外に対応する Oracle エラー・コードを取得します。

継承メンバーの概要

インタフェース javax.jms.JMSException から継承されるメソッド

getErrorCode、getLinkedException、setLinkedException

クラス java.lang.Throwable から継承されるメソッド

fillInStackTrace、getLocalizedMessage、getMessage、printStackTrace、toString

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、wait

メソッド

getErrorNumber()

```
public int getErrorNumber()
```

例外に対応する Oracle エラー・コードを取得します。

AQjmsFactory

構文

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

java.lang.Object
  |
  +--oracle.jms.AQjmsFactory
```

説明

このクラスは、Oracle による JMS の実装で、管理される ConnectionFactory オブジェクトにアクセスするために使用します。

メンバーの概要	説明
メソッド	-
getQueueConnectionFactory(String, Properties)	キュー・コネクション・ファクトリを取得します。
getQueueConnectionFactory(String, String, int, String)	キュー・コネクション・ファクトリを取得します。
getTopicConnectionFactory(String, Properties)	トピック・コネクション・ファクトリを取得します。
getTopicConnectionFactory(String, String, int, String)	トピック・コネクション・ファクトリを取得します。
registerConnectionFactory(java.sql.Connection, String, String, String, int, String, String)	データベースを通じて LDAP にキューまたはトピックのコネクション・ファクトリを登録します。
registerConnectionFactory(java.sql.Connection, String, String, java.util.Properties, String)	データベースを通じて LDAP にキューまたはトピックのコネクション・ファクトリを登録します。
registerConnectionFactory(java.util.Hashtable, String, String, String, int, String, String)	LDAP に対してキューまたはトピックのコネクション・ファクトリを登録します。
registerConnectionFactory(java.util.Hashtable, String, String, java.util.Properties, String)	LDAP にキューまたはトピックのコネクション・ファクトリを登録します。

継承メンバーの概要

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、wait

メソッド

getQueueConnectionFactory(String, Properties)

```
public static javax.jms.QueueConnectionFactory
getQueueConnectionFactory(java.lang.String jdbc_url, java.util.Properties info)
```
キュー・コネクション・ファクトリを取得します。

パラメータ

jdbc_url － 接続する URL

info － プロパティ情報

戻り値

キュー・コネクション・ファクトリ

例外

JMSException － JMS エラーのために、JMS がトピック・コネクション・ファクトリの取得に失敗した場合

getQueueConnectionFactory(String, String, int, String)

```
public static javax.jms.QueueConnectionFactory
getQueueConnectionFactory(java.lang.String hostname, java.lang.String oracle_sid,
int portno, java.lang.String driver)
```
キュー・コネクション・ファクトリを取得します。

パラメータ

hostname － Oracle を実行しているホストの名前

oracle_sid － Oracle システム識別子（Oracle System Identifier: SID）

portno － ポート番号

driver － JDBC ドライバのタイプ（thin または oci8）

戻り値
キュー・コネクション・ファクトリ

例外
JMSException ― JMS エラーのために、JMS がトピック・コネクション・ファクトリの取得に失敗した場合

getTopicConnectionFactory(String, Properties)

```
public static javax.jms.TopicConnectionFactory
getTopicConnectionFactory(java.lang.String jdbc_url, java.util.Properties info)
```
トピック・コネクション・ファクトリを取得します。

パラメータ
jdbc_url ― 接続する URL

info ― プロパティ情報

戻り値
トピック・コネクション・ファクトリ

例外
JMSException ― JMS エラーのために、JMS がトピック・コネクション・ファクトリの取得に失敗した場合

getTopicConnectionFactory(String, String, int, String)

```
public static javax.jms.TopicConnectionFactory
getTopicConnectionFactory(java.lang.String hostname, java.lang.String oracle_sid,
int portno, java.lang.String driver)
```
トピック・コネクション・ファクトリを取得します。

パラメータ
hostname ― Oracle を実行しているホストの名前

oracle_sid ― Oracle システム識別子（Oracle System Identifier: SID）

portno ― ポート番号

driver ― JDBC ドライバのタイプ（thin または oci8）

戻り値

トピック・コネクション・ファクトリ

例外

`JMSException` － JMS エラーのために、JMS がトピック・コネクション・ファクトリの取得に失敗した場合

registerConnectionFactory(java.sql.Connection, String, String, String, int, String, String)

```
public static void registerConnectionFactory(
                   java.sql.Connection connection, String conn_name,
                   String hostname, String oracle_sid, int portno,
                   String driver, String type)
                   throws JMSException
```

Oracle データベースに対応付けられた LDAP にキューまたはトピックのコネクション・ファクトリを登録します。ユーザーはまず Oracle9*i* データベースにログオンすることにより、データベースの LDAP エントリを更新できます。データベースにログオンするには、この操作を実行するための `AQ_ADMINISTRATOR_ROLE` 権限が必要です。

パラメータ

`connection` － 有効なデータベース接続

`conn_name` － 登録するコネクション・ファクトリの名前

`hostname` － コネクション・ファクトリが表すデータベースを実行しているマシンのホスト名

`oracle_sid` － コネクション・ファクトリが表すデータベースの Oracle SID

`portno` － データベースのポート番号

`driver` － データベース（JMS プロバイダ）への接続に使用される JDBC ドライバのタイプ（thin または oci8）

`type` － QueueConnectionFactory を登録する場合は "queue" を指定します。TopicConnectionFactory を登録する場合は "topic" を指定します。

例外

`JMSException` － JMS エラーのために、JMS がコネクション・ファクトリの登録に失敗した場合

registerConnectionFactory(java.sql.Connection, String, String, java.util.Properties, String)

```
public static void registerConnectionFactory(
                   java.sql.Connection connection, String conn_name,
                   String jdbc_url, Properties info, String type)
                   throws JMSException
```

Oracle データベースに対応付けられた LDAP にキューまたはトピックのコネクション・ファクトリを登録します。ユーザーはまず Oracle9*i* データベースにログオンすることにより、データベースの LDAP エントリを更新できます。データベースにログオンするには、この操作を実行するための AQ_ADMINISTRATOR_ROLE 権限が必要です。

パラメータ

connection ― 有効なデータベース接続

conn_name ― 登録するコネクション・ファクトリの名前

jdbc_url ― このコネクション・ファクトリが表すデータベースに接続するための JDBC の URL

info ― JDBC 接続プロパティ

type ― QueueConnectionFactory を登録する場合は "queue" を指定します。TopicConnectionFactory を登録する場合は "topic" を指定します。

例外

JMSException ― JMS エラーのために、JMS がコネクション・ファクトリの登録に失敗した場合

registerConnectionFactory(java.util.Hashtable, String, String, String, int, String, String)

```
public static void registerConnectionFactory(
                   java.util.Hashtable env, String conn_name,
                   String hostname, String oracle_sid, int portno,
                   String driver, String type)
                   throws JMSException
```

LDAP サーバーにキューまたはトピックのコネクション・ファクトリを登録します。このメソッドにより、データベースに接続することなく、LDAP に直接コネクション・ファクトリを登録できます。

LDAP にコネクション・ファクトリを登録するには、GLOBAL_AQ_USER_ROLE 権限が必要です。

パラメータ

env ― 有効な LDAP 環境

conn_name ― 登録するコネクション・ファクトリの名前

hostname ― コネクション・ファクトリが表すデータベースを実行しているマシンのホスト名

oracle_sid ― コネクション・ファクトリが表すデータベースの Oracle SID

portno ― データベースのポート番号

driver ― データベース（JMS プロバイダ）への接続に使用される JDBC ドライバのタイプ（thin または oci8）

type ― QueueConnectionFactory を登録する場合は "queue" を指定します。TopicConnectionFactory を登録する場合は "topic" を指定します。

例外

JMSException ― JMS エラーのために、JMS がコネクション・ファクトリの登録に失敗した場合

registerConnectionFactory(java.util.Hashtable, String, String, java.util.Properties, String)

```
public static void registerConnectionFactory(
                   java.util.Hashtable env, String conn_name,
                   String jdbc_url, Properties info, String type)
                   throws JMSException
```

LDAP サーバーにキューまたはトピックのコネクション・ファクトリを登録します。このメソッドにより、データベースに接続することなく、LDAP に直接コネクション・ファクトリを登録できます。

LDAP にコネクション・ファクトリを登録するには、GLOBAL_AQ_USER_ROLE 権限が必要です。

パラメータ

env ― 有効な LDAP 環境

conn_name ― 登録するコネクション・ファクトリの名前

jdbc_url ― このコネクション・ファクトリが表すデータベースに接続するための JDBC の URL

info ― JDBC 接続プロパティ

type ― QueueConnectionFactory を登録する場合は "queue" を指定します。TopicConnectionFactory を登録する場合は "topic" を指定します。

例外

`JMSException` ― JMS エラーのために、JMS がコネクション・ファクトリの登録に失敗した場合

AQjmsInvalidDestinationException

構文

```
public class AQjmsInvalidDestinationException
   extends javax.jms.InvalidDestinationException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--javax.jms.JMSException
                    |
                    +--javax.jms.InvalidDestinationException
                        |
                      +--oracle.jms.AQjmsInvalidDestinationException
```

実装される全インタフェース

```
java.io.Serializable
```

説明

この例外は、InvalidDestinationException を拡張します。宛先が無効な場合に発生します。

継承メンバーの概要

インタフェース javax.jms.JMSException から継承されるメソッド

`getErrorCode`、`getLinkedException`、`setLinkedException`

クラス java.lang.Throwable から継承されるメソッド

`fillInStackTrace`、`getLocalizedMessage`、`getMessage`、`printStackTrace`、`toString`

クラス java.lang.Object から継承されるメソッド

`clone`、`equals`、`finalize`、`getClass`、`hashCode`、`notify`、`notifyAll`、`wait`

AQjmsInvalidSelectorException

構文

```
public class AQjmsInvalidSelectorException
   extends javax.jms.InvalidSelectorException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--javax.jms.JMSException
                    |
                    +--javax.jms.InvalidSelectorException
                          |
                          +--oracle.jms.AQjmsInvalidSelectorException
```

実装される全インタフェース

```
java.io.Serializable
```

説明

この例外は、InvalidSelectorException を拡張します。指定した MessageSelector が無効な場合に発生します。

継承メンバーの概要

インタフェース javax.jms.JMSException から継承されるメソッド

`getErrorCode`、`getLinkedException`、`setLinkedException`

クラス java.lang.Throwable から継承されるメソッド

`fillInStackTrace`、`getLocalizedMessage`、`getMessage`、`printStackTrace`、`toString`

クラス java.lang.Object から継承されるメソッド

`clone`、`equals`、`finalize`、`getClass`、`hashCode`、`notify`、`notifyAll`、`wait`

AQjmsMapMessage

構文

```
public class AQjmsMapMessage extends AQjmsMessage
   implements javax.jms.MapMessage

java.lang.Object
  |
  +--AQjmsMessage
        |
        +--oracle.jms.AQjmsMapMessage
```

実装される全インタフェース

```
javax.jms.MapMessage, javax.jms.Message
```

説明

このクラスは、MapMessage インタフェースを実装します。MapMessage は、名前 - 値のペアを送信するために使用します。名前は String 型、値は Java プリミティブ型です。

メンバーの概要	説明
メソッド	-
clearBody()	メッセージ本体を消去します。
clearProperties()	メッセージのプロパティを消去します。
getBoolean(String)	指定した名前の boolean 型の値を戻します。
getByte(String)	指定した名前の byte 型の値を戻します。
getBytes(String)	指定した名前のバイト配列の値を戻します。
getChar(String)	指定した名前の Unicode 文字の値を戻します。
getDouble(String)	指定した名前の double 型の値を戻します。
getFloat(String)	指定した名前の float 型の値を戻します。
getInt(String)	指定した名前の int 型の値を戻します。
getLong(String)	指定した名前の long 型の値を戻します。
getMapNames()	すべてのマップ・メッセージ名の一覧を戻します。
getObject(String)	指定した名前の Java オブジェクトの値を戻します。

メンバーの概要（続き）	説明
getShort(String)	指定した名前の short 型の値を戻します。
getString(String)	指定した名前の String 型の値を戻します。
itemExists(String)	任意の項目がこの MapMessage 内に存在しているかどうかをチェックします。
setBoolean(String, boolean)	指定した名前の boolean 型の値をマップに設定します。
setByte(String, byte)	指定した名前の byte 型の値をマップに設定します。
setBytes(String, byte[])	指定した名前のバイト配列の値をマップに設定します。
setBytes(String, byte[], int, int)	指定した名前のバイト配列値の一部をマップに設定します。
setChar(String, char)	指定した名前の Unicode 文字の値をマップに設定します。
setDouble(String, double)	指定した名前の double 型の値をマップに設定します。
setFloat(String, float)	指定した名前の float 型の値をマップに設定します。
setInt(String, int)	指定した名前の int 型の値をマップに設定します。
setLong(String, long)	指定した名前の long 型の値をマップに設定します。
setObject(String, Object)	指定した名前の Java オブジェクトの値をマップに設定します。
setShort(String, short)	指定した名前の short 型の値をマップに設定します。
setString(String, String)	指定した名前の String 型の値をマップに設定します。

継承メンバーの概要

インタフェース javax.jms.Message から継承されるフィールド

DEFAULT_DELIVERY_MODE、DEFAULT_PRIORITY、DEFAULT_TIME_TO_LIVE

クラス AQjmsMessage から継承されるメソッド

継承メンバーの概要（続き）

getBooleanProperty(String)、getByteProperty(String)、
getDoubleProperty(String)、getFloatProperty(String)、
getIntProperty(String)、getJMSCorrelationID()、
getJMSCorrelationIDAsBytes()、getJMSDeliveryMode()、getJMSDestination()、
getJMSExpiration()、getJMSMessageID()、getJMSMessageIDAsBytes()、
getJMSPriority()、getJMSRedelivered()、getJMSReplyTo()、
getJMSTimestamp()、getJMSType()、getLongProperty(String)、
getObjectProperty(String)、getPropertyNames()、getSenderID()、
getShortProperty(String)、getStringProperty(String)、
propertyExists(String)、setBooleanProperty(String, boolean)、
setByteProperty(String, byte)、setDoubleProperty(String, double)、
setFloatProperty(String, float)、setIntProperty(String, int)、
setJMSCorrelationID(String)、setJMSCorrelationIDAsBytes(byte[])、
setJMSDestination(Destination)、setJMSExpiration(long)、
setJMSMessageID(String)、setJMSPriority(int)、
setJMSRedelivered(boolean)、setJMSReplyTo(Destination)、
setJMSTimestamp(long)、setJMSType(String)、setLongProperty(String,
long)、setObjectProperty(String, Object)、setSenderID(AQjmsAgent)、
setShortProperty(String, short)、setStringProperty(String, String)

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、
wait

インタフェース javax.jms.Message から継承されるメソッド

getBooleanProperty、getByteProperty、getDoubleProperty、
getFloatProperty、getIntProperty、getJMSCorrelationID、
getJMSCorrelationIDAsBytes、getJMSDeliveryMode、getJMSDestination、
getJMSExpiration、getJMSMessageID、getJMSPriority、getJMSRedelivered、
getJMSReplyTo、getJMSTimestamp、getJMSType、getLongProperty、
getObjectProperty、getPropertyNames、getShortProperty、
getStringProperty、propertyExists、setBooleanProperty、setByteProperty、
setDoubleProperty、setFloatProperty、setIntProperty、
setJMSCorrelationID、setJMSCorrelationIDAsBytes、setJMSDeliveryMode、
setJMSDestination、setJMSExpiration、setJMSMessageID、setJMSPriority、
setJMSRedelivered、setJMSReplyTo、setJMSTimestamp、setJMSType、
setLongProperty、setObjectProperty、setShortProperty、setStringProperty

メソッド

clearBody()

```
public void clearBody()
```
メッセージ本体を消去します。メッセージの他の部分はすべてそのまま残ります。メッセージは読込みおよび書込みが可能になります。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearBody()

オーバーライド
クラス AQjmsMessage 内の clearBody()

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージ本体の消去に失敗した場合

clearProperties()

```
public void clearProperties()
```
メッセージのプロパティを消去します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearProperties()

オーバーライド
クラス AQjmsMessage 内の clearProperties()

例外
`JMSException` ― 内部 JMS エラーのために、JMS が JMS メッセージ・プロパティの消去に失敗した場合

getBoolean(String)

```
public boolean getBoolean(java.lang.String name)
```
指定した名前の boolean 型の値を戻します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.getBoolean(java.lang.String)

パラメータ
name － boolean 型の名前

戻り値
指定した名前の boolean 型の値

例外
JMSException － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

MessageFormatException － この型変換が無効な場合

getByte(String)

```
public byte getByte(java.lang.String name)
```
指定した名前の byte 型の値を戻します。

指定方法
インタフェース javax.jms.MapMessage 内の javax.jms.MapMessage.getByte(java.lang.String)

パラメータ
name － byte 型の名前

戻り値
指定した名前の byte 型の値

例外
JMSException － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

MessageFormatException － この型変換が無効な場合

getBytes(String)

```
public byte[] getBytes(java.lang.String name)
```
指定した名前のバイト配列の値を戻します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.getBytes(java.lang.String)

パラメータ
name ー バイト配列の名前

戻り値
指定した名前のバイト配列の値。この名前の項目がない場合は、null が戻されます。

例外
JMSException ー 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

MessageFormatException ー この型変換が無効な場合

getChar(String)

```
public char getChar(java.lang.String name)
```
指定した名前の Unicode 文字の値を戻します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.getChar(java.lang.String)

パラメータ
name ー Unicode 文字の名前

戻り値
指定した名前の Unicode 文字の値

例外
JMSException ー 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

MessageFormatException ー この型変換が無効な場合

getDouble(String)

```
public double getDouble(java.lang.String name)
```
指定した名前の double 型の値を戻します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.getDouble(java.lang.String)

パラメータ
name － double 型の名前

戻り値
指定した名前の double 型の値

例外
JMSException － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

MessageFormatException － この型変換が無効な場合

getFloat(String)

```
public float getFloat(java.lang.String name)
```
指定した名前の float 型の値を戻します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.getFloat(java.lang.String)

パラメータ
name － float 型の名前

戻り値
指定した名前の float 型の値

例外
JMSException － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

MessageFormatException － この型変換が無効な場合

getInt(String)

```
public int getInt(java.lang.String name)
```
指定した名前の int 型の値を戻します。

指定方法
インタフェース javax.jms.MapMessage 内の javax.jms.MapMessage.getInt(java.lang.String)

パラメータ
name — int 型の名前

戻り値
指定した名前の int 型の値

例外
JMSException — 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

MessageFormatException — この型変換が無効な場合

getLong(String)

```
public long getLong(java.lang.String name)
```
指定した名前の long 型の値を戻します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.getLong(java.lang.String)

パラメータ
name — long 型の名前

戻り値
指定した名前の long 型の値

例外
JMSException — 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

MessageFormatException — この型変換が無効な場合

getMapNames()

```
public java.util.Enumeration getMapNames()
```
すべてのマップ・メッセージ名の一覧を戻します。

指定方法
インタフェース javax.jms.MapMessage 内の javax.jms.MapMessage.getMapNames()

戻り値
このマップ・メッセージ内にあるすべての名前の一覧

例外
`JMSException` － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

getObject(String)

```
public java.lang.Object getObject(java.lang.String name)
```
指定した名前の Java オブジェクトの値を戻します。

このメソッドは、`setObject` メソッド・コールまたは基本型設定メソッドでマップ内に格納されていたオブジェクトを、オブジェクト化された形式で戻すために使用できることに注意してください。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.getObject(java.lang.String)

パラメータ
`name` － Java オブジェクトの名前

戻り値
指定された名前の Java オブジェクトの値をオブジェクト化された形式で戻します（つまり、int で設定された場合は、Integer が戻されます）。この名前の項目がない場合は、null が戻されます。

例外
`JMSException` － 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

getShort(String)

```
public short getShort(java.lang.String name)
```
指定した名前の short 型の値を戻します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.getShort(java.lang.String)

パラメータ
name ー short 型の名前

戻り値
指定した名前の short 型の値

例外
JMSException ー 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

MessageFormatException ー この型変換が無効な場合

getString(String)

```
public java.lang.String getString(java.lang.String name)
```
指定した名前の String 型の値を戻します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.getString(java.lang.String)

パラメータ
name ー String 型の名前

value ー マップに設定する String 型の値

例外
JMSException ー 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

MessageFormatException ー 型変換が無効な場合

itemExists(String)

```
public boolean itemExists(java.lang.String name)
```
任意の項目がこの MapMessage 内に存在しているかどうかをチェックします。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.itemExists(java.lang.String)

パラメータ
name ー テストする項目の名前

戻り値
項目が存在している場合は true

例外
JMSException － JMS エラーが発生した場合

setBoolean(String, boolean)

```
public void setBoolean(java.lang.String name, boolean value)
```
指定した名前の boolean 型の値をマップに設定します。

指定方法
javax.jms.MapMessage 内の javax.jms.MapMessage.setBoolean(java.lang.String, boolean)

パラメータ
name － boolean 型の名前

value － マップに設定する boolean 型の値

例外
JMSException － 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

MessageNotWriteableException － メッセージが読取り専用モードの場合

setByte(String, byte)

```
public void setByte(java.lang.String name, byte value)
```
指定した名前の byte 型の値をマップに設定します。

指定方法
インタフェース javax.jms.MapMessage 内の javax.jms.MapMessage.setByte(java.lang.String, byte)

パラメータ
name － byte 型の名前

value － マップに設定する byte 型の値

例外
JMSException － 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

MessageNotWriteableException － メッセージが読取り専用モードの場合

setBytes(String, byte[])

```
public void setBytes(java.lang.String name, byte[] value)
```
指定した名前のバイト配列の値をマップに設定します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.setBytes(java.lang.String, byte[])

パラメータ
`name` ― バイト配列の名前

`value` ― マップに設定するバイト配列の値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

setBytes(String, byte[], int, int)

```
public void setBytes(java.lang.String name, byte[] value, int offset, int length)
```
指定した名前のバイト配列値の一部をマップに設定します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.setBytes(java.lang.String, byte[], int, int)

パラメータ
`name` ― バイト配列の名前

`value` ― マップに設定するバイト配列の値

`offset` ― バイト配列内の初期オフセット

`length` ― 使用するバイト数

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

setChar(String, char)

```
public void setChar(java.lang.String name, char value)
```
指定した名前の Unicode 文字の値をマップに設定します。

指定方法
インタフェース javax.jms.MapMessage 内の javax.jms.MapMessage.setChar(java.lang.String, char)

パラメータ
`name` − Unicode 文字の名前

`value` − マップに設定する Unicode 文字の値

例外
`JMSException` − 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

`MessageNotWriteableException` − メッセージが読取り専用モードの場合

setDouble(String, double)

```
public void setDouble(java.lang.String name, double value)
```
指定した名前の double 型の値をマップに設定します。

指定方法
インタフェース javax.jms.MapMessage 内の javax.jms.MapMessage.setDouble(java.lang.String, double)

パラメータ
`name` − double 型の名前

`value` − マップに設定する double 型の値

例外
`JMSException` − 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

`MessageNotWriteableException` − メッセージが読取り専用モードの場合

setFloat(String, float)

```
public void setFloat(java.lang.String name, float value)
```
指定した名前の float 型の値をマップに設定します。

指定方法
インタフェース javax.jms.MapMessage 内の javax.jms.MapMessage.setFloat(java.lang.String, float)

パラメータ
`name` ― float 型の名前

`value` ― マップに設定する float 型の値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

setInt(String, int)

```
public void setInt(java.lang.String name, int value)
```
指定した名前の int 型の値をマップに設定します。

指定方法
インタフェース javax.jms.MapMessage 内の javax.jms.MapMessage.setInt(java.lang.String, int)

パラメータ
`name` ― int 型の名前

`value` ― マップに設定する int 型の値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

setLong(String, long)

```
public void setLong(java.lang.String name, long value)
```
指定した名前の long 型の値をマップに設定します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.setLong(java.lang.String, long)

パラメータ
`name` ― long 型の名前

`value` ― マップに設定する long 型の値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

setObject(String, Object)

```
public void setObject(java.lang.String name, java.lang.Object value)
```
指定した名前の Java オブジェクトの値をマップに設定します。

このメソッドは、オブジェクト化された基本オブジェクト型（Integer、Double、Long など）、String 型およびバイト配列に対してのみ機能することに注意してください。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.setObject(java.lang.String, java.lang.Object)

パラメータ
`name` ― Java オブジェクトの名前

`value` ― マップに設定する Java オブジェクトの値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

`MessageFormatException` ― オブジェクトが無効な場合

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

setShort(String, short)

```
public void setShort(java.lang.String name, short value)
```
指定した名前の short 型の値をマップに設定します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.setShort(java.lang.String, short)

パラメータ
`name` ― short 型の名前

`value` ― マップに設定する short 型の値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

setString(String, String)

```
public void setString(java.lang.String name, java.lang.String value)
```
指定した名前の String 型の値をマップに設定します。

指定方法
インタフェース javax.jms.MapMessage 内の
javax.jms.MapMessage.setString(java.lang.String, java.lang.String)

パラメータ
`name` ― String 型の名前

`value` ― マップに設定する String 型の値

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの書込みに失敗した場合

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

AQjmsMessage

構文

```
public class AQjmsMessage extends java.lang.Object
   implements javax.jms.Message

java.lang.Object
  |
  +--oracle.jms.AQjmsMessage
```

ダイレクト・サブクラス

AQjmsAdtMessage, AQjmsBytesMessage, AQjmsMapMessage, AQjmsObjectMessage, AQjmsStreamMessage, AQjmsTextMessage

実装される全インタフェース

javax.jms.Message

説明

このクラスは、Message インタフェースを実装します。これは、すべての JMS メッセージのスーパークラスです。

メンバーの概要	説明
メソッド	-
acknowledge()	CLIENT_ACKNOWLEDGE モードを使用しているときに、メッセージ受信に応答します。
clearBody()	メッセージ本体を消去します。
clearProperties()	メッセージのプロパティを消去します。
getBooleanProperty(String)	指定した名前の boolean 型プロパティの値を戻します。
getByteProperty(String)	指定した名前の byte 型プロパティの値を戻します。
getDoubleProperty(String)	指定した名前の double 型プロパティの値を戻します。
getFloatProperty(String)	指定した名前の float 型プロパティの値を戻します。
getIntProperty(String)	指定した名前の int 型プロパティの値を戻します。
getJMSCorrelationID()	メッセージの相関 ID を取得します。

メンバーの概要	説明
getJMSCorrelationIDAsBytes()	メッセージの相関 ID をバイトの配列で取得します。
getJMSDeliveryMode()	このメッセージの配信モードを取得します。
getJMSDestination()	このメッセージの宛先を取得します。
getJMSExpiration()	メッセージのタイムアウト値を取得します。
getJMSMessageID()	メッセージ ID を取得します。
getJMSMessageIDAsBytes()	メッセージ ID を取得します。
getJMSPriority()	メッセージの優先順位を取得します。
getJMSRedelivered()	このメッセージが再度配信されているかどうかの識別を取得します。
getJMSReplyTo()	このメッセージの replyTo フィールドを取得します。
getJMSTimestamp()	メッセージのタイムスタンプを取得します。
getJMSType()	メッセージ・タイプを取得します。
getLongProperty(String)	指定した名前の long 型プロパティの値を戻します。
getObjectProperty(String)	指定した名前の Java オブジェクト型プロパティの値を戻します。
getPropertyNames()	すべてのプロパティ名の一覧を戻します。
getSenderID()	メッセージの senderID を取得します。
getShortProperty(String)	指定した名前の short 型プロパティの値を戻します。
getStringProperty(String)	指定した名前の String 型プロパティの値を戻します。
propertyExists(String)	プロパティの値が存在するかどうかをチェックします。
setBooleanProperty(String, boolean)	指定した名前の boolean 型プロパティの値をメッセージに設定します。
setByteProperty(String, byte)	指定した名前の byte 型プロパティの値をメッセージに設定します。
setDoubleProperty(String, double)	指定した名前の double 型プロパティの値をメッセージに設定します。
setFloatProperty(String, float)	指定した名前の float 型プロパティの値をメッセージに設定します。

メンバーの概要	説明
setIntProperty(String, int)	指定した名前の int 型プロパティの値をメッセージに設定します。
setJMSCorrelationID(String)	メッセージの相関 ID を設定します。
setJMSCorrelationIDAsBytes (byte[])	メッセージの相関 ID をバイトの配列で設定します。
setJMSDestination(Destination)	このメッセージの宛先を設定します。
setJMSExpiration(long)	メッセージのタイムアウト値を設定します。このフィールドは、メッセージの送信時にプロバイダが設定します。
setJMSMessageID(String)	メッセージ ID を設定します。
setJMSPriority(int)	このメッセージの優先順位を設定します。
setJMSRedelivered(boolean)	このメッセージが再度配信されているかどうかの識別を設定します。
setJMSReplyTo(Destination)	このメッセージの返信先を設定します。
setJMSTimestamp(long)	メッセージのタイムスタンプを設定します。
setJMSType(String)	メッセージ・タイプを設定します。
setLongProperty(String, long)	指定した名前の long 型プロパティの値をメッセージに設定します。
setObjectProperty(String, Object)	指定した名前の Java オブジェクト・プロパティの値をメッセージに設定します。
setSenderID(AQjmsAgent)	メッセージの senderID を設定します。
setShortProperty(String, short)	指定した名前の short 型プロパティの値をメッセージに設定します。
setStringProperty(String, String)	指定した名前の String 型プロパティの値をメッセージに設定します。

継承メンバーの概要

インタフェース javax.jms.Message から継承されるフィールド

DEFAULT_DELIVERY_MODE、DEFAULT_PRIORITY、DEFAULT_TIME_TO_LIVE

継承メンバーの概要

クラス java.lang.Object から継承されるメソッド

`clone`、`equals`、`finalize`、`getClass`、`hashCode`、`notify`、`notifyAll`、`toString`、`wait`

メソッド

acknowledge()

```
public void acknowledge()
```
CLIENT_ACKNOWLEDGE モードを使用しているときに、メッセージ受信に応答するために使用します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.acknowledge()

例外
`JMSException` － メッセージ応答時に内部例外が発生した場合

clearBody()

```
public void clearBody()
```
メッセージ本体を消去します。メッセージの他の部分はすべてそのまま残ります。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearBody()

例外
`JMSException` － JMS がメッセージの消去に失敗した場合

clearProperties()

```
public void clearProperties()
```
メッセージのプロパティを消去します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearProperties()

例外

JMSException － 内部 JMS エラーのために、JMS が JMS メッセージ・プロパティの消去に失敗した場合

getBooleanProperty(String)

```
public boolean getBooleanProperty(java.lang.String name)
```

指定した名前の boolean 型プロパティの値を戻します。

指定方法

インタフェース javax.jms.Message 内の javax.jms.Message.getBooleanProperty(java.lang.String)

パラメータ

name － boolean 型プロパティの名前

戻り値

指定した名前の boolean 型プロパティの値

例外

JMSException － JMS がプロパティの取得に失敗した場合

MessageFormatException － この型変換が無効な場合

getByteProperty(String)

```
public byte getByteProperty(java.lang.String name)
```

指定した名前の byte 型プロパティの値を戻します。

指定方法

インタフェース javax.jms.Message 内の javax.jms.Message.getByteProperty(java.lang.String)

パラメータ

name － byte 型プロパティの名前

戻り値

指定した名前の byte 型プロパティの値

例外

JMSException － JMS がプロパティの取得に失敗した場合

MessageFormatException － この型変換が無効な場合

getDoubleProperty(String)

```
public double getDoubleProperty(java.lang.String name)
```
指定した名前の double 型プロパティの値を戻します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.getDoubleProperty(java.lang.String)

パラメータ
name － double 型プロパティの名前

戻り値
指定した名前の double 型プロパティの値

例外
JMSException － JMS がプロパティの取得に失敗した場合

MessageFormatException － この型変換が無効な場合

getFloatProperty(String)

```
public float getFloatProperty(java.lang.String name)
```
指定した名前の float 型プロパティの値を戻します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getFloatProperty(java.lang.String)

パラメータ
name － float 型プロパティの名前

戻り値
指定した名前の float 型プロパティの値

例外
JMSException － JMS がプロパティの取得に失敗した場合

MessageFormatException － この型変換が無効な場合

getIntProperty(String)

```
public int getIntProperty(java.lang.String name)
```
指定した名前の int 型プロパティの値を戻します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getIntProperty(java.lang.String)

パラメータ
`name` ― int 型プロパティの名前

戻り値
指定した名前の int 型プロパティの値

例外
`JMSException` ― JMS がプロパティの取得に失敗した場合

`MessageFormatException` ― この型変換が無効な場合

getJMSCorrelationID()

```
public java.lang.String getJMSCorrelationID()
```
メッセージの相関 ID を取得します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSCorrelationID()

戻り値
文字列としてのメッセージの相関 ID

例外
`JMSException` ― 内部 JMS エラーのために、JMS が相関 ID の取得に失敗した場合

getJMSCorrelationIDAsBytes()

```
public byte[] getJMSCorrelationIDAsBytes()
```
メッセージの相関 ID をバイトの配列で取得します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSCorrelationIDAsBytes()

戻り値
バイトの配列としてのメッセージの相関 ID

例外
`JMSException` ― 内部 JMS エラーのために、JMS が相関 ID の取得に失敗した場合

getJMSDeliveryMode()

```
public int getJMSDeliveryMode()
```
このメッセージの配信モードを取得します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSDeliveryMode()

戻り値
このメッセージの配信モード（DeliverMode.PERSISTENT または DeliveryMode.NON_PERSISTENT のいずれか）

例外
`JMSException` ― 内部 JMS エラーのために、JMS が JMS DeliveryMode の取得に失敗した場合

getJMSDestination()

```
public javax.jms.Destination getJMSDestination()
```
このメッセージの宛先を取得します。宛先フィールドには、メッセージの送信先が含まれています。メッセージの送信時には、この値は無視されます。送信メソッドの完了後、送信によって指定された宛先が保持されます。メッセージの受信時には、その宛先の値が、送信時に割り当てられた値と同じであることが必要です。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSDestination()

戻り値
このメッセージの宛先

例外
JMSException ー 内部 JMS エラーのために、JMS が JMS Destination の取得に失敗した場合

getJMSExpiration()

```
public long getJMSExpiration()
```
メッセージのタイムアウト値を取得します。メッセージの送信時には、タイムアウト値は未割当てのままです。送信メソッドの完了後、メッセージのタイムアウト値が保持されます。これは、送信時のグリニッジ標準時（GMT）値とクライアントが指定した Time-To-Live（TTL）値の合計です。Time-To-Live（TTL）に 0（ゼロ）を指定すると、タイムアウト値に 0（ゼロ）が設定され、メッセージはタイムアウトになりません。メッセージがタイムアウトになると、そのメッセージは宛先のキューまたはトピックに対応する例外キューに移されます。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSExpiration()

戻り値
メッセージがタイムアウトになるまでの時間。送信時のグリニッジ標準時（GMT）値とクライアントが指定した Time-To-Live（TTL）値の合計です。

例外
JMSException ー 内部 JMS エラーのために、JMS が JMS のメッセージのタイムアウト取得に失敗した場合

関連項目
javax.jms.Message.setJMSExpiration()

getJMSMessageID()

```
public java.lang.String getJMSMessageID()
```
メッセージ ID を取得します。メッセージ ID ヘッダー・フィールドには、プロバイダが送信した各メッセージを一意に識別する値が含まれています。送信メソッド終了時にプロバイダが割り当てた値が含まれています。JMSMessageID のすべての文字列値は、接頭辞 'ID:' で始まります。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSMessageID()

戻り値
文字列としてのメッセージ ID（接頭辞 'ID:'）

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージ ID の取得に失敗した場合

getJMSMessageIDAsBytes()

```
public byte[] getJMSMessageIDAsBytes()
```
メッセージ ID を取得します。

戻り値
バイト配列としてのメッセージ ID

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージ ID の取得に失敗した場合

getJMSPriority()

```
public int getJMSPriority()
```
メッセージの優先順位を取得します。JMS では、0（ゼロ）～ 9 までの 10 レベルの優先順位値が定義されています。0（ゼロ）が最も低い優先順位、9 が最も高い優先順位です。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSPriority()

戻り値
メッセージに設定されている優先順位

getJMSRedelivered()

```
public boolean getJMSRedelivered()
```
このメッセージが再度配信されているかどうかの識別を取得します。

再配信のインジケータが設定されたメッセージをクライアントが受信した場合、このメッセージはクライアントに以前に配信され、クライアントがトランザクションをコミットしなかった可能性があります。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSRedelivered()

戻り値
このメッセージが再度配信されている場合は true

例外
`JMSException` ― 内部 JMS エラーのために、JMS が JMS Redelivered フラグの取得に失敗した場合

getJMSReplyTo()

```
public javax.jms.Destination getJMSReplyTo()
```
このメッセージの replyTo フィールドを取得します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSReplyTo()

戻り値
replyTo の宛先（形式は AQjmsAgent）

getJMSTimestamp()

```
public long getJMSTimestamp()
```
メッセージのタイムスタンプを取得します。JMSTimestamp ヘッダー・フィールドには、送信用のメッセージがプロバイダに渡された時間が含まれています。メッセージの送信時には、JMSTimestamp は無視されます。送信が完了すると、このメソッドにはメッセージがエンキューされた時間が含まれます。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSTimestamp()

例外
`JMSException` ― JMS がタイムスタンプの取得に失敗した場合

getJMSType()

```
public java.lang.String getJMSType()
```
メッセージ・タイプを取得します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getJMSType()

戻り値
メッセージ・タイプ

例外
JMSException ― 内部 JMS エラーのために、JMS が JMS メッセージ・タイプの取得に失敗した場合

getLongProperty(String)

```
public long getLongProperty(java.lang.String name)
```
指定した名前の long 型プロパティの値を戻します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.getLongProperty(java.lang.String)

パラメータ
name ― long 型プロパティの名前

戻り値
指定した名前の long 型プロパティの値

例外
JMSException ― JMS がプロパティの取得に失敗した場合

MessageFormatException ― この型変換が無効な場合

getObjectProperty(String)

```
public java.lang.Object getObjectProperty(java.lang.String name)
```
指定した名前のプロパティの値を Java オブジェクト型で戻します。このメソッドは、setObject メソッド・コールまたは基本型設定メソッドでプロパティとしてメッセージ内に格納されていたオブジェクトを、オブジェクト化された形式で戻すために使用できることに注意してください。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.getObjectProperty(java.lang.String)

パラメータ
name ― Java オブジェクト・プロパティの名前

戻り値
指定された名前の Java オブジェクト・プロパティの値をオブジェクト化された形式で戻します（つまり、int で設定された場合は、Integer が戻されます）。この名前のプロパティがない場合は、null が戻されます。

例外
`JMSException` － 内部 JMS エラーのために、JMS がプロパティの取得に失敗した場合

getPropertyNames()

```
public synchronized java.util.Enumeration getPropertyNames()
```
すべてのプロパティ名の一覧を戻します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.getPropertyNames()

戻り値
プロパティ値のすべての名前の一覧

例外
`JMSException` － 内部 JMS エラーのために、JMS がプロパティ名の取得に失敗した場合

getSenderID()

```
public AQjmsAgent getSenderID()
```
メッセージの senderID を取得します。この値は、センダーがメッセージの送信前に設定した場合のみ使用可能です。

例外
`JMSException` － JMS が SenderID の取得に失敗した場合

getShortProperty(String)

```
public short getShortProperty(java.lang.String name)
```
指定した名前の short 型プロパティの値を戻します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.getShortProperty(java.lang.String)

パラメータ
name ー short 型プロパティの名前

戻り値
指定した名前の short 型プロパティの値

例外
JMSException ー JMS がプロパティの取得に失敗した場合

MessageFormatException ー この型変換が無効な場合

getStringProperty(String)

```
public java.lang.String getStringProperty(java.lang.String name)
```
指定した名前の String 型プロパティの値を戻します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.getStringProperty(java.lang.String)

パラメータ
name ー String 型プロパティの名前

戻り値
指定した名前の String 型プロパティの値。この名前のプロパティがない場合は、null が戻されます。

例外
JMSException ー JMS がプロパティの取得に失敗した場合

MessageFormatException ー この型変換が無効な場合

propertyExists(String)

```
public boolean propertyExists(java.lang.String name)
```
プロパティの値が存在するかどうかをチェックします。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.propertyExists(java.lang.String)

パラメータ
`name` − テストするプロパティの名前

戻り値
プロパティが存在している場合は true

例外
`JMSException` − 内部 JMS エラーのために、JMS がプロパティの存在のチェックに失敗した場合

setBooleanProperty(String, boolean)

```
public void setBooleanProperty(java.lang.String name, boolean value)
```
指定した名前の boolean 型プロパティの値をメッセージに設定します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.setBooleanProperty(java.lang.String, boolean)

パラメータ
`name` − boolean 型プロパティの名前

`value` − メッセージに設定する boolean 型プロパティの値

例外
`JMSException` − JMS がプロパティの設定に失敗した場合

`MessageNotWriteableException` − プロパティが読取り専用の場合

setByteProperty(String, byte)

```
public void setByteProperty(java.lang.String name, byte value)
```
指定した名前の byte 型プロパティの値をメッセージに設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setByteProperty(java.lang.String, byte)

パラメータ
`name` ― byte 型プロパティの名前

`value` ― メッセージに設定する byte 型プロパティの値

例外
`JMSException` ― JMS がプロパティの設定に失敗した場合

`MessageNotWriteableException` ― プロパティが読取り専用の場合

setDoubleProperty(String, double)

```
public void setDoubleProperty(java.lang.String name, double value)
```
指定した名前の double 型プロパティの値をメッセージに設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setDoubleProperty(java.lang.String, double)

パラメータ
`name` ― double 型プロパティの名前

`value` ― メッセージに設定する double 型プロパティの値

例外
`JMSException` ― JMS がプロパティの設定に失敗した場合

`MessageNotWriteableException` ― プロパティが読取り専用の場合

setFloatProperty(String, float)

```
public void setFloatProperty(java.lang.String name, float value)
```
指定した名前の float 型プロパティの値をメッセージに設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setFloatProperty(java.lang.String, float)

パラメータ
`name` ― float 型プロパティの名前

`value` ― メッセージに設定する float 型プロパティの値

例外
`JMSException` ― JMS がプロパティの設定に失敗した場合

`MessageNotWriteableException` ― プロパティが読取り専用の場合

setIntProperty(String, int)

```
public void setIntProperty(java.lang.String name, int value)
```
指定した名前の int 型プロパティの値をメッセージに設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setIntProperty(java.lang.String, int)

パラメータ
`name` ― int 型プロパティの名前

`value` ― メッセージに設定する int 型プロパティの値

例外
`JMSException` ― JMS がプロパティの設定に失敗した場合

`MessageNotWriteableException` ― プロパティが読取り専用の場合

setJMSCorrelationID(String)

```
public void setJMSCorrelationID(java.lang.String correlationID)
```
メッセージの相関 ID を設定します。クライアントは、JMSCorrelationID ヘッダー・フィールドを使用して、あるメッセージを別のメッセージにリンクできます。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.setJMSCorrelationID(java.lang.String)

パラメータ
`correlationID` ― 参照先のメッセージのメッセージ ID

例外
`JMSException` ― 内部 JMS エラーのために、JMS が相関 ID の設定に失敗した場合

setJMSCorrelationIDAsBytes(byte[])

```
public void setJMSCorrelationIDAsBytes(byte[] correlationID)
```
メッセージの相関 ID をバイトの配列で設定します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.setJMSCorrelationIDAsBytes(byte[])

パラメータ
`correlationID` ― バイトの配列としての相関 ID の値

例外
`JMSException` ― 内部 JMS エラーのために、JMS が相関 ID の設定に失敗した場合

setJMSDeliveryMode()

```
public void setJMSDeliveryMode(int deliveryMode)
```
現行モードに対する配信モードの設定に使用します。配信モードは PERSISTENT または NON_PERSISTENT のいずれかです。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.setJMSDeliveryMode(java.lang.int)

setJMSDestination(Destination)

```
public void setJMSDestination(javax.jms.Destination destination)
```
このメッセージの宛先を設定します。このフィールドは、メッセージの送信時にプロバイダが設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setJMSDestination(javax.jms.Destination)

パラメータ
`destination` ― このメッセージの宛先

例外
`JMSException` ― 内部 JMS エラーのために、JMS が JMS Destination の設定に失敗した場合

setJMSExpiration(long)

```
public void setJMSExpiration(long expiration)
```
メッセージのタイムアウト値を設定します。このフィールドは、メッセージの送信時にプロバイダが設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setJMSExpiration(long)

パラメータ
`expiration` ― メッセージのタイムアウト時刻

例外
`JMSException` ― 内部 JMS エラーのために、JMS が JMS のメッセージのタイムアウト設定に失敗した場合

setJMSMessageID(String)

```
public void setJMSMessageID(java.lang.String id)
```
メッセージ ID を設定します。このフィールドは、メッセージの送信時にプロバイダが設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setJMSMessageID(java.lang.String)

パラメータ
id ー メッセージの ID

例外
JMSException ー 内部 JMS エラーのために、JMS がメッセージ ID の設定に失敗した場合

setJMSPriority(int)

public void setJMSPriority(int priority)
このメッセージの優先順位を設定します。このフィールドは、メッセージの送信時にプロバイダが設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setJMSPriority(int)

パラメータ
priority ー このメッセージの優先順位

例外
JMSException ー 内部 JMS エラーのために、JMS が JMS メッセージ優先順位の設定に失敗した場合

setJMSRedelivered(boolean)

public void setJMSRedelivered(boolean redelivered)
このメッセージが再度配信されているかどうかの識別を設定します。このフィールドは、メッセージの配信時に設定されます。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setJMSRedelivered(boolean)

パラメータ
redelivered ー このメッセージが再度配信されているかどうかの識別

例外
JMSException ー 内部 JMS エラーのために、JMS が JMS Redelivered フラグの設定に失敗した場合

setJMSReplyTo(Destination)

```
public void setJMSReplyTo(javax.jms.Destination replyTo)
```
このメッセージの返信先を設定します。

指定方法
インタフェース javax.jms.Message 内の
javax.jms.Message.setJMSReplyTo(javax.jms.Destination)

パラメータ
`replyTo` ― このメッセージの応答の送信先。宛先は、（consumer_name とキューまたはトピックのアドレスを持つ）AQjmsAgent として指定する必要があります。

例外
`JMSException` ― 内部 JMS エラーのために、JMS が ReplyTo Destination の設定に失敗した場合

setJMSTimestamp(long)

```
public void setJMSTimestamp(long timestamp)
```
メッセージのタイムスタンプを設定します。このフィールドは、メッセージの送信時にプロバイダが設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setJMSTimestamp(long)

パラメータ
`timestamp` ― このメッセージのタイムスタンプ

例外
`JMSException` ― 内部 JMS エラーのために、JMS がタイムスタンプの設定に失敗した場合

setJMSType(String)

```
public void setJMSType(java.lang.String type)
```
メッセージ・タイプを設定します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.setJMSType(java.lang.String)

パラメータ

type ー メッセージのタイプ

例外

JMSException ー 内部 JMS エラーのために、JMS が JMS メッセージ・タイプの設定に失敗した場合

setLongProperty(String, long)

```
public void setLongProperty(java.lang.String name, long value)
```

指定した名前の long 型プロパティの値をメッセージに設定します。

指定方法

インタフェース javax.jms.Message 内の javax.jms.Message.setLongProperty(java.lang.String, long)

パラメータ

name ー long 型プロパティの名前

value ー メッセージに設定する long 型プロパティの値

例外

JMSException ー JMS がプロパティの設定に失敗した場合

MessageNotWriteableException ー プロパティが読取り専用の場合

setObjectProperty(String, Object)

```
public void setObjectProperty(java.lang.String name, java.lang.Object value)
```

指定した名前の Java オブジェクト・プロパティの値をメッセージに設定します。このメソッドは、オブジェクト化された基本オブジェクト型（Integer、Double、Long など）と String 型に対してのみ機能することに注意してください。

指定方法

インタフェース javax.jms.Message 内の javax.jms.Message.setObjectProperty(java.lang.String, java.lang.Object)

パラメータ

name ー Java オブジェクト・プロパティの名前

value ー メッセージに設定する Java オブジェクトのプロパティ値

例外

JMSException － JMS がプロパティの設定に失敗した場合

MessageFormatException － オブジェクトが無効な場合

MessageNotWriteableException － プロパティが読取り専用の場合

setSenderID(AQjmsAgent)

public void setSenderID(AQjmsAgent sender)
メッセージの senderID を設定します。

例外

JMSException － JMS が SenderID の設定に失敗した場合

setShortProperty(String, short)

public void setShortProperty(java.lang.String name, short value)
指定した名前の short 型プロパティの値をメッセージに設定します。

指定方法

インタフェース javax.jms.Message 内の javax.jms.Message.setShortProperty(java.lang.String, short)

パラメータ

name － short 型プロパティの名前

value － メッセージに設定する short 型プロパティの値

例外

JMSException － JMS がプロパティの設定に失敗した場合

MessageNotWriteableException － プロパティが読取り専用の場合

setStringProperty(String, String)

public void setStringProperty(java.lang.String name, java.lang.String value)
指定した名前の String 型プロパティの値をメッセージに設定します。

指定方法

インタフェース javax.jms.Message 内の javax.jms.Message.setStringProperty(java.lang.String, java.lang.String)

パラメータ

`name` ― String 型プロパティの名前

`value` ― メッセージに設定する String 型プロパティの値

例外

`JMSException` ― JMS がプロパティの設定に失敗した場合

`MessageNotWriteableException` ― プロパティが読取り専用の場合

AQjmsMessageEOFException

構文

```
public class AQjmsMessageEOFException
   extends javax.jms.MessageEOFException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--javax.jms.JMSException
                    |
                    +--javax.jms.MessageEOFException
                          |
                          +--oracle.jms.AQjmsMessageEOFException
```

実装される全インタフェース

```
java.io.Serializable
```

説明

この例外は、MessageEOFException を拡張します。StreamMessage または BytesMessage の読込み時に、予期しないストリームの終了に到達した場合に発生します。

継承メンバーの概要

インタフェース javax.jms.JMSException から継承されるメソッド

`getErrorCode`、`getLinkedException`、`setLinkedException`

クラス java.lang.Throwable から継承されるメソッド

`fillInStackTrace`、`getLocalizedMessage`、`getMessage`、`printStackTrace`、`toString`

クラス java.lang.Object から継承されるメソッド

`clone`、`equals`、`finalize`、`getClass`、`hashCode`、`notify`、`notifyAll`、`wait`

AQjmsMessageFormatException

構文

```
public class AQjmsMessageFormatException
   extends javax.jms.MessageFormatException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--javax.jms.JMSException
                    |
                    +--javax.jms.MessageFormatException
                          |
                          +--oracle.jms.AQjmsMessageFormatException
```

実装される全インタフェース

```
java.io.Serializable
```

説明

この例外は、MessageFormatException を拡張します。クライアントが、メッセージでサポートされていないデータ型を使用しようとしたとき、またはメッセージ内のデータを異なる型で読み込もうとしたときに発生します。

継承メンバーの概要

インタフェース javax.jms.JMSException から継承されるメソッド

`getErrorCode`、`getLinkedException`、`setLinkedException`

クラス java.lang.Throwable から継承されるメソッド

`fillInStackTrace`、`getLocalizedMessage`、`getMessage`、`printStackTrace`、`toString`

クラス java.lang.Object から継承されるメソッド

`clone`、`equals`、`finalize`、`getClass`、`hashCode`、`notify`、`notifyAll`、`wait`

AQjmsMessageNotReadableException

構文

```
public class AQjmsMessageNotReadableException
   extends javax.jms.MessageNotReadableException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--javax.jms.JMSException
                    |
                    +--javax.jms.MessageNotReadableException
                       |
                       +--oracle.jms.AQjmsMessageNotReadableException
```

実装される全インタフェース

```
java.io.Serializable
```

説明

この例外は、MessageNotReadableException を拡張します。クライアントが、書込み専用メッセージを読み込もうとしたときに発生します。

継承メンバーの概要

インタフェース javax.jms.JMSException から継承されるメソッド

`getErrorCode`、`getLinkedException`、`setLinkedException`

クラス java.lang.Throwable から継承されるメソッド

`fillInStackTrace`、`getLocalizedMessage`、`getMessage`、`printStackTrace`、`toString`

クラス java.lang.Object から継承されるメソッド

`clone`、`equals`、`finalize`、`getClass`、`hashCode`、`notify`、`notifyAll`、`wait`

AQjmsMessageNotWriteableException

構文

```
public class AQjmsMessageNotWriteableException
   extends javax.jms.MessageNotWriteableException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--javax.jms.JMSException
                    |
                    +--javax.jms.MessageNotWriteableException
                       |
                        +--oracle.jms.AQjmsMessageNotWriteableException
```

実装される全インタフェース

```
java.io.Serializable
```

説明

この例外は、MessageNotWriteableException を拡張します。クライアントが、読取り専用メッセージを書き込もうとしたときに発生します。

継承メンバーの概要

インタフェース javax.jms.JMSException から継承されるメソッド

`getErrorCode`、`getLinkedException`、`setLinkedException`

クラス java.lang.Throwable から継承されるメソッド

`fillInStackTrace`、`getLocalizedMessage`、`getMessage`、`printStackTrace`、`toString`

クラス java.lang.Object から継承されるメソッド

`clone`、`equals`、`finalize`、`getClass`、`hashCode`、`notify`、`notifyAll`、`wait`

AQjmsObjectMessage

構文

```
public class AQjmsObjectMessage extends AQjmsMessage
   implements javax.jms.ObjectMessage

java.lang.Object
  |
  +--AQjmsMessage
        |
        +--oracle.jms.AQjmsObjectMessage
```

実装される全インタフェース

javax.jms.Message, javax.jms.ObjectMessage

説明

このクラスは、ObjectMessage インタフェースを実装します。ObjectMessage は、シリアル化可能な Java オブジェクトを含んだメッセージを送信するために使用します。

メンバーの概要	説明
メソッド	-
clearBody()	メッセージ本体を消去します。
clearProperties()	メッセージのプロパティを消去します。
getObject()	このメッセージのデータを含んだシリアル化可能なオブジェクトを取得します。
setObject(Serializable)	このメッセージのデータを含んだシリアル化可能なオブジェクトを設定します。

継承メンバーの概要

インタフェース javax.jms.Message から継承されるフィールド

DEFAULT_DELIVERY_MODE、DEFAULT_PRIORITY、DEFAULT_TIME_TO_LIVE

クラス AQjmsMessage から継承されるメソッド

継承メンバーの概要

getBooleanProperty(String)、getByteProperty(String)、
getDoubleProperty(String)、getFloatProperty(String)、
getIntProperty(String)、getJMSCorrelationID()、
getJMSCorrelationIDAsBytes()、getJMSDeliveryMode()、getJMSDestination()、
getJMSExpiration()、getJMSMessageID()、getJMSMessageIDAsBytes()、
getJMSPriority()、getJMSRedelivered()、getJMSReplyTo()、
getJMSTimestamp()、getJMSType()、getLongProperty(String)、
getObjectProperty(String)、getPropertyNames()、getSenderID()、
getShortProperty(String)、getStringProperty(String)、
propertyExists(String)、setBooleanProperty(String, boolean)、
setByteProperty(String, byte)、setDoubleProperty(String, double)、
setFloatProperty(String, float)、setIntProperty(String, int)、
setJMSCorrelationID(String)、setJMSCorrelationIDAsBytes(byte[])、
setJMSDestination(Destination)、setJMSExpiration(long)、
setJMSMessageID(String)、setJMSPriority(int)、
setJMSRedelivered(boolean)、setJMSReplyTo(Destination)、
setJMSTimestamp(long)、setJMSType(String)、setLongProperty(String,
long)、setObjectProperty(String, Object)、setSenderID(AQjmsAgent)、
setShortProperty(String, short)、setStringProperty(String, String)

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、
wait

インタフェース javax.jms.Message から継承されるメソッド

getBooleanProperty、getByteProperty、getDoubleProperty、
getFloatProperty、getIntProperty、getJMSCorrelationID、
getJMSCorrelationIDAsBytes、getJMSDeliveryMode、getJMSDestination、
getJMSExpiration、getJMSMessageID、getJMSPriority、getJMSRedelivered、
getJMSReplyTo、getJMSTimestamp、getJMSType、getLongProperty、
getObjectProperty、getPropertyNames、getShortProperty、
getStringProperty、propertyExists、setBooleanProperty、setByteProperty、
setDoubleProperty、setFloatProperty、setIntProperty、
setJMSCorrelationID、setJMSCorrelationIDAsBytes、setJMSDeliveryMode、
setJMSDestination、setJMSExpiration、setJMSMessageID、setJMSPriority、
setJMSRedelivered、setJMSReplyTo、setJMSTimestamp、setJMSType、
setLongProperty、setObjectProperty、setShortProperty、setStringProperty

メソッド

clearBody()

```
public void clearBody()
```
メッセージ本体を消去します。メッセージの他の部分はすべてそのまま残ります。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearBody()

オーバーライド
クラス AQjmsMessage 内の clearBody()

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージ本体の消去に失敗した場合

clearProperties()

```
public void clearProperties()
```
メッセージのプロパティを消去します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearProperties()

オーバーライド
クラス AQjmsMessage 内の clearProperties()

例外
`JMSException` ― 内部 JMS エラーのために、JMS が JMS メッセージ・プロパティの消去に失敗した場合

getObject()

```
public java.io.Serializable getObject()
```
このメッセージのデータを含んだシリアル化可能なオブジェクトを取得します。デフォルト値は null です。

指定方法
インタフェース javax.jms.ObjectMessage 内の javax.jms.ObjectMessage.getObject()

戻り値
このメッセージのデータを含んだシリアル化可能なオブジェクト

例外
`JMSException` ― 内部 JMS エラーのために、JMS がオブジェクトの取得に失敗した場合

`MessageFormatException` ― オブジェクトの非シリアル化に失敗した場合

setObject(Serializable)

```
public void setObject(java.io.Serializable object)
```
このメッセージのデータを含んだシリアル化可能なオブジェクトを設定します。

指定方法
インタフェース javax.jms.ObjectMessage 内の
javax.jms.ObjectMessage.setObject(java.io.Serializable)

パラメータ
`object` ― メッセージのデータ

例外
`JMSException` ― 内部 JMS エラーのために、JMS がオブジェクトの設定に失敗した場合

`MessageFormatException` ― オブジェクトのシリアル化に失敗した場合

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

AQjmsOracleDebug

構文

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

java.lang.Object
  |
  +--oracle.jms.AQjmsOracleDebug
```

説明

AQ Oracle デバッグ・クラス。オラクル社カスタマ・サポート・センターから指示がないかぎり、このクラスは使用しないでください。

メンバーの概要	説明
メソッド	-
getLogStream()	ログ・ストリームを取得します。
setLogStream(OutputStream)	ログ・ストリームを設定します。
setTraceLevel(int)	トレース・レベルを設定します。0 ― トレースなし(デフォルト)、1 ― 致命的なエラー、2 ― その他のエラー /imp メッセージ、3 ― 例外トレース、その他のトレース情報、4 ― メソッドの開始 / 終了、5 ― スタック・トレース / 変数情報の出力

継承メンバーの概要

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、wait

メソッド

getLogStream()

```
public static java.io.OutputStream getLogStream()
```
ログ・ストリームを取得します。

setLogStream(OutputStream)

```
public static void setLogStream(java.io.OutputStream output_stream)
```
ログ・ストリームを設定します。

パラメータ

`output_stream` ― ログ・ストリーム

setTraceLevel(int)

```
public static void setTraceLevel(int level)
```
トレース・レベルを設定します。0 ― トレースなし（デフォルト）、1 ― 致命的なエラー、2 ― その他のエラー /imp メッセージ、3 ― 例外トレース、その他のトレース情報、4 ― メソッドの開始 / 終了、5 ― スタック・トレース / 変数情報の出力

AQjmsProducer

構文

```
public class AQjmsProducer extends java.lang.Object
   implements AQjmsQueueSender, AQjmsTopicPublisher

java.lang.Object
  |
  +--oracle.jms.AQjmsProducer
```

実装される全インタフェース

AQjmsQueueSender, AQjmsTopicPublisher, javax.jms.MessageProducer, javax.jms.QueueSender, javax.jms.TopicPublisher

説明

このクラスは、MessageProducer インタフェースを実装します。MessageProducer は、宛先にメッセージを送信するために使用します。

メンバーの概要	説明
メソッド	-
close()	プロバイダは、JVM の外部で MessageProducer 用に一部のリソースを割り当てている場合があります。このため、クライアントでは、不要なリソースをクローズしておく必要があります。
getDeliveryMode()	プロデューサのデフォルトの配信モードを取得します。
getDisableMessageID()	メッセージ ID が使用禁止にされているかどうかの識別を取得します。
getDisableMessageTimestamp()	メッセージのタイムスタンプが使用禁止にされているかどうかの識別を取得します。
getPriority()	プロデューサのデフォルトの優先順位を取得します。
getQueue()	このキュー・センダーに関連付けられているキューを取得します。
getTimeToLive()	発送時以降、メッセージ・システムが生成したメッセージを保持している時間のデフォルト長（ミリ秒）を取得します。

メンバーの概要（続き）	説明
getTopic()	このパブリッシャに関連付けられているトピックを取得します。
publish(Message)	トピックにメッセージを公開します。
publish(Message, AQjmsAgent[])	特定のレシピエントのリストにメッセージを公開します。
publish(Message, AQjmsAgent[], int, int, long)	レシピエントのリスト、配信モード、優先順位および TTL を指定して、トピックにメッセージを公開します。
publish(Message, int, int, long)	配信モード、優先順位および TTL を指定して、トピックにメッセージを公開します。
publish(Topic, Message)	メッセージ・プロデューサが未指定のトピックにメッセージを公開します。
publish(Topic, Message, AQjmsAgent[])	レシピエントのリストを指定してトピックにメッセージを公開します。
publish(Topic, Message, AQjmsAgent[], int, int, long)	レシピエントのリスト、配信モード、優先順位および TTL を指定して、トピックにメッセージを公開します。
publish(Topic, Message, int, int, long)	配信モード、優先順位および TTL を指定して、メッセージ・プロデューサが未指定のトピックにメッセージを公開します。
send(Message)	メッセージを送信します。
send(Message, int, int, long)	メッセージを送信します。
send(Queue, Message)	メッセージを送信します。
send(Queue, Message, int, int, long)	メッセージを送信します。
setDeliveryMode(int)	プロデューサのデフォルトの配信モードを設定します。
setDisableMessageID(boolean)	メッセージ ID を使用禁止にするかどうかを設定します。
setDisableMessageTimestamp(boolean)	メッセージのタイムスタンプを使用禁止にするかどうかを設定します。
setPriority(int)	プロデューサのデフォルトの優先順位を設定します。
setTimeToLive(long)	発送時以降、メッセージ・システムが生成したメッセージを保持している時間のデフォルト長（ミリ秒）を設定します。

継承メンバーの概要

クラス java.lang.Object から継承されるメソッド

`clone`、`equals`、`finalize`、`getClass`、`hashCode`、`notify`、`notifyAll`、`toString`、`wait`

メソッド

close()

```
public void close()
```
プロバイダは、JVM の外部で MessageProducer 用に一部のリソースを割り当てている場合があります。このため、クライアントでは、不要なリソースをクローズしておく必要があります。これらのリソースは最終的にガベージ・コレクションで再生されますが、適切な時期に行われない可能性があります。

指定方法
インタフェース javax.jms.MessageProducer 内の javax.jms.MessageProducer.close()

例外
`JMSException` ― エラーのために、JMS がプロデューサのクローズに失敗した場合

getDeliveryMode()

```
public synchronized int getDeliveryMode()
```
プロデューサのデフォルトの配信モードを取得します。

指定方法
インタフェース javax.jms.MessageProducer 内の
javax.jms.MessageProducer.getDeliveryMode()

戻り値
このメッセージ・プロデューサのメッセージ配信モード

例外
`JMSException` ― 内部エラーのために、JMS が配信モードの取得に失敗した場合

getDisableMessageID()

```
public synchronized boolean getDisableMessageID()
```
メッセージ ID が使用禁止にされているかどうかの識別を取得します。

指定方法
インタフェース javax.jms.MessageProducer 内の
javax.jms.MessageProducer.getDisableMessageID()

戻り値
メッセージのタイムスタンプが使用禁止にされているかどうかの識別

例外
`JMSException` ― 内部エラーのために、JMS が使用禁止にされているメッセージ ID の取得に失敗した場合

getDisableMessageTimestamp()

```
public synchronized boolean getDisableMessageTimestamp()
```
メッセージのタイムスタンプが使用禁止にされているかどうかの識別を取得します。

指定方法
インタフェース javax.jms.MessageProducer 内の
javax.jms.MessageProducer.getDisableMessageTimestamp()

戻り値
メッセージのタイムスタンプが使用禁止にされているかどうかの識別

例外
`JMSException` ― 内部エラーのために、JMS が使用禁止にされているメッセージ・タイムスタンプの取得に失敗した場合

getPriority()

```
public synchronized int getPriority()
```
プロデューサのデフォルトの優先順位を取得します。

指定方法
インタフェース javax.jms.MessageProducer 内の javax.jms.MessageProducer.getPriority()

戻り値
このメッセージ・プロデューサのメッセージ優先順位

例外
`JMSException` ― 内部エラーのために、JMS が優先順位の取得に失敗した場合

getQueue()

```
public synchronized javax.jms.Queue getQueue()
```
このキュー・センダーに関連付けられているキューを取得します。

指定方法
インタフェース javax.jms.QueueSender 内の javax.jms.QueueSender.getQueue()

戻り値
キュー

例外
`JMSException` ― 内部エラーのために、JMS がこのキュー・センダー用のキューの取得に失敗した場合

getTimeToLive()

```
public synchronized long getTimeToLive()
```
発送時以降、メッセージ・システムが生成したメッセージを保持している時間のデフォルト長（ミリ秒）を取得します。

指定方法
インタフェース javax.jms.MessageProducer 内の
javax.jms.MessageProducer.getTimeToLive()

戻り値
メッセージの TTL（ミリ秒）。0（ゼロ）は無制限です。

例外
`JMSException` ― 内部エラーのために、JMS が TTL の取得に失敗した場合

getTopic()

```
public synchronized javax.jms.Topic getTopic()
```
このパブリッシャに関連付けられているトピックを取得します。

指定方法
インタフェース javax.jms.TopicSubscriber 内の javax.jms.TopicSubscriber.getTopic()

戻り値
このパブリッシャのトピック

例外
JMSException ― 内部エラーのために、JMS がこのトピック・パブリッシャ用のトピックの取得に失敗した場合

publish(Message)

```
public synchronized void publish(javax.jms.Message message)
```
トピックにメッセージを公開します。

指定方法
インタフェース javax.jms.TopicPublisher 内の javax.jms.TopicPublisher.publish(javax.jms.Message)

パラメータ
message ― 公開するメッセージ

例外
JMSException ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

publish(Message, AQjmsAgent[])

```
public synchronized void publish(javax.jms.Message message, AQjmsAgent recipient_list)
```
特定のレシピエントのリストにメッセージを公開します。

指定方法
インタフェース AQjmsTopicPublisher 内の publish(Message, AQjmsAgent[])

パラメータ

message ― 公開するメッセージ

recipient_list ― メッセージの公開先のレシピエントのリスト。レシピエントのタイプは AQjmsAgent です。

例外

JMSException ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

publish(Message, AQjmsAgent[], int, int, long)

public synchronized void publish(javax.jms.Message message, AQjmsAgent recipient_list, int deliveryMode, int priority, long timeToLive)
レシピエントのリスト、配信モード、優先順位および TTL を指定して、トピックにメッセージを公開します。

指定方法

インタフェース AQjmsTopicPublisher 内の publish(Message, AQjmsAgent[], int, int, long)

パラメータ

message ― 公開するメッセージ

recipient_list ― メッセージの公開先のレシピエントのリスト。レシピエントのタイプは AQjmsAgent です。

deliveryMode ― 配信モード。persistent または non_persistent。

priority ― メッセージの優先順位

timeToLive ― メッセージの TTL（ミリ秒）。0（ゼロ）は無制限です。

例外

JMSException ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

publish(Message, int, int, long)

public synchronized void publish(javax.jms.Message message, int deliveryMode, int priority, long timeToLive)
配信モード、優先順位および TTL を指定して、トピックにメッセージを公開します。

指定方法

インタフェース javax.jms.TopicPublisher 内の
javax.jms.TopicPublisher.publish(javax.jms.Message, int, int, long)

パラメータ

message ― 公開するメッセージ

deliveryMode ― メッセージ配信モード。persistent または non_persistent。

priority ― メッセージの優先順位

timeToLive ― メッセージの TTL（ミリ秒）。0（ゼロ）は無制限です。

例外

JMSException ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

publish(Topic, Message)

```
public synchronized void publish(javax.jms.Topic topic, javax.jms.Message message)
```

メッセージ・プロデューサが未指定のトピックにメッセージを公開します。プロデューサのデフォルトの配信モード、TTL および優先順位を使用します。

指定方法

インタフェース javax.jms.TopicPublisher 内の
javax.jms.TopicPublisher.publish(javax.jms.Topic, javax.jms.Message)

パラメータ

topic ― メッセージを公開するトピック。この指定は、メッセージ・プロデューサのデフォルトのトピックをオーバーライドします。

message ― 公開するメッセージ

例外

JMSException ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

publish(Topic, Message, AQjmsAgent[])

```
public synchronized void publish(javax.jms.Topic topic, javax.jms.Message message,
AQjmsAgent recipient_list)
```

レシピエントのリストを指定してトピックにメッセージを公開します。

指定方法

インタフェース AQjmsTopicPublisher 内の publish(Topic, Message, AQjmsAgent[])

パラメータ

topic ― メッセージを公開するトピック。この指定は、メッセージ・プロデューサのデフォルトのトピックをオーバーライドします。

message ― 公開するメッセージ

recipient_list ― メッセージの公開先のレシピエントのリスト。レシピエントのタイプは AQjmsAgent です。

例外

JMSException ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

publish(Topic, Message, AQjmsAgent[], int, int, long)

```
public synchronized void publish(javax.jms.Topic topic, javax.jms.Message message,
AQjmsAgent recipient_list, int deliveryMode, int priority, long timeToLive)
```

レシピエントのリスト、配信モード、優先順位および TTL を指定して、トピックにメッセージを公開します。

指定方法

インタフェース AQjmsTopicPublisher 内の publish(Topic, Message, AQjmsAgent[], int, int, long)

パラメータ

topic ― メッセージを公開するトピック。この指定は、メッセージ・プロデューサのデフォルトのトピックをオーバーライドします。

message ― 公開するメッセージ

recipient_list ― メッセージの公開先のレシピエントのリスト。レシピエントのタイプは AQjmsAgent です。

deliveryMode ― 配信モード。persistent または non_persistent。

priority ― メッセージの優先順位

timeToLive ― メッセージの TTL（ミリ秒）。0（ゼロ）は無制限です。

例外

JMSException ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

publish(Topic, Message, int, int, long)

```
public synchronized void publish(javax.jms.Topic topic, javax.jms.Message message,
int deliveryMode, int priority, long timeToLive)
```

配信モード、優先順位および TTL を指定して、メッセージ・プロデューサが未指定のトピックにメッセージを公開します。

指定方法

インタフェース javax.jms.TopicPublisher 内の
javax.jms.TopicPublisher.publish(javax.jms.Topic, javax.jms.Message, int, int, long)

パラメータ

`topic` ― メッセージを公開するトピック。この指定は、メッセージ・プロデューサのデフォルトのトピックをオーバーライドします。

`message` ― 公開するメッセージ

`deliveryMode` ― メッセージ配信モード。persistent または non_persistent。

`priority` ― メッセージの優先順位

`timeToLive` ― メッセージの TTL（ミリ秒）。0（ゼロ）は無制限です。

例外

`JMSException` ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

send(Message)

```
public synchronized void send(javax.jms.Message message)
```

メッセージを送信します。

指定方法

インタフェース javax.jms.QueueSender 内の
javax.jms.QueueSender.send(javax.jms.Message)

パラメータ

`message` ― 送信するメッセージ

例外

`JMSException` ― 内部エラーのために、JMS がメッセージの送信に失敗した場合

send(Message, int, int, long)

```
public synchronized void send(javax.jms.Message message, int deliveryMode,
int priority, long timeToLive)
```
メッセージを送信します。

指定方法
インタフェース javax.jms.QueueSender 内の
javax.jms.QueueSender.send(javax.jms.Message, int, int, long)

パラメータ
`message` ― 送信するメッセージ

`deliverMode` ― メッセージ配信モード。persistent または non_persistent。

例外
`JMSException` ― 内部エラーのために、JMS がメッセージの送信に失敗した場合

send(Queue, Message)

```
public synchronized void send(javax.jms.Queue queue, javax.jms.Message message)
```
メッセージを送信します。

指定方法
インタフェース javax.jms.QueueSender 内の javax.jms.QueueSender.send(javax.jms.Queue, javax.jms.Message)

パラメータ
`queue` ― メッセージの送信先の宛先キュー。この指定は、メッセージ・プロデューサのデフォルトのキューをオーバーライドします。

`message` ― 送信するメッセージ

例外
`JMSException` ― 内部エラーのために、JMS がメッセージの送信に失敗した場合

send(Queue, Message, int, int, long)

```
public synchronized void send(javax.jms.Queue queue, javax.jms.Message message,
int deliveryMode, int priority, long timeToLive)
```
メッセージを送信します。

指定方法
インタフェース javax.jms.QueueSender 内の javax.jms.QueueSender.send(javax.jms.Queue, javax.jms.Message, int, int, long)

パラメータ
`queue` ― メッセージの送信先の宛先キュー。この指定は、メッセージ・プロデューサのデフォルトのキューをオーバーライドします。

`message` ― 送信するメッセージ

`deliveryMode` ― メッセージ配信モード。persistent または non_persistent。

`priority` ― メッセージの優先順位

`timeToLive` ― メッセージの TTL（ミリ秒）。0（ゼロ）は無制限です。

例外
`JMSException` ― 内部エラーのために、JMS がメッセージの送信に失敗した場合

setDeliveryMode(int)

```
public synchronized void setDeliveryMode(int deliveryMode)
```
プロデューサのデフォルトの配信モードを設定します。

配信モードは、デフォルトで PERSISTENT に設定されます。

指定方法
インタフェース javax.jms.MessageProducer 内の javax.jms.MessageProducer.setDeliveryMode(int)

パラメータ
`deliveryMode` ― このメッセージ・プロデューサのメッセージ配信モード

例外
`JMSException` ― 内部エラーのために、JMS が配信モードの設定に失敗した場合

setDisableMessageID(boolean)

```
public synchronized void setDisableMessageID(boolean value)
```
メッセージ ID を使用禁止にするかどうかを設定します。

メッセージ ID の作成には手間がかかり、メッセージのサイズも増えます。アプリケーションではメッセージ ID が使用されないというヒントを与えると、JMS プロバイダによっては、メッセージのオーバーヘッドを最適化できる場合があります。JMS メッセージ・プロデューサは、メッセージ ID を使用禁止にするヒントを提供します。クライアントが、メッセージ ID を使用禁止にするようにプロデューサを設定すると、プロデューサは、生成するメッセージに対してメッセージ ID の値に依存しないことを宣言します。これらのメッセージ ID は null に設定するか、ヒントを無視する場合は通常の一意の値に設定する必要があります。

メッセージ ID は、デフォルトでは使用可能です。

指定方法
インタフェース javax.jms.MessageProducer 内の
javax.jms.MessageProducer.setDisableMessageID(boolean)

パラメータ
`value` ― メッセージ ID を使用禁止にするかどうかの指定

例外
`JMSException` ― 内部エラーのために、JMS が使用禁止メッセージ ID の設定に失敗した場合

setDisableMessageTimestamp(boolean)

```
public synchronized void setDisableMessageTimestamp(boolean value)
```
メッセージのタイムスタンプを使用禁止にするかどうかを設定します。

指定方法
インタフェース javax.jms.MessageProducer 内の
javax.jms.MessageProducer.setDisableMessageTimestamp(boolean)

パラメータ
`value` ― メッセージのタイムスタンプを使用禁止にするかどうかの指定

例外
`JMSException` ― 内部エラーのために、JMS が使用禁止メッセージ・タイムスタンプの設定に失敗した場合

setPriority(int)

```
public synchronized void setPriority(int priority)
```
プロデューサのデフォルトの優先順位を設定します。

優先順位は、デフォルトで 4 に設定されます。

指定方法
インタフェース javax.jms.MessageProducer 内の javax.jms.MessageProducer.setPriority(int)

パラメータ
`priority` ― このメッセージ・プロデューサのメッセージ優先順位

例外
`JMSException` ― 内部エラーのために、JMS が優先順位の設定に失敗した場合

setTimeToLive(long)

```
public synchronized void setTimeToLive(long timeToLive)
```
発送時以降、メッセージ・システムが生成したメッセージを保持している時間のデフォルト長（ミリ秒）を設定します。

TTL は、デフォルトで 0（ゼロ）に設定されます。

指定方法
インタフェース javax.jms.MessageProducer 内の
javax.jms.MessageProducer.setTimeToLive(long)

パラメータ
`timeToLive` ― メッセージの TTL（ミリ秒）。0（ゼロ）は無制限です。

例外
`JMSException` ― 内部エラーのために、JMS が TTL の設定に失敗した場合

AQjmsQueueBrowser

構文

```
public class AQjmsQueueBrowser extends java.lang.Object
   implements javax.jms.QueueBrowser, java.util.Enumeration

java.lang.Object
  |
  +--oracle.jms.AQjmsQueueBrowser
```

実装される全インタフェース

```
java.util.Enumeration, javax.jms.QueueBrowser
```

説明

このクラスは、QueueBrowser インタフェースを実装します。QueueBrowser は、キュー内のメッセージを、そのメッセージを削除せずに参照するために使用します。

メンバーの概要	説明
メソッド	-
close()	プロバイダは、JVM の外部で QueueBrowser 用に一部のリソースを割り当てている場合があります。このため、クライアントでは、不要なリソースをクローズしておく必要があります。
getEnumeration()	現行のキュー・メッセージを受信順にブラウズするための一覧を取得します。
getMessageSelector()	このキュー・ブラウザのメッセージの選択指定式を取得します。
getQueue()	このキュー・ブラウザに関連付けられているキューを取得します。
getTransformation()	このキュー・ブラウザの変換を取得します。
hasMoreElements()	この一覧にこれ以上の要素が含まれているかどうかをテストします。
nextElement()	この一覧の次の要素を戻します。
setTransformation(String)	このキュー・ブラウザの変換を設定します。

継承メンバーの概要

クラス java.lang.Object から継承されるメソッド

`clone`、`equals`、`finalize`、`getClass`、`hashCode`、`notify`、`notifyAll`、`toString`、`wait`

メソッド

close()

```
public void close()
```

プロバイダは、JVM の外部で QueueBrowser 用に一部のリソースを割り当てている場合があります。このため、クライアントでは、不要なリソースをクローズしておく必要があります。これらのリソースは最終的にガベージ・コレクションで再生されますが、適切な時期に行われない可能性があります。

指定方法

インタフェース javax.jms.QueueBrowser 内の javax.jms.QueueBrowser.close()

getEnumeration()

```
public java.util.Enumeration getEnumeration()
```

現行のキュー・メッセージを受信順にブラウズするための一覧を取得します。

指定方法

インタフェース javax.jms.QueueBrowser 内の javax.jms.QueueBrowser.getEnumeration()

戻り値

メッセージをブラウズするための一覧

例外

`JMSException` － JMS エラーのために、JMS がこのブラウザに対する一覧の取得に失敗した場合

getMessageSelector()

```
public java.lang.String getMessageSelector()
```
このキュー・ブラウザのメッセージの選択指定式を取得します。

指定方法
インタフェース javax.jms.QueueBrowser 内の javax.jms.QueueBrowser.getMessageSelector()

戻り値
このキュー・ブラウザのメッセージ・セレクタ

例外
`JMSException` ― JMS エラーのために、JMS がメッセージ・セレクタの取得に失敗した場合

getQueue()

```
public javax.jms.Queue getQueue()
```
このキュー・ブラウザに関連付けられているキューを取得します。

指定方法
インタフェース javax.jms.QueueBrowser 内の javax.jms.QueueBrowser.getQueue()

戻り値
キュー

例外
`JMSException` ― JMS エラーのために、JMS がこのブラウザに関連付けられているキューの取得に失敗した場合

getTransformation()

```
public String getTransformation()
```
このブラウザの変換を取得します。

戻り値
変換

例外
`JMSException` ― 変換の取得時にエラーが発生した場合

hasMoreElements()

```
public boolean hasMoreElements()
```
この一覧にこれ以上の要素が含まれているかどうかをテストします。

指定方法
インタフェース java.util.Enumeration 内の java.util.Enumeration.hasMoreElements()

戻り値
一覧に要素がまだ存在している場合は true、存在していない場合は false

nextElement()

```
public java.lang.Object nextElement()
```
この一覧の次の要素を戻します。

指定方法
インタフェース java.util.Enumeration 内の java.util.Enumeration.nextElement()

戻り値
この一覧の次の要素

例外
`NoSuchElementException` ― 要素がこれ以上存在しない場合

setTransformation(String)

```
public void setTransformation(String transformation)
```
このブラウザの変換を設定します。メッセージをユーザーに戻す前に、変換が適用されます。

パラメータ
`transformation` ― メッセージを戻す前に適用する変換

例外
`JMSException` ― 変換の設定時にエラーが発生した場合

AQjmsQueueConnectionFactory

構文

```
public class AQjmsQueueConnectionFactory extends java.lang.Object
   implements javax.jms.QueueConnectionFactory

java.lang.Object
  |
  +--oracle.jms.AQjmsQueueConnectionFactory
```

実装される全インタフェース

javax.jms.ConnectionFactory, javax.jms.QueueConnectionFactory,
java.lang.Referenceable, java.lang.Serializable

説明

このクラスは、QueueConnectionFactory インタフェースを実装します。
QueueConnectionFactory は、QueueConnections を作成するために使用します。

メンバーの概要	説明
メソッド	-
createQueueConnection()	この QueueConnectionFactory のホストである JMS Server に対するキュー・コネクションを作成します。
createQueueConnection(Connection)	すでにオープンしている JDBC 接続を使用してキュー・コネクションを作成します。
createQueueConnection(String, String)	接続の作成時の認証に指定されたユーザー名とパスワードを使用して、キュー・コネクションを作成します。

継承メンバーの概要

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、wait

メソッド

createQueueConnection()

```
public javax.jms.QueueConnection createQueueConnection()
```
この QueueConnectionFactory のホストである JMS Server に対するキュー・コネクションを作成します。

指定方法
インタフェース javax.jms.QueueConnectionFactory 内の
javax.jms.QueueConnectionFactory.createQueueConnection()

戻り値
キュー・コネクション

例外
`JMSException` ― JMS エラーのために、JMS がキュー・コネクションの取得に失敗した場合

createQueueConnection(Connection)

```
public static javax.jms.QueueConnection createQueueConnection(java.sql.Connection
jdbc_connection)
```
すでにオープンしている JDBC 接続を使用してキュー・コネクションを作成します。この作成方法では、データベースに対して別の接続が作成されることはありません。かわりに、JMS は指定されたデータベースへの接続をバインドし、JMS で定義されているキューイング・メカニズムへのインタフェースを提供します。

パラメータ
`jdbc_connection` ― データベースに対する有効なオープン接続

戻り値
キュー・コネクション

例外
`JMSException` ― JMS エラーのために、JMS がキュー・コネクションの取得に失敗した場合

createQueueConnection(String, String)

```
public javax.jms.QueueConnection createQueueConnection(java.lang.String username,
java.lang.String password)
```
接続の作成時の認証に指定されたユーザー名とパスワードを使用して、キュー・コネクションを作成します。

指定方法
インタフェース javax.jms.QueueConnectionFactory 内の javax.jms.QueueConnectionFactory.createQueueConnection(java.lang.String, java.lang.String)

パラメータ
`username` − キューイングするために DB に接続するユーザーの名前

`password` − DB に接続するユーザーのパスワード

戻り値
キュー・コネクション

例外
`JMSException` − JMS エラーのために、JMS がキュー・コネクションの取得に失敗した場合

AQjmsQueueReceiver

構文

```
public interface AQjmsQueueReceiver extends javax.jms.QueueReceiver
```

全スーパーインタフェース

```
javax.jms.MessageConsumer, javax.jms.QueueReceiver
```

既知の全実装クラス

```
AQjmsConsumer
```

説明

このインタフェースは javax.jms.QueueReceiver を拡張し、JMS に対する AQ 拡張を定義します。クライアントは、QueueReceiver を使用して、キューに配信されたメッセージを受信します。

メンバーの概要	説明
メソッド	-
getNavigationMode()	メッセージの受信に使用されるナビゲーション・モードを取得します。
getTransformation()	このレシーバの変換を取得します。
receiveNoData()	メッセージを、ユーザーに戻さずに使用します。
receiveNoData(long)	メッセージを、ユーザーに戻さずに使用します。
setNavigationMode(int)	メッセージの受信に使用されるナビゲーション・モードを設定します。
setTransformation(String)	このレシーバの変換を設定します。

継承メンバーの概要

インタフェース javax.jms.QueueReceiver から継承されるメソッド

getQueue

インタフェース javax.jms.MessageConsumer から継承されるメソッド

close、getMessageListener、getMessageSelector、receive、receiveNoWait、setMessageListener

メソッド

getNavigationMode()

```
public int getNavigationMode()
```
メッセージの受信に使用されるナビゲーション・モードを取得します。

戻り値
ナビゲーション・モード

例外
`JMSException` ― ナビゲーション・モードの取得時にエラーが発生した場合

getTransformation()

```
public String getTransformation()
```
このレシーバの変換を取得します。

戻り値
変換

例外
`JMSException` ― 変換の取得時にエラーが発生した場合

receiveNoData()

```
public void receiveNoData()
```
メッセージを、ユーザーに戻さずに使用します。このコールは、データベースからメッセージをフェッチするときのオーバーヘッドを回避します。このため、JMS クライアントが、キュー・ブラウザを使用するなどしてメッセージをすでに取得した場合に、最適化するために使用できます。

例外
`JMSException` ― エラーのためにメッセージの受信ができなかった場合

receiveNoData(long)

```
public void receiveNoData(long timeout)
```
メッセージを、ユーザーに戻さずに使用します。このコールは、データベースからメッセージをフェッチするときのオーバーヘッドを回避します。このため、JMS クライアントが、キュー・ブラウザを使用するなどしてメッセージをすでに取得した場合に、最適化するために使用できます。このコールは、メッセージが到着するまで、またはタイムアウトになるまでブロックします。

パラメータ

`timeout` ― タイムアウト値（ミリ秒）

例外

`JMSException` ― エラーのためにメッセージの受信ができなかった場合

setNavigationMode(int)

```
public void setNavigationMode(int mode)
```
メッセージの受信に使用されるナビゲーション・モードを設定します。

パラメータ

`mode` ― ナビゲーション・モードの新しい値

例外

`JMSException` ― ナビゲーション・モードの取得時にエラーが発生した場合

setTransformation(String)

```
public void setTransformation(String transformation)
```
このレシーバの変換を設定します。メッセージをユーザーに戻す前に、変換が適用されます。

パラメータ

`transformation` ― メッセージを戻す前に適用する変換

例外

`JMSException` ― 変換の設定時にエラーが発生した場合

AQjmsQueueSender

構文

```
public interface AQjmsQueueSender extends javax.jms.QueueSender
```

全スーパーインタフェース

```
javax.jms.MessageProducer, javax.jms.QueueSender
```

既知の全実装クラス

```
AQjmsProducer
```

説明

このインタフェースは QueueSender を拡張し、JMS に対する AQ 拡張を定義します。クライアントは、QueueSender を使用してキューにメッセージを送信します。

メンバーの概要	説明
メソッド	-
getTransformation()	このセンダーの変換を取得します。
setTransformation(String)	このセンダーの変換を設定します。

継承メンバーの概要

インタフェース javax.jms.QueueSender から継承されるメソッド

getQueue、send

インタフェース javax.jms.MessageProducer から継承されるメソッド

close、getDeliveryMode、getDisableMessageID、getDisableMessageTimestamp、getPriority、getTimeToLive、setDeliveryMode、setDisableMessageID、setDisableMessageTimestamp、setPriority、setTimeToLive

メソッド

getTransformation()

```
public String getTransformation()
```
このセンダーの変換を取得します。

戻り値
変換

例外
`JMSException` ― 変換の取得時にエラーが発生した場合

setTransformation(String)

```
public void setTransformation(String transformation)
```
このセンダーの変換を設定します。メッセージがキューに挿入される前にこの変換が適用されます。

パラメータ
`transformation` ― メッセージを送信する前に適用する変換

例外
`JMSException` ― 変換の設定時にエラーが発生した場合

AQjmsSession

構文

```
public class AQjmsSession extends java.lang.Object
   implements javax.jms.QueueSession, javax.jms.TopicSession

java.lang.Object
  |
  +--oracle.jms.AQjmsSession
```

実装される全インタフェース

```
javax.jms.QueueSession, java.lang.Runnable, javax.jms.Session,
javax.jms.TopicSession
```

説明

このクラスは、javax.jms.Session インタフェースを実装します。JMS セッションは、メッセージを生成および使用するための単一スレッド・コンテキストです。

メンバーの概要	説明
メソッド	-
close()	JMS セッションをクローズします。プロバイダは、JVM の外部で Session 用に一部のリソースを割り当てている場合があります。このため、クライアントでは、不要なリソースをクローズしておく必要があります。
commit()	このトランザクションで実行したすべてのメッセージをコミットし、現在保持しているロックを解放します。
createAdtMessage()	AdtMessage を作成します。
createAdtMessage(CustomDatum)	初期化された AdtMessage を作成します。
createBrowser(Queue)	指定したキューのメッセージを参照する QueueBrowser を作成します。
createBrowser(Queue, CustomDatumFactory)	指定したキューで、ユーザー定義型メッセージを含んだメッセージを参照する QueueBrowser を作成します。
createBrowser(Queue, String)	指定したキューのメッセージを参照する QueueBrowser を作成します。

メンバーの概要（続き）	説明
createBrowser(Queue, String, boolean)	指定したキューのメッセージを参照する QueueBrowser を作成します。
createBrowser(Queue, String, CustomDatumFactory)	指定したキューで、ユーザー定義型メッセージを含んだメッセージを参照する QueueBrowser を作成します。
createBrowser(Queue, String, CustomDatumFactory, boolean)	指定したキューで、ユーザー定義型メッセージを含んだメッセージを参照する QueueBrowser を作成します。
createBytesMessage()	BytesMessage を作成します。
createDurableSubscriber(Topic, String)	指定したトピックに対する永続的なサブスクライバを作成します。
createDurableSubscriber(Topic, String, CustomDatumFactory)	指定したトピックに対する永続的なサブスクライバを作成します。
createDurableSubscriber(Topic, String, String, boolean)	指定したトピックに対する永続的なサブスクライバを作成します。
createDurableSubscriber(Topic, String, String, boolean, String)	指定したトピックに対する永続的なサブスクライバを作成します。サブスクライバの変換を指定します。
createDurableSubscriber(Topic, String, String, boolean, CustomDatumFactory, String)	指定した Oracle オブジェクト（ユーザー定義型）トピックに対する永続的なサブスクライバを作成します。
createDurableSubscriber(Topic, String, String, boolean, CustomDatumFactory, String)	指定した Oracle オブジェクト（ユーザー定義型）トピックに対する永続的なサブスクライバを作成します。サブスクライバの変換を指定します。
createMapMessage()	MapMessage を作成します。
createObjectMessage()	ObjectMessage を作成します。
createObjectMessage(Serializable)	初期化された ObjectMessage を作成します。
createPublisher(Topic)	指定したトピックのパブリッシャを作成します。
createQueue(AQQueueTable, String, AQjmsDestinationProperty)	キューを作成します。
createQueueTable(String, String, AQQueueTableProperty)	キュー表を作成します。
createReceiver(Queue)	指定したキューからメッセージを受信する QueueReceiver を作成します。

メンバーの概要（続き）	説明
createReceiver(Queue, CustomDatumFactory)	指定したキューからユーザー定義型メッセージを含んだメッセージを受信する QueueReceiver を作成します。
createReceiver(Queue, String)	指定したキューからメッセージを受信する QueueReceiver を作成します。
createReceiver(Queue, String, CustomDatumFactory)	指定したキューからユーザー定義型メッセージを含んだメッセージを受信する QueueReceiver を作成します。
createRemoteSubscriber(Topic, AQjmsAgent, String)	トピックのリモート・サブスクライバを作成します。
createRemoteSubscriber(Topic, AQjmsAgent, String, String)	トピックのリモート・サブスクライバを作成します。リモート・サブスクライバの変換を指定します。
createRemoteSubscriber(Topic, AQjmsAgent, String, CustomDatumFactory)	トピックのリモート・サブスクライバを作成します。
createRemoteSubscriber(Topic, AQjmsAgent, String, CustomDatumFactory, String)	Oracle オブジェクト（ユーザー定義型）トピックのリモート・サブスクライバを作成します。リモート・サブスクライバの変換を指定します。
createRemoteSubscriber(Topic, AQjmsAgent, String, CustomDatumFactory, String)	指定したキューにメッセージを送信する QueueSender を作成します。
createStreamMessage()	StreamMessage を作成します。
createSubscriber(Topic)	指定したトピックに対する非永続的なサブスクライバを作成します。
createSubscriber(Topic, String, boolean)	指定したトピックに対する非永続的なサブスクライバを作成します。
createTextMessage()	TextMessage を作成します。
createTextMessage(StringBuffer)	初期化された TextMessage を作成します。
createTopic(AQQueueTable, String, AQjmsDestinationProperty)	トピックを作成します。
createTopicReceiver(Topic, String, String)	指定したトピックからメッセージを受信する TopicReceiver を作成します。
createTopicReceiver(Topic, String, String, CustomDatumFactory)	

メンバーの概要（続き）	説明
getDBConnection()	
getJmsConnection()	
getMessageListener()	セッションの識別メッセージ・リスナーを戻します。
getQueue(String, String)	既存のキューを取得します。
getQueueTable(String, String)	既存のキュー表のハンドルを取得します。キュー表の所有者が接続をオープンしたユーザーと異なる場合は、呼出し側にキュー表内のキューまたはトピックの AQ エンキュー / デキュー権限が必要です。
getTopic(String, String)	既存のトピックを取得します。
getTransacted()	セッションがトランザクション・モードかどうかをチェックします。
grantSystemPrivilege(String, String, boolean)	AQ システム権限をユーザーまたはロールに付与します。
revokeSystemPrivilege(String, String)	ユーザーまたはロールから AQ システム権限を取り消します。
rollback()	このトランザクションで実行したすべてのメッセージをロールバックし、現在保持しているロックを解放します。
run()	
setMessageListener(MessageListener)	セッションの識別メッセージ・リスナーを設定します。
unsubscribe(Topic, AQjmsAgent)	指定したトピックにクライアントが作成した永続的なリモート・サブスクリプションを非サブスクライブ化します。
unsubscribe(Topic, String)	指定したトピックにクライアントが作成した永続的なサブスクリプションを非サブスクライブ化します。

継承メンバーの概要

インタフェース javax.jms.Session から継承されるフィールド

AUTO_ACKNOWLEDGE、CLIENT_ACKNOWLEDGE、DUPS_OK_ACKNOWLEDGE

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、wait

メソッド

close()

public void close()
JMS セッションをクローズします。プロバイダは、JVM の外部で Session 用に一部のリソースを割り当てている場合があります。このため、クライアントでは、不要なリソースをクローズしておく必要があります。これらのリソースは最終的にガベージ・コレクションで再生されますが、適切な時期に行われない可能性があります。このコールでは、無限タイムアウトの受信コールでブロックされているレシーバが存在する場合、数分を要する場合があります。

指定方法

インタフェース javax.jms.Session 内の javax.jms.Session.close()

例外

JMSException ー 内部エラーのために、JMS 実装がセッションのクローズに失敗した場合

commit()

public synchronized void commit()
このトランザクションで実行したすべてのメッセージをコミットし、現在保持しているロックを解放します。

指定方法

インタフェース javax.jms.Session 内の javax.jms.Session.commit()

例外

JMSException ー 内部エラーのために、JMS 実装がトランザクションのコミットに失敗した場合。リンクしている SQL 例外に、エラーの詳細情報があります。

createAdtMessage()

```
public synchronized AdtMessage createAdtMessage()
```

AdtMessage を作成します。AdtMessage は、Oracle SQL ユーザー定義型にマップされる Java オブジェクトを含んだメッセージを送信するために使用します。このオブジェクトは、OracleCustomDatum インタフェースをサポートしている必要があります。

例外

JMSException ー メッセージ作成時にエラーが発生した場合

createAdtMessage(CustomDatum)

```
public synchronized AQjmsAdtMessage createAdtMessage(oracle.sql.CustomDatum payload)
```

初期化された AdtMessage を作成します。AQjmsAdtMessage は、Oracle SQL ユーザー定義型にマップされる Java オブジェクトを含んだメッセージを送信するために使用します。このオブジェクトは、OracleCustomDatum インタフェースをサポートしている必要があります。

パラメータ

payload ー このメッセージの初期化に使用するオブジェクト

例外

JMSException ー メッセージ作成時にエラーが発生した場合

createBrowser(Queue)

```
public synchronized javax.jms.QueueBrowser createBrowser(javax.jms.Queue queue)
```

指定したキューのメッセージを参照する QueueBrowser を作成します。このメソッドは、タイプ AQ$_JMS_TEXT_MESSAGE、AQ$_JMS_STREAM_MESSAGE、AQ$_JMS_BYTES_MESSAGE、AQ$_JMS_MAP_MESSAGE または AQ$_JMS_OBJECT_MESSAGE のペイロードを含んだキュー用のブラウザを作成するために使用できます。

指定方法

インタフェース javax.jms.QueueSession 内の
javax.jms.QueueSession.createBrowser(javax.jms.Queue)

パラメータ

queue ― アクセスするキュー

例外

JMSException ― JMS エラーのために、セッションがブラウザの作成に失敗した場合

InvalidDestinationException ― 無効なキューが指定された場合

createBrowser(Queue, CustomDatumFactory)

```
public synchronized javax.jms.QueueBrowser createBrowser(javax.jms.Queue queue,
oracle.sql.CustomDatumFactory payload_factory)
```

指定したキューで、ユーザー定義型メッセージを含んだメッセージを参照する QueueBrowser を作成します。このメソッドは、(標準の JMS 定義ペイロードではなく) Oracle ユーザー定義型ペイロードを含んだキューのレシーバを作成するために使用します。

パラメータ

queue ― アクセスするキュー

payload_factory ― Oracle ユーザー定義型にマップされる Java クラス用の CustomDatumFactory

例外

JMSException ― JMS エラーのために、セッションがブラウザの作成に失敗した場合

InvalidDestinationException ― 無効なキューが指定された場合

createBrowser(Queue, String)

```
public synchronized javax.jms.QueueBrowser createBrowser(javax.jms.Queue queue,
java.lang.String messageSelector)
```

指定したキューのメッセージを参照する QueueBrowser を作成します。このメソッドは、タイプ AQ$_JMS_TEXT_MESSAGE、AQ$_JMS_STREAM_MESSAGE、AQ$_JMS_BYTES_MESSAGE、AQ$_JMS_MAP_MESSAGE または AQ$_JMS_OBJECT_MESSAGE のペイロードを含んだキュー用のブラウザを作成するために使用できます。

指定方法

インタフェース javax.jms.QueueSession 内の javax.jms.QueueSession.createBrowser(javax.jms.Queue, java.lang.String)

パラメータ

queue ― アクセスするキュー

messageSelector ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。セレクタには、次の 1 つ以上を組み合せた式を使用できます。

- 指定したメッセージ ID を持つメッセージを取り出します。JMSMessageID = 'ID:23452345'
- JMS メッセージのヘッダー・フィールドまたはプロパティ

```
JMSPriority < 3 AND JMSCorrelationID = 'Fiction'
```

- ユーザー定義のメッセージ・プロパティ

```
color IN ('RED', BLUE', 'GREEN') AND price < 30000
```

メッセージ ID は接頭辞 'ID:' で始まる必要があります。

例外

JMSException ― JMS エラーのために、セッションがブラウザの作成に失敗した場合

InvalidDestinationException ― 無効なキューが指定された場合

InvalidSelectorException ― メッセージ・セレクタが無効な場合

createBrowser(Queue, String, boolean)

```
public synchronized javax.jms.QueueBrowser createBrowser(javax.jms.Queue queue,
java.lang.String messageSelector, boolean locked)
```

指定したキューのメッセージを参照する QueueBrowser を作成します。このメソッドは、タイプ AQ$_JMS_TEXT_MESSAGE、AQ$_JMS_STREAM_MESSAGE、AQ$_JMS_BYTES_MESSAGE、AQ$_JMS_MAP_MESSAGE または AQ$_JMS_OBJECT_MESSAGE のペイロードを含んだキュー用のブラウザを作成するために使用できます。

パラメータ

queue ― アクセスするキュー

messageSelector ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。

セレクタには、次の 1 つ以上を組み合せた式を使用できます。

- 指定したメッセージ ID を持つメッセージを取り出します。

```
JMSMessageID = 'ID:23452345'
```

- JMS メッセージのヘッダー・フィールドまたはプロパティ

  ```
  JMSPriority < 3 AND JMSCorrelationID = 'Fiction'
  ```

- ユーザー定義のメッセージ・プロパティ

  ```
  color IN ('RED', BLUE', 'GREEN') AND price < 30000
  ```

メッセージ ID は接頭辞 'ID:' で始まる必要があります。

`locked` ― true の場合、メッセージはブラウズするときにロックされます（UPDATE の SELECT と同様）。

例外

`JMSException` ― JMS エラーのために、セッションがブラウザの作成に失敗した場合

`InvalidDestinationException` ― 無効なキューが指定された場合

`InvalidSelectorException` ― メッセージ・セレクタが無効な場合

createBrowser(Queue, String, CustomDatumFactory)

```
public synchronized javax.jms.QueueBrowser createBrowser(javax.jms.Queue queue,
java.lang.String messageSelector, oracle.sql.CustomDatumFactory payload_factory)
```

指定したキューで、ユーザー定義型メッセージを含んだメッセージを参照する QueueBrowser を作成します。このメソッドは、（標準の JMS 定義ペイロードではなく）Oracle ユーザー定義型ペイロードを含んだキューのブラウザを作成するために使用します。

パラメータ

`queue` ― アクセスするキュー

`messageSelector` ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。`AdtMessages` を含むキューの場合、QueueBrowser のセレクタには、メッセージ・ペイロードの内容、メッセージ ID、優先順位または相関 `ID` に基づいた SQL 式を使用できます。

- メッセージ ID のセレクタ ― 指定したメッセージ ID を持つメッセージを取り出します。

  ```
  msgid = '23434556566767676'
  ```

  注意 : この場合、メッセージ ID は接頭辞 'ID:' で始まっていないことが必要です。

- 優先順位または相関のセレクタは次のように指定します。

  ```
  priority < 3 AND corrid = 'Fiction'
  ```

- メッセージ・ペイロードのセレクタは次のように指定します。

  ```
  tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
  ```

payload_factory ― Oracle ユーザー定義型にマップされる Java クラス用の CustomDatumFactory

例外

JMSException ― JMS エラーのために、セッションがブラウザの作成に失敗した場合

InvalidDestinationException ― 無効なキューが指定された場合

createBrowser(Queue, String, CustomDatumFactory, boolean)

```
public synchronized javax.jms.QueueBrowser createBrowser(javax.jms.Queue queue,
java.lang.String messageSelector, oracle.sql.CustomDatumFactory payload_factory,
boolean locked)
```

指定したキューで、ユーザー定義型メッセージを含んだメッセージを参照する QueueBrowser を作成します。このメソッドは、(標準の JMS 定義ペイロードではなく) Oracle ユーザー定義型ペイロードを含んだキューのブラウザを作成するために使用します。

パラメータ

queue ― アクセスするキュー

messageSelector ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。AdtMessages を含むキューの場合、QueueBrowser のセレクタには、メッセージ・ペイロードの内容、メッセージ ID、優先順位または相関 ID に基づいた SQL 式を使用できます。

- メッセージ ID のセレクタ ― 指定したメッセージ ID を持つメッセージを取り出します。

  ```
  msgid = '23434556566767676'
  ```

  注意 : この場合、メッセージ ID は接頭辞 'ID:' で始まっていないことが必要です。

- 優先順位または相関のセレクタは次のように指定します。

  ```
  priority < 3 AND corrid = 'Fiction'
  ```

- メッセージ・ペイロードのセレクタは次のように指定します。

  ```
  tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
  ```

payload_factory ― Oracle ユーザー定義型にマップされる Java クラス用の CustomDatumFactory

locked ― true の場合、メッセージはブラウズするときにロックされます (UPDATE の SELECT と同様)。

例外

`JMSException` ー JMS エラーのために、セッションがブラウザの作成に失敗した場合

`InvalidDestinationException` ー 無効なキューが指定された場合

createBytesMessage()

```
public synchronized javax.jms.BytesMessage createBytesMessage()
```

BytesMessage を作成します。BytesMessage は、未解釈のバイト・ストリームを含んだメッセージを送信するために使用します。

指定方法

インタフェース javax.jms.Session 内の javax.jms.Session.createBytesMessage()

例外

`JMSException` ー メッセージ作成時にエラーが発生した場合

createDurableSubscriber(Topic, String)

```
public synchronized javax.jms.TopicSubscriber
createDurableSubscriber(javax.jms.Topic topic, java.lang.String subs_name)
```

指定したトピックに対する永続的なサブスクライバを作成します。このメソッドは、タイプ AQ$_JMS_TEXT_MESSAGE、AQ$_JMS_STREAM_MESSAGE、AQ$_JMS_BYTES_MESSAGE、AQ$_JMS_MAP_MESSAGE または AQ$_JMS_OBJECT_MESSAGE のペイロードを含んだキューに対するサブスクライバを作成するために使用できます。クライアントは、同じ名前の永続的な TopicSubscriber とメッセージ・セレクタを作成して、既存の永続的なサブスクリプションを変更できます。

指定方法

インタフェース javax.jms.TopicSession 内の
javax.jms.TopicSession.createDurableSubscriber(javax.jms.Topic, java.lang.String)

パラメータ

`topic` ー サブスクライブするトピック

`subs_name` ー このサブスクリプションの識別に使用する名前

例外

`JMSException` ー JMS エラーのために、セッションがサブスクライバの作成に失敗した場合

`InvalidDestinationException` ー 無効なトピックが指定された場合

createDurableSubscriber(Topic, String, CustomDatumFactory)

```
public synchronized javax.jms.TopicSubscriber
createDurableSubscriber(javax.jms.Topic topic, java.lang.String subs_name,
oracle.sql.CustomDatumFactory payload_factory)
```

指定したトピックに対する永続的なサブスクライバを作成します。このメソッドは、（標準の JMS 定義ペイロードではなく）Oracle ユーザー定義型ペイロードを含んだキューのブラウザを作成するために使用します。クライアントは、同じ名前の永続的な TopicSubscriber とメッセージ・セレクタを作成して、既存の永続的なサブスクリプションを変更できます。

パラメータ

`topic` ― サブスクライブするトピック

`subs_name` ― このサブスクリプションの識別に使用する名前

`payload_factory` ― Oracle ユーザー定義型にマップされる Java クラス用の CustomDatumFactory

例外

`JMSException` ― JMS エラーのために、セッションがサブスクライバの作成に失敗した場合

`InvalidDestinationException` ― 無効なトピックが指定された場合

createDurableSubscriber(Topic, String, String, boolean)

```
public synchronized javax.jms.TopicSubscriber
createDurableSubscriber(javax.jms.Topic topic, java.lang.String subs_name,
java.lang.String messageSelector, boolean noLocal)
```

指定したトピックに対する永続的なサブスクライバを作成します。このメソッドは、タイプ AQ$_JMS_TEXT_MESSAGE、AQ$_JMS_STREAM_MESSAGE、AQ$_JMS_BYTES_MESSAGE、AQ$_JMS_MAP_MESSAGE または AQ$_JMS_OBJECT_MESSAGE のペイロードを含んだキューに対するサブスクライバを作成するために使用できます。クライアントは、同じ名前の永続的な TopicSubscriber とメッセージ・セレクタを作成して、既存の永続的なサブスクリプションを変更できます。

指定方法

インタフェース javax.jms.TopicSession 内の javax.jms.TopicSession.createDurableSubscriber(javax.jms.Topic, java.lang.String, java.lang.String, boolean)

パラメータ

`topic` ― サブスクライブするトピック

`subs_name` ― このサブスクリプションの識別に使用する名前

messageSelector ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。この値には null を設定することもできます。

セレクタには、次の 1 つ以上を組み合せた任意の SQL92 式を含めることができます。

a. JMS メッセージのヘッダー・フィールドまたはプロパティ JMSPriority (int)、JMSCorrelationID (string)、JMSType (string)、JMSXUserID (string)、JMSXAppID (string)、JMSXGroupID (string)、JMSXGroupSeq (int)

例 : `JMSPriority < 3 AND JMSCorrelationID = 'Fiction'`

b. ユーザー定義のメッセージ・プロパティ

例 : `color IN ('RED', BLUE', 'GREEN') AND price < 30000`

使用可能な演算子は次のとおりです。

― 優先順位順の論理演算子 NOT、AND、OR

― 比較演算子 =、>、>=、<、<=、<>、!（<> および ! は等しくないことを示すために使用できます。）

― 優先順位順の算術演算子 +、- 単項、*、/、+、-

― 識別子 [NOT] IN (string-literal1, string-literal2, ..)

― arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 and arithmetic-expr3

― 識別子 [NOT] LIKE パターン - 値 [ESCAPE escape-character] パターン - 値は、文字列リテラルです。「%」は、文字列を表し、「_」は、単一文字を表します。オプションのエスケープ文字は、パターン - 値において特別な意味の「-」および「%」をエスケープするために使用します。

― 識別子 IS [NOT] NULL

noLocal ― false に設定する必要があります。noLocal=true はサポートされていません。

例外

JMSException ― JMS エラーのために、セッションがサブスクライバの作成に失敗した場合

InvalidDestinationException ― 無効なトピックが指定された場合

InvalidSelectorException ― メッセージ・セレクタが無効な場合

createDurableSubscriber(Topic, String, String, boolean, String)

```
public synchronized javax.jms.TopicSubscriber
createDurableSubscriber(javax.jms.Topic topic, java.lang.String subs_name,
java.lang.String messageSelector, boolean noLocal, String transformation)
```

指定したトピックに対する永続的なサブスクライバを作成し、そのサブスクライバの変換を指定します。

このメソッドは、タイプ AQ$_JMS_TEXT_MESSAGE、AQ$_JMS_STREAM_MESSAGE、AQ$_JMS_BYTES_MESSAGE、AQ$_JMS_MAP_MESSAGE または AQ$_JMS_OBJECT_MESSAGE のペイロードを含んだトピックに対するサブスクライバを作成するために使用できます。クライアントは、同じ名前の永続的な TopicSubscriber とメッセージ・セレクタを作成して、既存の永続的なサブスクリプションを変更できます。

指定方法

インタフェース javax.jms.TopicSession 内の
javax.jms.TopicSession.createDurableSubscriber(javax.jms.Topic, java.lang.String, java.lang.String, boolean)

パラメータ

`topic` ― サブスクライブするトピック

`subs_name` ― このサブスクリプションの識別に使用する名前

`messageSelector` ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。この値には null を設定することもできます。

セレクタには、次の 1 つ以上を組み合せた任意の SQL92 式を含めることができます。

a. JMS メッセージのヘッダー・フィールドまたはプロパティ JMSPriority (int)、JMSCorrelationID (string)、JMSType (string)、JMSXUserID (string)、JMSXAppID (string)、JMSXGroupID (string)、JMSXGroupSeq (int)

例 : `JMSPriority < 3 AND JMSCorrelationID = 'Fiction'`

b. ユーザー定義のメッセージ・プロパティ

例 : `color IN ('RED', BLUE', 'GREEN') AND price < 30000`

使用可能な演算子は次のとおりです。

― 優先順位順の論理演算子 NOT、AND、OR

― 比較演算子 =、>、>=、<、<=、<>、!（<> および ! は等しくないことを示すために使用できます。）

― 優先順位順の算術演算子 +、- 単項、*、/、+、-

― 識別子 [NOT] IN (string-literal1, string-literal2, ..)

— arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 and arithmetic-expr3

— 識別子 [NOT] LIKE パターン - 値 [ESCAPE escape-character] パターン - 値は、文字列リテラルです。「%」は、文字列を表し、「_」は、単一文字を表します。オプションのエスケープ文字は、パターン - 値において特別な意味の「-」および「%」をエスケープするために使用します。

— 識別子 IS [NOT] NULL

`noLocal` — false に設定する必要があります。noLocal=true はサポートされていません。

`transformation` — このサブスクライバに関連付けられている変換。このサブスクライバにメッセージが配信される前に、この変換が適用されます。

例外

`JMSException` — JMS エラーのために、セッションがサブスクライバの作成に失敗した場合

`InvalidDestinationException` — 無効なトピックが指定された場合

`InvalidSelectorException` — メッセージ・セレクタが無効な場合

createDurableSubscriber(Topic, String, String, boolean, CustomDatumFactory, String)

```
public synchronized javax.jms.TopicSubscriber
createDurableSubscriber(javax.jms.Topic topic, java.lang.String subs_name,
java.lang.String messageSelector, boolean noLocal, oracle.sql.CustomDatumFactory
payload_factory, String transformation)
```

指定したトピックに対する永続的なサブスクライバを作成し、そのサブスクライバの変換を指定します。

このメソッドは、（標準の JMS 定義ペイロードではなく）Oracle ユーザー定義型ペイロードを含んだトピックのサブスクライバを作成するために使用します。クライアントは、同じ名前の永続的な TopicSubscriber とメッセージ・セレクタを作成して、既存の永続的なサブスクリプションを変更できます。

パラメータ

`topic` — サブスクライブするトピック

`subs_name` — このサブスクリプションの識別に使用する名前

`messageSelector` — メッセージの選択指定式と一致する属性を持つメッセージのみ配信されます。この値には null を設定することもできます。

ユーザー定義型メッセージを含んだキューのセレクタの構文は、標準の JMS ペイロード（テキスト、ストリーム、オブジェクト、バイト、マップ）を含んだキューのセレクタの構文とは異なります。セレクタは、AQ ルール構文と同じです。

a. 優先順位または相関のセレクタは次のように指定します。例：`priority < 3 AND corrid = 'Fiction'`

b. メッセージ・ペイロードのセレクタは次のように指定します。属性名は、接頭辞 tab.user_data で始まる必要があります。例：`tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000`

`noLocal` ― false に設定する必要があります。noLocal=true はサポートされていません。

`payload_factory` ― Oracle ユーザー定義型メッセージにマップされる Java クラス用の CustomDatumFactory。ユーザー定義プロパティは含まれません。

`transformation` ― このサブスクライバに関連付けられている変換。このサブスクライバにメッセージが配信される前に、この変換が適用されます。

例外

`JMSException` ― JMS エラーのために、セッションがサブスクライバの作成に失敗した場合

`InvalidDestinationException` ― 無効なトピックが指定された場合

createMessage()

```
public synchronized javax.jms.Message createMessage()
```
一般的なヘッダーのみのメッセージを作成します。

指定方法

インタフェース javax.jms.Session 内の javax.jms.Session.createMessage()

例外

`JMSException` ― メッセージ作成時にエラーが発生した場合

createMapMessage()

```
public synchronized javax.jms.MapMessage createMapMessage()
```
MapMessage を作成します。MapMessage は、自己定義の名前 - 値のペアを送信するために使用します。名前は String 型、値は Java プリミティブ型です。

指定方法

インタフェース javax.jms.Session 内の javax.jms.Session.createMapMessage()

例外

`JMSException` ― メッセージ作成時にエラーが発生した場合

createObjectMessage()

```
public synchronized javax.jms.ObjectMessage createObjectMessage()
```
ObjectMessage を作成します。ObjectMessage は、シリアル化可能な Java オブジェクトを含んだメッセージを送信するために使用します。

指定方法
インタフェース javax.jms.Session 内の javax.jms.Session.createObjectMessage()

例外
`JMSException` ― メッセージ作成時にエラーが発生した場合

createObjectMessage(Serializable)

```
public synchronized javax.jms.ObjectMessage
createObjectMessage(java.io.Serializable object)
```
初期化された ObjectMessage を作成します。ObjectMessage は、シリアル化可能な Java オブジェクトを含んだメッセージを送信するために使用します。

指定方法
インタフェース javax.jms.Session 内の
javax.jms.Session.createObjectMessage(java.io.Serializable)

パラメータ
`object` ― このメッセージの初期化に使用するオブジェクト

例外
`JMSException` ― メッセージ作成時にエラーが発生した場合

createPublisher(Topic)

```
public synchronized javax.jms.TopicPublisher createPublisher(javax.jms.Topic topic)
```
指定したトピックのパブリッシャを作成します。クライアントは、TopicPublisher を使用して、トピックにメッセージを公開します。

指定方法
インタフェース javax.jms.TopicSession 内の
javax.jms.TopicSession.createPublisher(javax.jms.Topic)

パラメータ
`topic` ― 公開先のトピック、またはプロデューサが未指定の場合は null

例外

JMSException ー JMS エラーのために、セッションがパブリッシャの作成に失敗した場合

InvalidDestinationException ー 無効なトピックが指定された場合

createQueue()

```
public synchronized javax.jms.Queue createQueue(String)
```

キュー名を指定してキューを作成します。このメソッドは、クライアントがキュー ID を動的に操作する必要がある場合（通常は不要です）に備えて用意されています。プロバイダ固有の名前でキュー ID を作成できます。この機能に依存しているクライアントは移植できません。このメソッドは、物理的なキューを作成するためのメソッドではないことに注意してください。キューの物理的な作成は管理タスクの 1 つです。

キュー名の形式は [schema].name です。schema が指定されていない場合は、現行セッションの user が使用されます。

指定方法

インタフェース javax.jms.QueueSession 内の javax.jms.QueueSession.createQueue()

例外

JMSException ー キューを作成できなかった場合

createQueue(AQQueueTable, String, AQjmsDestinationProperty)

```
public synchronized javax.jms.Queue createQueue(oracle.jms.AQQueueTable q_table,
java.lang.String queue_name, AQjmsDestinationProperty dest_property)
```

キューを作成します。

パラメータ

q_table ー キューを作成するキュー表。マルチ・コンシューマ対応ではないキュー表であることが必要です。

queue_name ー 作成するキューの名前

dest_property ー キューのプロパティ

例外

JMSException ー キューを作成できなかった場合

関連項目

AQjmsDestinationProperty

createQueueTable(String, String, AQQueueTableProperty)

```
public synchronized oracle.jms.AQQueueTable createQueueTable(java.lang.String owner,
java.lang.String name, oracle.jms.AQQueueTableProperty property)
```
キュー表を作成します。キュー表には、キューまたはトピックが保持されます。

パラメータ

owner ― キュー表の所有者（スキーマ）

name ― キュー表の名前

property ― キュー表のプロパティ。キュー表がキューの保持に使用される場合、そのキュー表はマルチ・コンシューマ対応ではない（デフォルト）ことが必要です。キュー表がトピックの保持に使用される場合、そのキュー表はマルチ・コンシューマ対応であることが必要です。

例外

JMSException ― キュー表を作成できなかった場合

関連項目

oracle.AQ.AQQueueTableProperty

createReceiver(Queue)

```
public synchronized javax.jms.QueueReceiver createReceiver(javax.jms.Queue queue)
```
指定したキューからメッセージを受信する QueueReceiver を作成します。このメソッドは、タイプ AQ$_JMS_TEXT_MESSAGE、AQ$_JMS_STREAM_MESSAGE、AQ$_JMS_BYTES_MESSAGE、AQ$_JMS_MAP_MESSAGE または AQ$_JMS_OBJECT_MESSAGE のペイロードを含んだキューに対するレシーバを作成するために使用できます。

指定方法

インタフェース javax.jms.QueueSession 内の
javax.jms.QueueSession.createReceiver(javax.jms.Queue)

パラメータ

queue ― アクセスするキュー

例外

JMSException ― JMS エラーのために、セッションがレシーバの作成に失敗した場合

InvalidDestinationException ― 無効なキューが指定された場合

createReceiver(Queue, CustomDatumFactory)

```
public synchronized javax.jms.QueueReceiver createReceiver(javax.jms.Queue queue,
oracle.sql.CustomDatumFactory payload_factory)
```

指定したキューからユーザー定義型メッセージを含んだメッセージを受信する QueueReceiver を作成します。このメソッドは、（標準の JMS 定義ペイロードではなく）Oracle ユーザー定義型ペイロードを含んだキューのレシーバを作成するために使用します。

パラメータ

`queue` ― アクセスするキュー

`payload_factory` ― Oracle ユーザー定義型にマップされる Java クラス用の CustomDatumFactory

例外

`JMSException` ― JMS エラーのために、セッションがレシーバの作成に失敗した場合

`InvalidDestinationException` ― 無効なキューが指定された場合

createReceiver(Queue, String)

```
public synchronized javax.jms.QueueReceiver createReceiver(javax.jms.Queue queue,
java.lang.String messageSelector)
```

指定したキューからメッセージを受信する QueueReceiver を作成します。このメソッドは、タイプ AQ$_JMS_TEXT_MESSAGE、AQ$_JMS_STREAM_MESSAGE、AQ$_JMS_BYTES_MESSAGE、AQ$_JMS_MAP_MESSAGE または AQ$_JMS_OBJECT_MESSAGE のペイロードを含んだキューに対するレシーバを作成するために使用できます。

指定方法

インタフェース javax.jms.QueueSession 内の javax.jms.QueueSession.createReceiver(javax.jms.Queue, java.lang.String)

パラメータ

`queue` ― アクセスするキュー

`messageSelector` ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。セレクタには、次の 1 つ以上を組み合せた式を使用できます。

- 指定したメッセージ ID を持つメッセージを取り出します。

  ```
  JMSMessageID = 'ID:23452345'
  ```

- JMS メッセージのヘッダー・フィールドまたはプロパティ

  ```
  JMSPriority < 3 AND JMSCorrelationID = 'Fiction'
  ```

- ユーザー定義のメッセージ・プロパティ

  ```
  color IN ('RED', BLUE', 'GREEN') AND price < 30000
  ```

メッセージ ID は接頭辞 'ID:' で始まる必要があります。

例外

`JMSException` − JMS エラーのために、セッションがレシーバの作成に失敗した場合

`InvalidDestinationException` − 無効なキューが指定された場合

`InvalidSelectorException` − メッセージ・セレクタが無効な場合

createReceiver(Queue, String, CustomDatumFactory)

```
public synchronized javax.jms.QueueReceiver createReceiver(javax.jms.Queue queue,
java.lang.String messageSelector, oracle.sql.CustomDatumFactory payload_factory)
```

指定したキューからユーザー定義型メッセージを含んだメッセージを受信する QueueReceiver を作成します。このメソッドは、(標準の JMS 定義ペイロードではなく) Oracle ユーザー定義型ペイロードを含んだキューのレシーバを作成するために使用します。

パラメータ

`queue` − アクセスするキュー

`messageSelector` − メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。`AdtMessages` を含んだキューの場合、QueueReceiver のセレクタには、メッセージ・ペイロードの内容、メッセージ ID、優先順位または相関 `ID` に基づいた SQL 式を使用できます。

- メッセージ ID のセレクタ − 指定したメッセージ ID を持つメッセージを取り出します。

  ```
  msgid = '23434556566767676'
  ```

  注意 : この場合、メッセージ ID は接頭辞 'ID:' で始まっていないことが必要です。

- 優先順位または相関のセレクタは次のように指定します。

  ```
  priority < 3 AND corrid = 'Fiction'
  ```

- メッセージ・ペイロードのセレクタは次のように指定します。

  ```
  tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
  ```

`payload_factory` − Oracle ユーザー定義型にマップされる Java クラス用の CustomDatumFactory

例外

`JMSException` ― JMS エラーのために、セッションがレシーバの作成に失敗した場合

`InvalidDestinationException` ― 無効なキューが指定された場合

`InvalidSelectorException` ― メッセージ・セレクタが無効な場合

createRemoteSubscriber(Topic, AQjmsAgent, String)

```
public synchronized void createRemoteSubscriber(javax.jms.Topic topic, AQjmsAgent
remote_subscriber, java.lang.String messageSelector)
```

トピックのリモート・サブスクライバを作成します。このメソッドは、タイプ AQ$_JMS_TEXT_MESSAGE、AQ$_JMS_STREAM_MESSAGE、AQ$_JMS_BYTES_MESSAGE、AQ$_JMS_MAP_MESSAGE または AQ$_JMS_OBJECT_MESSAGE のペイロードを含んだキュー用のリモート・サブスクライバを作成するために使用できます。

AQ では、トピックにリモート・サブスクライバを設定できます。つまり、同じデータベースまたは別のデータベース内の他のトピックのサブスクライバを設定できます。リモート・サブスクライバを使用するには、ローカルおよびリモートの 2 つのトピック間の伝播を設定する必要があります。

パラメータ

`topic` ― サブスクライブするトピック

`remote_subscriber` ― リモート・サブスクライバを表す AQjmsAgent

`messageSelector` ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。この値には null を設定することができます。

セレクタ構文は、createDurableSubscriber の構文と同じです。リモート・サブスクライバには、リモート・トピックの特定のコンシューマまたはリモート・トピックのすべてのサブスクライバを設定できます。

リモート・サブスクライバは、AQjmsAgent 構造を使用して定義します。AQjmsAgent は、名前とアドレスで構成されます。名前は、リモート・トピックの consumer_name を表します。アドレスは、リモート・トピックを表します。構文は (schema).(topic_name)[@dblink] です。

1. リモート・トピックの特定のコンシューマにメッセージを公開するには、AQjmsAgent の名前フィールドに、リモート・トピックのレシピエントの subscription_name を指定する必要があります。また、AQjmsAgent のアドレス・フィールドにリモート・トピックを指定してください。

2. リモート・トピックのすべてのサブスクライバにメッセージを公開するには、AQjmsAgent の名前フィールドに null を設定する必要があります。また、AQjmsAgent のアドレス・フィールドにリモート・トピックを指定してください。

createRemoteSubscriber(Topic, AQjmsAgent, String, String)

```
public synchronized void createRemoteSubscriber(javax.jms.Topic topic, AQjmsAgent
remote_subscriber, java.lang.String messageSelector, java.lang.String
transformation)
```

トピックのリモート・サブスクライバを作成し、そのサブスクライバの変換を指定します。このメソッドは、タイプ AQ$_JMS_TEXT_MESSAGE、AQ$_JMS_STREAM_MESSAGE、AQ$_JMS_BYTES_MESSAGE、AQ$_JMS_MAP_MESSAGE または AQ$_JMS_OBJECT_MESSAGE のペイロードを含んだキュー用のリモート・サブスクライバを作成するために使用できます。

AQ では、トピックにリモート・サブスクライバを設定できます。つまり、同じデータベースまたは別のデータベース内の他のトピックのサブスクライバを設定できます。リモート・サブスクライバを使用するには、ローカルおよびリモートの 2 つのトピック間の伝播を設定する必要があります。

パラメータ

`topic` ― サブスクライブするトピック

`remote_subscriber` ― リモート・サブスクライバを表す AQjmsAgent

`messageSelector` ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。この値には null を設定することができます。

セレクタ構文は、createDurableSubscriber の構文と同じです。リモート・サブスクライバには、リモート・トピックの特定のコンシューマまたはリモート・トピックのすべてのサブスクライバを設定できます。

`transformation` ― このサブスクライバの変換。メッセージがサブスクライバに配信される前に、この変換が適用されます。

リモート・サブスクライバは、AQjmsAgent 構造を使用して定義します。AQjmsAgent は、名前とアドレスで構成されます。名前は、リモート・トピックの consumer_name を表します。アドレスは、リモート・トピックを表します。構文は (schema).(topic_name)[@dblink] です。

1. リモート・トピックの特定のコンシューマにメッセージを公開するには、AQjmsAgent の名前フィールドに、リモート・トピックのレシピエントの subscription_name を指定する必要があります。また、AQjmsAgent のアドレス・フィールドにリモート・トピックを指定してください。

2. リモート・トピックのすべてのサブスクライバにメッセージを公開するには、AQjmsAgent の名前フィールドに null を設定する必要があります。また、AQjmsAgent のアドレス・フィールドにリモート・トピックを指定してください。

createRemoteSubscriber(Topic, AQjmsAgent, String, CustomDatumFactory)

```
public synchronized void createRemoteSubscriber(javax.jms.Topic topic, AQjmsAgent
remote_subscriber, java.lang.String messageSelector, oracle.sql.CustomDatumFactory
payload_factory)
```

トピックのリモート・サブスクライバを作成します。このメソッドは、（標準の JMS 定義ペイロードではなく）Oracle ユーザー定義型ペイロードを含んだキューのブラウザを作成するために使用します。

AQ では、トピックにリモート・サブスクライバを設定できます。つまり、同じデータベースまたは別のデータベース内の他のトピックのサブスクライバを設定できます。リモート・サブスクライバを使用するには、ローカルおよびリモートの 2 つのトピック間の伝播を設定する必要があります。

パラメータ

`topic` ― サブスクライブするトピック

`remote_subscriber` ― リモート・サブスクライバを表す AQjmsAgent

`messageSelector` ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。この値には null を設定することができます。セレクタ構文は、ユーザー定義型メッセージを含むトピック用の createDurableSubscriber の構文と同じです。

`payload_factory` ― Oracle ユーザー定義型にマップされる Java クラス用の CustomDatumFactory

リモート・サブスクライバには、リモート・トピックの特定のコンシューマまたはリモート・トピックのすべてのサブスクライバを設定できます。リモート・サブスクライバは、AQjmsAgent 構造を使用して定義します。AQjmsAgent は、名前とアドレスで構成されます。名前は、リモート・トピックの consumer_name を表します。アドレスは、リモート・トピックを表します。構文は (schema).(topic_name)[@dblink] です。

1. リモート・トピックの特定のコンシューマにメッセージを公開するには、AQjmsAgent の名前フィールドに、リモート・トピックのレシピエントの subscription_name を指定する必要があります。また、AQjmsAgent のアドレス・フィールドにリモート・トピックを指定してください。

2. リモート・トピックのすべてのサブスクライバにメッセージを公開するには、AQjmsAgent の名前フィールドに null を設定する必要があります。また、AQjmsAgent のアドレス・フィールドにリモート・トピックを指定してください。

createRemoteSubscriber(Topic, AQjmsAgent, String, CustomDatumFactory, String)

```
public synchronized void createRemoteSubscriber(javax.jms.Topic topic,
AQjmsAgent remote_subscriber, java.lang.String messageSelector,
oracle.sql.CustomDatumFactory payload_factory, String transformation)
```

トピックのリモート・サブスクライバを作成し、このサブスクライバの変換を指定します。このメソッドは、（標準の JMS 定義ペイロードではなく）Oracle ユーザー定義型ペイロードを含んだキューのブラウザを作成するために使用します。

AQ では、トピックにリモート・サブスクライバを設定できます。つまり、同じデータベースまたは別のデータベース内の他のトピックのサブスクライバを設定できます。リモート・サブスクライバを使用するには、ローカルおよびリモートの 2 つのトピック間の伝播を設定する必要があります。

パラメータ

`topic` ― サブスクライブするトピック

`remote_subscriber` ― リモート・サブスクライバを表す AQjmsAgent

`messageSelector` ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。この値には null を設定することもできます。セレクタ構文は、ユーザー定義型メッセージを含むトピック用の createDurableSubscriber の構文と同じです。

`payload_factory` ― Oracle ユーザー定義型にマップされる Java クラス用の CustomDatumFactory

`transformation` ― このサブスクライバの変換。メッセージがサブスクライバに配信される前に、この変換が適用されます。

リモート・サブスクライバには、リモート・トピックの特定のコンシューマまたはリモート・トピックのすべてのサブスクライバを設定できます。リモート・サブスクライバは、AQjmsAgent 構造を使用して定義します。AQjmsAgent は、名前とアドレスで構成されます。名前は、リモート・トピックの consumer_name を表します。アドレスは、リモート・トピックを表します。構文は (schema).(topic_name)[@dblink] です。

1. リモート・トピックの特定のコンシューマにメッセージを公開するには、AQjmsAgent の名前フィールドに、リモート・トピックのレシピエントの subscription_name を指定する必要があります。また、AQjmsAgent のアドレス・フィールドにリモート・トピックを指定してください。

2. リモート・トピックのすべてのサブスクライバにメッセージを公開するには、AQjmsAgent の名前フィールドに null を設定する必要があります。また、AQjmsAgent のアドレス・フィールドにリモート・トピックを指定してください。

createSender(Queue)

```
public synchronized javax.jms.QueueSender createSender(javax.jms.Queue queue)
```
指定したキューにメッセージを送信する QueueSender を作成します。

指定方法
インタフェース javax.jms.QueueSession 内の
javax.jms.QueueSession.createSender(javax.jms.Queue)

パラメータ
`queue` ― アクセスするキュー、プロデューサが未指定の場合は null

例外
`JMSException` ― JMS エラーのために、セッションがセンダーの作成に失敗した場合

`InvalidDestinationException` ― 無効なキューが指定された場合

createStreamMessage()

```
public synchronized javax.jms.StreamMessage createStreamMessage()
```
StreamMessage を作成します。StreamMessage は、Java プリミティブ型の自己定義ストリームを送信するために使用します。

指定方法
インタフェース javax.jms.Session 内の javax.jms.Session.createStreamMessage()

例外
`JMSException` ― メッセージ作成時にエラーが発生した場合

createSubscriber(Topic)

```
public synchronized javax.jms.TopicSubscriber createSubscriber
(javax.jms.Topic topic)
```
指定したトピックに対する非永続的なサブスクライバを作成します。

指定方法
インタフェース javax.jms.TopicSession 内の
javax.jms.TopicSession.createSubscriber(javax.jms.Topic)

例外
`JMSException` ― 非永続サブスクライバの作成時にエラーが発生した場合

createSubscriber(Topic, String, boolean)

```
public synchronized javax.jms.TopicSubscriber createSubscriber
(javax.jms.Topic topic, java.lang.String messageSelector, boolean noLocal)
```
指定したトピックに対する非永続的なサブスクライバを作成します。

指定方法
インタフェース javax.jms.TopicSession 内の
javax.jms.TopicSession.createSubscriber(javax.jms.Topic, java.lang.String, boolean)

例外
JMSException ― 非永続サブスクライバ作成時にエラーが発生した場合

createTemporaryQueue()

```
public synchronized javax.jms.TemporaryQueue createTemporaryQueue()
```
一時キューを作成します。存続期間は、先に削除されないかぎり QueueConnection の存続期間と同じです。

指定方法
インタフェース javax.jms.Session 内の javax.jms.Session.createTemporaryQueue()

例外
JMSException ― 一時キューを作成できなかった場合

createTemporaryTopic()

```
public synchronized javax.jms.TemporaryTopic createTemporaryTopic()
```
一時トピックを作成します。存続期間は、先に削除されていないかぎり、TopicConnection の存続期間と同じです。

指定方法
インタフェース javax.jms.Session 内の javax.jms.Session.createTemporaryTopic()

例外
JMSException ― 一時トピックを作成できなかった場合

createTextMessage()

```
public synchronized javax.jms.TextMessage createTextMessage()
```
TextMessage を作成します。TextMessage は、StringBuffer を含んだメッセージを送信するために使用します。

指定方法
インタフェース javax.jms.Session 内の javax.jms.Session.createTextMessage()

例外
`JMSException` ― メッセージ作成時にエラーが発生した場合

createTextMessage(StringBuffer)

```
public synchronized javax.jms.TextMessage createTextMessage(java.lang.StringBuffer
stringBuffer)
```
初期化された TextMessage を作成します。TextMessage は、StringBuffer を含んだメッセージを送信するために使用します。

指定方法
インタフェース javax.jms.Session 内の
javax.jms.Session.createTextMessage(java.lang.StringBuffer)

パラメータ
`stringBuffer` ― このメッセージを初期化するために使用する文字列バッファ

例外
`JMSException` ― メッセージ作成時にエラーが発生した場合

createTopic()

```
public synchronized javax.jms.Topic createTopic(String)
```
トピック名を指定してトピックを作成します。このメソッドは、クライアントがトピック ID を動的に操作する必要がある場合（通常は不要です）に備えて用意されています。プロバイダ固有の名前でトピック ID を作成できます。この機能に依存するクライアントは移植できません。このメソッドは、物理的なトピックを作成するためのメソッドではないことに注意してください。トピックの物理的な作成は管理タスクの 1 つです。

トピック名の形式は [schema].name です。schema が指定されていない場合は、現行セッションの user が使用されます。

指定方法

インタフェース javax.jms.QueueSession 内の javax.jms.QueueSession.createQueue()

例外

JMSException ― キューを作成できなかった場合

createTopic(AQQueueTable, String, AQjmsDestinationProperty)

```
public synchronized javax.jms.Topic createTopic(oracle.jms.AQQueueTable q_table,
java.lang.String topic_name, AQjmsDestinationProperty dest_property)
```

トピックを作成します。

パラメータ

q_table ― トピックが作成されるキュー表。マルチ・コンシューマ対応のキュー表であることが必要です。

topic_name ― 作成するトピックの名前

dest_property ― トピックのプロパティ

例外

JMSException ― トピックを作成できなかった場合

関連項目

AQjmsDestinationProperty

createTopicReceiver(Topic, String, String)

```
public synchronized TopicReceiver createTopicReceiver(javax.jms.Topic topic,
java.lang.String receiver_name, java.lang.String messageSelector)
```

指定したトピックからメッセージを受信する TopicReceiver を作成します。AQ では、メッセージをトピックのすべてのサブスクライバまたは指定したレシピエントに送信できます。このレシーバは、トピックのサブスクライバであってもなくてもかまいません。レシーバがトピックのサブスクライバでない場合は、明示されているメッセージのみ受信します。このメソッドは、トピックの永続的なサブスクライバではないコンシューマに対して TopicReceiver を作成するために使用してください。

このメソッドは、タイプ AQ$_JMS_TEXT_MESSAGE、AQ$_JMS_STREAM_MESSAGE、AQ$_JMS_BYTES_MESSAGE、AQ$_JMS_MAP_MESSAGE または AQ$_JMS_OBJECT_MESSAGE のペイロードを含んだトピック用の TopicReceiver を作成するために使用できます。

パラメータ

topic ― アクセスするトピック

receiver_name ― レシピエント（またはサブスクライバ）の名前

messageSelector ― メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。セレクタには、次の 1 つ以上を組み合せた式を使用できます。

- 指定したメッセージ ID を持つメッセージを取り出します。

  ```
  JMSMessageID = 'ID:23452345'
  ```

- JMS メッセージのヘッダー・フィールドまたはプロパティ

  ```
  JMSPriority < 3 AND JMSCorrelationID = 'Fiction'
  ```

- ユーザー定義のメッセージ・プロパティ

  ```
  color IN ('RED', BLUE', 'GREEN') AND price < 30000
  ```

メッセージ ID は接頭辞 'ID:' で始まる必要があります。

例外

JMSException ― JMS エラーのために、セッションがレシーバの作成に失敗した場合

InvalidDestinationException ― 無効なトピックが指定された場合

InvalidSelectorException ― メッセージ・セレクタが無効な場合

createTopicReceiver(Topic, String, String, CustomDatumFactory)

```
public synchronized TopicReceiver createTopicReceiver(javax.jms.Topic topic,
java.lang.String receiver_name, java.lang.String messageSelector,
oracle.sql.CustomDatumFactory payload_factory)
```

指定したトピックからユーザー定義型メッセージを含んだメッセージを受信する TopicReceiver を作成します。AQ では、メッセージをトピックのすべてのサブスクライバまたは指定したレシピエントに送信できます。このレシーバは、トピックのサブスクライバであってもなくてもかまいません。レシーバがトピックのサブスクライバでない場合は、明示されているメッセージのみ受信します。このメソッドは、トピックの永続的なサブスクライバではないコンシューマに対して TopicReceiver を作成するために使用してください。

このメソッドは、（標準の JMS 定義ペイロードではなく）Oracle ユーザー定義型ペイロードを含んだトピックの TopicReceiver を作成するために使用します。

パラメータ

topic − アクセスするトピック

receiver_name − レシピエント（またはサブスクライバ）の名前

messageSelector − メッセージの選択指定式と一致するプロパティを持つメッセージのみ配信されます。AdtMessages を含んだキューの場合、セレクタには、メッセージ・ペイロードの内容、メッセージ ID、優先順位または相関 ID に基づいた SQL 式を使用できます。

- メッセージ ID のセレクタ − 指定したメッセージ ID を持つメッセージを取り出します。

  ```
  msgid = '23434556566767676'
  ```

  注意：この場合、メッセージ ID は接頭辞 'ID:' で始まっていないことが必要です。

- 優先順位または相関のセレクタは次のように指定します。

  ```
  priority < 3 AND corrid = 'Fiction'
  ```

- メッセージ・ペイロードのセレクタは次のように指定します。

  ```
  tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
  ```

payload_factory − Oracle ユーザー定義型にマップされる Java クラス用の CustomDatumFactory

getDBConnection()

```
public synchronized java.sql.Connection getDBConnection()
```

getJmsConnection()

```
public AQjmsConnection getJmsConnection()
```

getMessageListener()

```
public synchronized javax.jms.MessageListener getMessageListener()
```

セッションの識別メッセージ・リスナーを戻します。

指定方法

インタフェース javax.jms.Session 内の javax.jms.Session.getMessageListener()

戻り値

セッションに関連付けられているメッセージ・リスナー

例外

JMSException ― JMS プロバイダにおける内部エラーのために、JMS がメッセージ・リスナーの取得に失敗した場合

getQueue(String, String)

```
public synchronized javax.jms.Queue getQueue(java.lang.String owner,
java.lang.String name)
```

既存のキューを取得します。キューは、ユーザーがそのキューの作成者であるか、または指定したキューのエンキュー / デキュー権限を持っている場合のみ戻されます。

パラメータ

owner ― キューの所有者（スキーマ）

name ― キューの名前

例外

JMSException ― エラーのために、キューを戻すことができなかった場合

getQueueTable(String, String)

```
public synchronized oracle.jms.AQQueueTable getQueueTable(java.lang.String owner,
java.lang.String name)
```

既存のキュー表のハンドルを取得します。キュー表の所有者が接続をオープンしたユーザーと異なる場合は、呼出し側にキュー表内のキューまたはトピックの AQ エンキュー / デキュー権限が必要です。権限がない場合、キュー表は戻されません。

パラメータ

owner ― キュー表の所有者（スキーマ）

name ― キュー表の名前

例外

JMSException ― キュー表が存在しない、またはユーザーにそのキュー表内のキューまたはトピックに関する権限がない場合

getTopic(String, String)

```
public synchronized javax.jms.Topic getTopic(java.lang.String owner,
java.lang.String name)
```
既存のトピックを取得します。トピックは、ユーザーがトピックの作成者であるか、指定したトピックのエンキュー / デキュー権限を持っている場合のみ戻されます。

パラメータ

owner ― トピックの所有者（スキーマ）

name ― トピックの名前

例外

JMSException ― エラーのために、トピックを戻すことができなかった場合

getTransacted()

```
public synchronized boolean getTransacted()
```
セッションがトランザクション・モードかどうかをチェックします。

指定方法

インタフェース javax.jms.Session 内の javax.jms.Session.getTransacted()

戻り値

トランザクション・モードの場合は true

例外

JMSException ― セッションがクローズされている場合

grantSystemPrivilege(String, String, boolean)

```
public void grantSystemPrivilege(java.lang.String privilege,
java.lang.String grantee, boolean admin_option)
```
AQ システム権限をユーザーまたはロールに付与します。初期状態では、SYS と SYSTEM のみがこのプロシージャを正常に使用できます。

パラメータ

privilege ― オプションは、ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY です。

ENQUEUE_ANY ― この権限を持つユーザーは、データベース内の任意のキュー / トピックにメッセージをエンキューできます。

DEQUEUE_ANY ― この権限を持つユーザーは、データベース内の任意のキュー / トピックからメッセージをデキューできます。

MANAGE_ANY ― この権限を持つユーザーは、データベース内の任意のキュー / トピックにアクセスして、管理コールを作成できます。

grantee ― 権限受領者を指定します。権限受領者には、ユーザー、ロールまたは PUBLIC ロールを指定できます。

admin_option ― この値が true に設定された場合、権限受領者は、このプロシージャを使用して、他のユーザーまたはロールにシステム権限を付与できます。

例外

JMSException ― システム権限を付与できなかった場合

recover()

```
public synchronized void recover()
```

このセッションのメッセージ配信を停止し、最も古い無応答メッセージからメッセージ配信を再度開始します。

すべてのコンシューマがシリアル順にメッセージを配信します。受信した 1 つのメッセージに応答することによって、クライアントに配信されたすべてのメッセージに自動的に応答します。

セッションを再起動すると、次の処理が行われます。

- メッセージ配信を停止します。
- 配信したが、応答がないすべてのメッセージに "redelivered" のマークを付けます。
- 前に配信したが応答のないすべてのメッセージを含めて、一連の配信を再度開始します。再配信するメッセージは、元の正確な配信順序で配信する必要はありません。

指定方法

インタフェース javax.jms.Session 内の javax.jms.Session.recover()

例外

JMSException ―内部エラーのために、JMS 実装がメッセージ配信の停止および再起動に失敗した場合

IllegalStateException ― 処理済みのセッションからメソッドがコールされた場合

revokeSystemPrivilege(String, String)

```
public void revokeSystemPrivilege(java.lang.String privilege,
java.lang.String grantee)
```
ユーザーまたはロールから AQ システム権限を取り消します。

パラメータ

`privilege` ― オプションは、`ENQUEUE_ANY`、`DEQUEUE_ANY` および `MANAGE_ANY` です。

`grantee` ― 権限受領者を指定します。権限受領者には、ユーザー、ロールまたは PUBLIC ロールを指定できます。

例外

`JMSException` ― システム権限を取り消すことができなかった場合

rollback()

```
public synchronized void rollback()
```
このトランザクションで実行したすべてのメッセージをロールバックし、現在保持しているロックを解放します。

指定方法

インタフェース javax.jms.Session 内の javax.jms.Session.rollback()

例外

`JMSException` ― 内部エラーのために、JMS 実装がトランザクションのロールバックに失敗した場合

run()

```
public void run()
```

指定方法

インタフェース java.lang.Runnable 内の java.lang.Runnable.run()

setMessageListener(MessageListener)

```
public synchronized void setMessageListener(javax.jms.MessageListener listener)
```

セッションの識別メッセージ・リスナーを設定します。これが設定されると、そのセッション内で他の形式のメッセージ受信は使用できません。ただし、メッセージ送信の形式はすべて今までどおりサポートされます。

指定方法

インタフェース javax.jms.Session 内の
javax.jms.Session.setMessageListener(javax.jms.MessageListener)

パラメータ

`listener` ― このセッションと関連付けるメッセージ・リスナー

例外

`JMSException` ― JMS プロバイダにおける内部エラーのために、JMS がメッセージ・リスナーの設定に失敗した場合

unsubscribe(String)

```
public synchronized void unsubscribe(String subs_name)
```

クライアントが作成したサブスクライバで、指定したサブスクライバ名を持つ永続的なサブスクリプションを非サブスクライブ化します。

パラメータ

`subs_name` ― 非サブスクライブ化する必要がある永続サブスクライバの名前

例外

`JMSException` ― JMS エラーのために、JMS が永続的なサブスクリプションの非サブスクライブ化に失敗した場合

unsubscribe(Topic, AQjmsAgent)

```
public synchronized void unsubscribe(javax.jms.Topic topic,
AQjmsAgent remote_subscriber)
```
指定したトピックにクライアントが作成した永続的なリモート・サブスクリプションを非サブスクライブ化します。

パラメータ

`topic` ― サブスクライブするトピック

`remote_subscriber` ― リモート・サブスクライバを表す AQjmsAgent。AQjmsAgent のアドレス・フィールドは null にできません。

例外

`JMSException` ― JMS エラーのために、JMS が永続的なサブスクリプションの非サブスクライブ化に失敗した場合

`InvalidDestinationException` ― 無効なトピックが指定された場合

unsubscribe(Topic, String)

```
public synchronized void unsubscribe(javax.jms.Topic topic,
java.lang.String subs_name)
```
指定したトピックにクライアントが作成した永続的なサブスクリプションを非サブスクライブ化します。

パラメータ

`topic` ― サブスクライブするトピック

`subs_name` ― このサブスクリプションの識別に使用する名前

例外

`JMSException` ― JMS エラーのために、JMS が永続的なサブスクリプションの非サブスクライブ化に失敗した場合

`InvalidDestinationException` ― 無効なトピックが指定された場合

AQjmsStreamMessage

構文

```
public class AQjmsStreamMessage extends AQjmsMessage
   implements javax.jms.StreamMessage

java.lang.Object
  |
  +--AQjmsMessage
        |
        +--oracle.jms.AQjmsStreamMessage
```

実装される全インタフェース

```
javax.jms.Message, javax.jms.StreamMessage
```

説明

このクラスは、StreamMessage インタフェースを実装します。StreamMessage は、Java プリミティブ型のストリームを送信するために使用します。

メンバーの概要	説明
メソッド	-
clearBody()	メッセージ本体を消去します。
clearProperties()	-
readBoolean()	ストリーム・メッセージから boolean 型の値を読み込みます。
readByte()	ストリーム・メッセージから 8 ビットの符号付きの値を読み込みます。
readBytes(byte[])	ストリーム・メッセージからバイト配列を読み込みます。
readChar()	ストリーム・メッセージから Unicode 文字の値を読み込みます。
readDouble()	ストリーム・メッセージから double 型の値を読み込みます。
readFloat()	ストリーム・メッセージから float 型の値を読み込みます。
readInt()	ストリーム・メッセージから 32 ビットの符号付き整数を読み込みます。

メンバーの概要（続き）	説明
readLong()	ストリーム・メッセージから 64 ビットの符号付き整数を読み込みます。
readObject()	ストリーム・メッセージから Java オブジェクトを読み込みます。
readShort()	ストリーム・メッセージから 16 ビットの符号付きの値を読み込みます。
readString()	ストリーム・メッセージから文字列を読み込みます。
reset()	メッセージを読取り専用モードにして、読み出し位置をストリームの先頭に設定し直します。
writeBoolean(boolean)	boolean 型の値を、1 バイトの値でストリーム・メッセージに書き込みます。
writeByte(byte)	byte 型の値を、1 バイトの値でストリーム・メッセージに書き込みます。
writeBytes(byte[])	ストリーム・メッセージにバイト配列を書き込みます。
writeBytes(byte[], int, int)	ストリーム・メッセージにバイト配列の一部を書き込みます。
writeChar(char)	char 型の値を、2 バイトで上位バイトから先にストリームに書き込みます。
writeDouble(double)	double 型の値を、8 バイトで上位バイトから先にストリームに書き込みます。
writeFloat(float)	float 型の値を、4 バイトで上位バイトから先にストリームに書き込みます。
writeInt(int)	int 型の値を、4 バイトで上位バイトから先にストリームに書き込みます。
writeLong(long)	long 型の値を、8 バイトで上位バイトから先にストリームに書き込みます。
writeObject(Object)	ストリーム・メッセージに Java オブジェクトを書き込みます。
writeShort(short)	short 型の値を、2 バイトで上位バイトから先にストリームに書き込みます。
writeString(String)	出力ストリームに文字列を書き込みます。

継承メンバーの概要

インタフェース javax.jms.Message から継承されるフィールド

DEFAULT_DELIVERY_MODE、DEFAULT_PRIORITY、DEFAULT_TIME_TO_LIVE

クラス AQjmsMessage から継承されるメソッド

getBooleanProperty(String)、getByteProperty(String)、
getDoubleProperty(String)、getFloatProperty(String)、
getIntProperty(String)、getJMSCorrelationID()、
getJMSCorrelationIDAsBytes()、getJMSDeliveryMode()、getJMSDestination()、
getJMSExpiration()、getJMSMessageID()、getJMSMessageIDAsBytes()、
getJMSPriority()、getJMSRedelivered()、getJMSReplyTo()、
getJMSTimestamp()、getJMSType()、getLongProperty(String)、
getObjectProperty(String)、getPropertyNames()、getSenderID()、
getShortProperty(String)、getStringProperty(String)、
propertyExists(String)、setBooleanProperty(String, boolean)、
setByteProperty(String, byte)、setDoubleProperty(String, double)、
setFloatProperty(String, float)、setIntProperty(String, int)、
setJMSCorrelationID(String)、setJMSCorrelationIDAsBytes(byte[])、
setJMSDestination(Destination)、setJMSExpiration(long)、
setJMSMessageID(String)、setJMSPriority(int)、
setJMSRedelivered(boolean)、setJMSReplyTo(Destination)、
setJMSTimestamp(long)、setJMSType(String)、setLongProperty(String,
long)、setObjectProperty(String, Object)、setSenderID(AQjmsAgent)、
setShortProperty(String, short)、setStringProperty(String, String)

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、
wait、wait、wait

インタフェース javax.jms.Message から継承されるメソッド

getBooleanProperty、getByteProperty、getDoubleProperty、
getFloatProperty、getIntProperty、getJMSCorrelationID、
getJMSCorrelationIDAsBytes、getJMSDeliveryMode、getJMSDestination、
getJMSExpiration、getJMSMessageID、getJMSPriority、getJMSRedelivered、
getJMSReplyTo、getJMSTimestamp、getJMSType、getLongProperty、
getObjectProperty、getPropertyNames、getShortProperty、
getStringProperty、propertyExists、setBooleanProperty、setByteProperty、
setDoubleProperty、setFloatProperty、setIntProperty、
setJMSCorrelationID、setJMSCorrelationIDAsBytes、setJMSDeliveryMode、
setJMSDestination、setJMSExpiration、setJMSMessageID、setJMSPriority、
setJMSRedelivered、setJMSReplyTo、setJMSTimestamp、setJMSType、
setLongProperty、setObjectProperty、setShortProperty、setStringProperty

メソッド

clearBody()

```
public void clearBody()
```
メッセージ本体を消去します。メッセージの他の部分はすべてそのまま残ります。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearBody()

オーバーライド
クラス AQjmsMessage 内の clearBody()

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージ本体の消去に失敗した場合

clearProperties()

```
public void clearProperties()
```

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearProperties()

オーバーライド
クラス AQjmsMessage 内の clearProperties()

readBoolean()

```
public boolean readBoolean()
```
ストリーム・メッセージから boolean 型の値を読み込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.readBoolean()

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

readByte()

```
public byte readByte()
```
ストリーム・メッセージから 8 ビットの符号付きの値を読み込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.readByte()

戻り値
ストリーム・メッセージから、8 ビットの符号付きの byte 型として次の 1 バイト

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

readBytes(byte[])

```
public int readBytes(byte[] value)
```
ストリーム・メッセージからバイト配列を読み込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.readBytes(byte[])

パラメータ
`value` ― データが読み込まれるバッファ

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

readChar()

```
public char readChar()
```
ストリーム・メッセージから Unicode 文字の値を読み込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.readChar()

戻り値
ストリーム・メッセージから、Unicode 文字として次の 2 バイト

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

readDouble()

```
public double readDouble()
```
ストリーム・メッセージから double 型の値を読み込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.readDouble()

戻り値
ストリーム・メッセージから、double 型として解釈された次の 8 バイト

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

readFloat()

```
public float readFloat()
```
ストリーム・メッセージから float 型の値を読み込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.readFloat()

戻り値
ストリーム・メッセージから、float 型として解釈された次の 4 バイト

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

readInt()

```
public int readInt()
```
ストリーム・メッセージから 32 ビットの符号付き整数を読み込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.readInt()

戻り値
ストリーム・メッセージから、int 型として解釈された次の 4 バイト

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

readLong()

```
public long readLong()
```
ストリーム・メッセージから 64 ビットの符号付き整数を読み込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.readLong()

戻り値
ストリーム・メッセージから、long 型として解釈された次の 8 バイト

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

readObject()

```
public java.lang.Object readObject()
```
ストリーム・メッセージから Java オブジェクトを読み込みます。

このメソッドは、`writeObject` メソッド・コールまたは基本書込みメソッドでストリームに書き込まれたオブジェクトを、オブジェクト化された形式で戻すために使用できることに注意してください。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.readObject()

戻り値
ストリーム・メッセージの Java オブジェクトをオブジェクト化された形式で戻します（つまり、int で設定された場合は、Integer が戻されます）。

例外
`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

`MessageNotReadableException` ― メッセージが書込み専用モードの場合

readShort()

```
public short readShort()
```
ストリーム・メッセージから 16 ビットの符号付きの値を読み込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.readShort()

戻り値
ストリーム・メッセージから、16 ビットの数値として解釈された次の 2 バイト

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

readString()

```
public java.lang.String readString()
```
ストリーム・メッセージから文字列を読み込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.readString()

戻り値
ストリーム・メッセージからの文字列

例外
`MessageNotReadableException` ― メッセージが書込み専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

`MessageEOFException` ― メッセージ・ストリームの終わりに達した場合

reset()

```
public void reset()
```
メッセージを読取り専用モードにして、読み出し位置をストリームの先頭に設定し直します。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.reset()

例外
`MessageNotWriteableException` ― メッセージが書込み専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeBoolean(boolean)

```
public void writeBoolean(boolean value)
```
boolean 型の値を、1 バイトの値でストリーム・メッセージに書き込みます。値 true は値 (byte)1 として書き込まれ、値 false は値 (byte)0 として書き込まれます。

指定方法
インタフェース javax.jms.StreamMessage 内の
javax.jms.StreamMessage.writeBoolean(boolean)

パラメータ
`value` ― 書き込まれる boolean 型の値

例外
`MessageNotWritableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeByte(byte)

```
public void writeByte(byte value)
```
byte 型の値を、1 バイトの値でストリーム・メッセージに書き込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.writeByte(byte)

パラメータ

value ― 書き込まれる byte 型の値

例外

MessageNotWritableException ― メッセージが読取り専用モードの場合

JMSException ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeBytes(byte[])

```
public void writeBytes(byte[] value)
```
ストリーム・メッセージにバイト配列を書き込みます。

指定方法

インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.writeBytes(byte[])

パラメータ

value ― 書き込まれるバイト配列

例外

MessageNotWritableException ― メッセージが読取り専用モードの場合

JMSException ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeBytes(byte[], int, int)

```
public void writeBytes(byte[] value, int offset, int length)
```
ストリーム・メッセージにバイト配列の一部を書き込みます。

指定方法

インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.writeBytes(byte[], int, int)

パラメータ

value ― 書き込まれるバイト配列

offset ― バイト配列内の初期オフセット

length ― 使用するバイト数

例外

`MessageNotWritableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeChar(char)

```
public void writeChar(char value)
```
char 型の値を、2 バイトで上位バイトから先にストリームに書き込みます。

指定方法

インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.writeChar(char)

パラメータ

`value` ― 書き込まれる char 型の値

例外

`MessageNotWritableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeDouble(double)

```
public void writeDouble(double value)
```
double 型の値を、8 バイトで上位バイトから先にストリームに書き込みます。

指定方法

インタフェース javax.jms.StreamMessage 内の
javax.jms.StreamMessage.writeDouble(double)

パラメータ

`value` ― 書き込まれる double 型の値

例外

`MessageNotWritableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeFloat(float)

```
public void writeFloat(float value)
```
float 型の値を、4 バイトで上位バイトから先にストリームに書き込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.writeFloat(float)

パラメータ
`value` ― 書き込まれる float 型の値

例外
`MessageNotWritableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeInt(int)

```
public void writeInt(int value)
```
int 型の値を、4 バイトで上位バイトから先にストリームに書き込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.writeInt(int)

パラメータ
`value` ― 書き込まれる int 型の値

例外
`MessageNotWritableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeLong(long)

```
public void writeLong(long value)
```
long 型の値を、8 バイトで上位バイトから先にストリームに書き込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.writeLong(long)

パラメータ
`value` ― 書き込まれる long 型の値

例外

`MessageNotWritableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeObject(Object)

```
public void writeObject(java.lang.Object value)
```
ストリーム・メッセージに Java オブジェクトを書き込みます。

指定方法

インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.writeObject(java.lang.Object)

パラメータ

`value` ― 書き込まれる Java オブジェクト

例外

`MessageNotWritableException` ― メッセージが読取り専用モードの場合

`MessageFormatException` ― オブジェクトの型が無効な場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeShort(short)

```
public void writeShort(short value)
```
short 型の値を、2 バイトで上位バイトから先にストリームに書き込みます。

指定方法

インタフェース javax.jms.StreamMessage 内の javax.jms.StreamMessage.writeShort(short)

パラメータ

`value` ― 書き込まれる short 型の値

例外

`MessageNotWritableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

writeString(String)

```
public void writeString(java.lang.String value)
```
出力ストリームに文字列を書き込みます。

指定方法
インタフェース javax.jms.StreamMessage 内の
javax.jms.StreamMessage.writeString(java.lang.String)

パラメータ
`value` ― 書き込まれる文字列

例外
`MessageNotWritableException` ― メッセージが読取り専用モードの場合

`JMSException` ― 内部 JMS エラーのために、JMS がメッセージの読込みに失敗した場合

AQjmsTextMessage

構文

```
public class AQjmsTextMessage extends AQjmsMessage
   implements javax.jms.TextMessage

java.lang.Object
  |
  +--AQjmsMessage
        |
        +--oracle.jms.AQjmsTextMessage
```

実装される全インタフェース

javax.jms.Message, javax.jms.TextMessage

説明

このクラスは、TextMessage インタフェースを実装します。TextMessage は、java.lang.StringBuffer を含んだメッセージを送信するために使用します。

メンバーの概要	説明
メソッド	-
clearBody()	本体を消去します。
clearProperties()	メッセージのプロパティを消去します。
getText()	このメッセージのデータを含んだ文字列を取得します。
setText(String)	このメッセージのデータを含んだ文字列を設定します。

継承メンバーの概要

インタフェース javax.jms.Message から継承されるフィールド

DEFAULT_DELIVERY_MODE、DEFAULT_PRIORITY、DEFAULT_TIME_TO_LIVE

クラス AQjmsMessage から継承されるメソッド

継承メンバーの概要

getBooleanProperty(String)、getByteProperty(String)、
getDoubleProperty(String)、getFloatProperty(String)、
getIntProperty(String)、getJMSCorrelationID()、
getJMSCorrelationIDAsBytes()、getJMSDeliveryMode()、getJMSDestination()、
getJMSExpiration()、getJMSMessageID()、getJMSMessageIDAsBytes()、
getJMSPriority()、getJMSRedelivered()、getJMSReplyTo()、
getJMSTimestamp()、getJMSType()、getLongProperty(String)、
getObjectProperty(String)、getPropertyNames()、getSenderID()、
getShortProperty(String)、getStringProperty(String)、
propertyExists(String)、setBooleanProperty(String, boolean)、
setByteProperty(String, byte)、setDoubleProperty(String, double)、
setFloatProperty(String, float)、setIntProperty(String, int)、
setJMSCorrelationID(String)、setJMSCorrelationIDAsBytes(byte[])、
setJMSDestination(Destination)、setJMSExpiration(long)、
setJMSMessageID(String)、setJMSPriority(int)、
setJMSRedelivered(boolean)、setJMSReplyTo(Destination)、
setJMSTimestamp(long)、setJMSType(String)、setLongProperty(String,
long)、setObjectProperty(String, Object)、setSenderID(AQjmsAgent)、
setShortProperty(String, short)、setStringProperty(String, String)

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、
wait

インタフェース javax.jms.Message から継承されるメソッド

getBooleanProperty、getByteProperty、getDoubleProperty、
getFloatProperty、getIntProperty、getJMSCorrelationID、
getJMSCorrelationIDAsBytes、getJMSDeliveryMode、getJMSDestination、
getJMSExpiration、getJMSMessageID、getJMSPriority、getJMSRedelivered、
getJMSReplyTo、getJMSTimestamp、getJMSType、getLongProperty、
getObjectProperty、getPropertyNames、getShortProperty、
getStringProperty、propertyExists、setBooleanProperty、setByteProperty、
setDoubleProperty、setFloatProperty、setIntProperty、
setJMSCorrelationID、setJMSCorrelationIDAsBytes、setJMSDeliveryMode、
setJMSDestination、setJMSExpiration、setJMSMessageID、setJMSPriority、
setJMSRedelivered、setJMSReplyTo、setJMSTimestamp、setJMSType、
setLongProperty、setObjectProperty、setShortProperty、setStringProperty

メソッド

clearBody()

```
public void clearBody()
```

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearBody()

オーバーライド
クラス AQjmsMessage 内の clearBody()

clearProperties()

```
public void clearProperties()
```
メッセージのプロパティを消去します。

指定方法
インタフェース javax.jms.Message 内の javax.jms.Message.clearProperties()

オーバーライド
クラス AQjmsMessage 内の clearProperties()

例外
`JMSException` ― 内部 JMS エラーのために、JMS が JMS メッセージ・プロパティの消去に失敗した場合

getText()

```
public java.lang.String getText()
```
このメッセージのデータを含んだ文字列を取得します。デフォルト値は null です。

指定方法
インタフェース javax.jms.TextMessage 内の javax.jms.TextMessage.getText()

戻り値
メッセージのデータを含んだ文字列

例外
`JMSException` ― 内部 JMS エラーのために、JMS がテキストの取得に失敗した場合

setText(String)

```
public void setText(java.lang.String string)
```
このメッセージのデータを含んだ文字列を設定します。

指定方法
インタフェース javax.jms.TextMessage 内の javax.jms.TextMessage.setText(java.lang.String)

パラメータ
`string` ― メッセージのデータを含んだ文字列

例外
`JMSException` ― 内部 JMS エラーのために、JMS がテキストの設定に失敗した場合

`MessageNotWriteableException` ― メッセージが読取り専用モードの場合

AQjmsTopicBrowser

```
java.lang.Object
    |
    + -- oracle.jms.AQjmsTopicBrowser
```

説明

クライアントは、AQjmsTopicBrowser のインスタンスを使用して、トピック上のメッセージを削除せずに参照します。この実装は、JMS に対する Oracle 固有の拡張機能です。

メンバーの概要	説明
メソッド	-
getTopic()	このトピック・ブラウザに関連付けられているトピックを取得します。
getEnumeration()	現行のトピック・メッセージを受信順にブラウズするための一覧を取得します。
getMessageSelector()	このトピック・ブラウザのメッセージの選択指定式を取得します。
nextElement()	この一覧の次の要素を戻します。
hasMoreElements()	この一覧にこれ以上の要素が含まれているかどうかをチェックします。
purgeSeen()	参照中に、その時点で参照済みのメッセージをパージします。
setTransformation(String transformation)	このブラウザの変換を設定します。
getTransformation()	このコンシューマの変換を取得します。

メソッド

close()

```
public void close()
```

トピック・ブラウザをクローズします。OJMS は、JVM の外部で TopicBrowser 用にリソースを割り当てている場合があります。このため、クライアントでは、不要なリソースをクローズしておく必要があります。これらのリソースは最終的にガベージ・コレクションで再生されますが、適切な時期に行われない可能性があります。

操作中のエラーはすべて、通知されないまま無視されます。

getTopic()

```
public Topic getTopic()
```
このトピック・ブラウザに関連付けられているトピックを取得します。

戻り値
このトピック・ブラウザに関連付けられているトピック

例外
JMSException − JMS エラーのために、JMS がこのブラウザに関連付けられているトピックの取得に失敗した場合

getEnumeration()

```
public Enumeration getEnumeration()
```
現行のトピック・メッセージを受信順にブラウズするための一覧を取得します。
getEnumeration() が同じ TopicBrowser で 2 回コールされると、同じ一覧オブジェクトが戻されます。したがって、1 つの一覧オブジェクトでの nextElement() コールによって、2 番目の一覧オブジェクトの状態も変更されます。

戻り値
メッセージをブラウズするための一覧

getMessageSelector()

```
public String getMessageSelector()
```
このトピック・ブラウザのメッセージの選択指定式を取得します。

戻り値
このトピック・ブラウザのメッセージ・セレクタ

例外
JMSException − JMS エラーのために、JMS がメッセージ・セレクタの取得に失敗した場合

nextElement()

```
public Object nextElement()
```
この一覧の次の要素を戻します。キャッシュ・メッセージを使用しようとします（以前の hasMoreElements() のコールから使用できる場合）。ブラウザのセレクタがメッセージ ID を使用した場合、参照時に戻すことができるメッセージは 1 つのみです。

戻り値
この一覧の次の要素

例外
`NoSuchElementException` ― 要素がこれ以上存在しない場合

hasMoreElements()

```
public boolean hasMoreElements()
```
この一覧にこれ以上の要素が含まれているかどうかをチェックします。

戻り値
一覧に要素がまだ存在している場合は true、存在していない場合は false

purgeSeen()

```
public void purgeSeen()
```
参照中に、その時点で参照済みのメッセージをパージします。メッセージは、参照中に nextElement() のコールによってクライアントに戻された場合、参照済みとみなされます。したがって、クライアントはトピック・ブラウザを作成し、パージを即時コールし、トピックの状態を変更しないようにできます（このメソッドを指定することでメッセージは参照されないため）。

- トピックのパージは、クライアントが参照中に、まだ参照していないメッセージの状態に影響を与えることもありません。
- パージは、LOCKED モードで作成されたトピック・ブラウザでのみサポートされます。LOCKED モードで作成されていないトピックをパージしようとすると、例外が発生します。
- パージ操作は、このトピック・ブラウザに対するセッションがコミットされる場合にのみ有効になります。セッションがロールバックされると、パージ操作は元に戻され、メッセージは再度参照できるようになります。
- パージが完了するのは、トピック・ブラウザに対するセッションがコミットされるときのみです。

例外

JMSException － パージ操作時に JMS エラーが発生した場合

setTransformation(String transformation)

```
public void setTransformation(String transformation)
```

このブラウザの変換を設定します。メッセージをユーザーに戻す前に、変換が適用されます。

パラメータ

transformation － 変換名

getTransformation()

```
public String getTransformation()
```

このコンシューマの変換を取得します。

戻り値

コンシューマの変換

AQjmsIllegalStateException

構文

```
    public class AQjmsIllegalStateException
       extends javax.jms.IllegalStateException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--javax.jms.JMSException
                    |
                    +--javax.jms.IllegalStateException
                        |
                      +--oracle.jms.AQjmsIllegalStateException
```

実装される全インタフェース

java.io.Serializable

説明

この例外は、IlegalStateException を拡張します。メソッドが無効な時間または不適切な時間に起動された場合、または要求された操作に対して OJMS が適切な状態ではない場合に発生します。たとえば、Session.commit が未処理のセッションでコールされると例外が発生します。

継承メンバーの概要

インタフェース javax.jms.JMSException から継承されるメソッド

getErrorCode、getLinkedException、setLinkedException

クラス java.lang.Throwable から継承されるメソッド

fillInStackTrace、getLocalizedMessage、getMessage、printStackTrace、toString

クラス java.lang.Object から継承されるメソッド

clone、equals、finalize、getClass、hashCode、notify、notifyAll、wait

AQjmsTopicConnectionFactory

構文

```
public class AQjmsTopicConnectionFactory extends java.lang.Object
   implements javax.jms.TopicConnectionFactory

java.lang.Object
  |
  +--oracle.jms.AQjmsTopicConnectionFactory
```

実装される全インタフェース

javax.jms.ConnectionFactory, javax.jms.TopicConnectionFactory,
java.lang.Referenceable, java.lang.Serializable

説明

このクラスは、TopicConnectionFactory インタフェースを実装します。
TopicConnectionFactory は、トピック・コネクションを作成するために使用します。

メンバーの概要	説明
メソッド	-
createTopicConnection()	このトピック・コネクション・ファクトリのホストである JMS Server に対するトピック・コネクションを作成します。
createTopicConnection(Connection)	すでにオープンしている JDBC 接続を使用してトピック・コネクションを作成します。
createTopicConnection(String, String)	接続の作成時の認証にユーザー名とパスワードを使用して、トピック・コネクションを作成します。

継承メンバーの概要
クラス java.lang.Object から継承されるメソッド
clone、equals、finalize、getClass、hashCode、notify、notifyAll、toString、wait

メソッド

createTopicConnection()

```
public javax.jms.TopicConnection createTopicConnection()
```
このトピック・コネクション・ファクトリのホストである JMS Server に対するトピック・コネクションを作成します。

指定方法
インタフェース javax.jms.TopicConnectionFactory 内の
javax.jms.TopicConnectionFactory.createTopicConnection()

戻り値
トピック・コネクション

例外
`JMSException` ― JMS エラーのために、JMS がトピック・コネクションの作成に失敗した場合

createTopicConnection(Connection)

```
public static javax.jms.TopicConnection createTopicConnection(java.sql.Connection
jdbc_connection)
```
すでにオープンしている JDBC 接続を使用してトピック・コネクションを作成します。この作成方法では、データベースに対して別の接続が作成されることはありません。かわりに、JMS は指定されたデータベースへの接続をバインドし、JMS で定義されている Pub/Sub 機能へのインタフェースを提供します。

パラメータ
`jdbc_connection` ― データベースに対する有効なオープン接続

戻り値
トピック・コネクション

例外
`JMSException` ― JMS エラーのために、JMS がトピック・コネクションの作成に失敗した場合

createTopicConnection(String, String)

```
public javax.jms.TopicConnection createTopicConnection(java.lang.String username,
java.lang.String password)
```

接続の作成時の認証にユーザー名とパスワードを使用して、トピック・コネクションを作成します。

指定方法

インタフェース javax.jms.TopicConnectionFactory 内の javax.jms.TopicConnectionFactory.createTopicConnection(java.lang.String, java.lang.String)

パラメータ

`username` − キューイングするために DB に接続するユーザーの名前

`password` − DB に接続するユーザーのパスワード

戻り値

トピック・コネクション

例外

`JMSException` − JMS エラーのために、JMS がトピック・コネクションの作成に失敗した場合

AQjmsTopicPublisher

構文

```
public interface AQjmsTopicPublisher extends javax.jms.TopicPublisher
```

全スーパーインタフェース

```
javax.jms.MessageProducer, javax.jms.TopicPublisher
```

既知の全実装クラス

```
AQjmsProducer
```

説明

このインタフェースは TopicPublisher を拡張し、JMS に対する AQ 拡張を定義します。クライアントは、TopicPublisher を使用して、トピックにメッセージを公開します。

メンバーの概要	説明
メソッド	-
getTransformation()	このパブリッシャの変換を取得します。
publish(Message, AQjmsAgent[])	特定のレシピエントのリストにメッセージを公開します。
publish(Message, AQjmsAgent[], int, int, long)	レシピエントのリスト、配信モード、優先順位および TTL を指定して、トピックにメッセージを公開します。
publish(Topic, Message, AQjmsAgent[])	レシピエントのリストを指定してトピックにメッセージを公開します。
publish(Topic, Message, AQjmsAgent[], int, int, long)	レシピエントのリスト、配信モード、優先順位および TTL を指定して、トピックにメッセージを公開します。
setTransformation(String)	このパブリッシャの変換を設定します。

継承メンバーの概要

インタフェース javax.jms.TopicPublisher から継承されるメソッド

getTopic、publish

継承メンバーの概要（続き）

インタフェース javax.jms.MessageProducer から継承されるメソッド

close、getDeliveryMode、getDisableMessageID、getDisableMessageTimestamp、getPriority、getTimeToLive、setDeliveryMode、setDisableMessageID、setDisableMessageTimestamp、setPriority、setTimeToLive

メソッド

getTransformation()

```
public String getTransformation()
```
このパブリッシャの変換を取得します。

戻り値
変換

例外
JMSException ― 変換の取得時にエラーが発生した場合

publish(Message, AQjmsAgent[])

```
public void publish(javax.jms.Message message, AQjmsAgent recipient_list)
```
特定のレシピエントのリストにメッセージを公開します。

パラメータ
message ― 公開するメッセージ

recipient_list ― メッセージの公開先のレシピエントのリスト。レシピエントのタイプは AQjmsAgent です。

例外
JMSException ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

publish(Message, AQjmsAgent[], int, int, long)

```
public void publish(javax.jms.Message message, AQjmsAgent recipient_list,
int deliveryMode, int priority, long timeToLive)
```
レシピエントのリスト、配信モード、優先順位および TTL を指定して、トピックにメッセージを公開します。

パラメータ

`message` ― 公開するメッセージ

`recipient_list` ― メッセージの公開先のレシピエントのリスト。レシピエントのタイプは AQjmsAgent です。

`deliveryMode` ― 配信モード。persistent または non_persistent。

`priority` ― メッセージの優先順位

`timeToLive` ― メッセージの TTL（ミリ秒）。0（ゼロ）は無制限です。

例外

`JMSException` ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

publish(Topic, Message, AQjmsAgent[])

```
public void publish(javax.jms.Topic topic, javax.jms.Message message,
AQjmsAgent recipient_list)
```
レシピエントのリストを指定してトピックにメッセージを公開します。

パラメータ

`topic` ― メッセージを公開するトピック。この指定は、メッセージ・プロデューサのデフォルトのトピックをオーバーライドします。

`message` ― 公開するメッセージ

`recipient_list` ― メッセージの公開先のレシピエントのリスト。レシピエントのタイプは AQjmsAgent です。

例外

`JMSException` ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

publish(Topic, Message, AQjmsAgent[], int, int, long)

```
public void publish(javax.jms.Topic topic, javax.jms.Message message,
AQjmsAgent recipient_list, int deliveryMode, int priority, long timeToLive)
```

レシピエントのリスト、配信モード、優先順位および TTL を指定して、トピックにメッセージを公開します。

パラメータ

`topic` ― メッセージを公開するトピック。この指定は、メッセージ・プロデューサのデフォルトのトピックをオーバーライドします。

`message` ― 公開するメッセージ

`recipient_list` ― メッセージの公開先のレシピエントのリスト。レシピエントのタイプは AQjmsAgent です。

`deliveryMode` ― 配信モード。persistent または non_persistent。

`priority` ― メッセージの優先順位

`timeToLive` ― メッセージの TTL（ミリ秒）。0（ゼロ）は無制限です。

例外

`JMSException` ― 内部エラーのために、JMS がメッセージの公開に失敗した場合

setTransformation(String)

```
public void setTransformation(String transformation)
```

このセンダーの変換を設定します。メッセージをトピックに公開する前に、この変換が適用されます。

パラメータ

`transformation` ― メッセージを公開する前に適用する変換

例外

`JMSException` ― 変換の設定時にエラーが発生した場合

AQjmsTopicReceiver

構文

```
public interface AQjmsTopicReceiver extends TopicReceiver
```

全スーパーインタフェース

```
javax.jms.MessageConsumer, TopicReceiver
```

既知の全実装クラス

```
AQjmsConsumer
```

説明

このインタフェースは TopicReceiver インタフェースを拡張し、(Point-to-Multipoint 通信において) リモートのサブスクライバと明示的に指定したレシピエントに対する AQ 拡張を定義します。TopicReceiver を使用して、トピックからメッセージを受信します。

メンバーの概要	説明
メソッド	-
getNavigationMode()	メッセージの受信に使用されるナビゲーション・モードを取得します。
getTransformation()	このレシーバの変換を取得します。
receiveNoData()	メッセージを、ユーザーに戻さずに使用します。
receiveNoData(long)	メッセージを、ユーザーに戻さずに使用します。
setNavigationMode(int)	メッセージの受信に使用されるナビゲーション・モードを設定します。
setTransformation(String)	このレシーバの変換を設定します。

継承メンバーの概要

インタフェース TopicReceiver から継承されるメソッド

```
getTopic()
```

インタフェース javax.jms.MessageConsumer から継承されるメソッド

```
close、getMessageListener、getMessageSelector、receive、receiveNoWait、
setMessageListener
```

メソッド

getNavigationMode()

```
public int getNavigationMode()
```
メッセージの受信に使用されるナビゲーション・モードを取得します。

戻り値
ナビゲーション・モード

例外
`JMSException` ― ナビゲーション・モードの取得時にエラーが発生した場合

getTransformation()

```
public String getTransformation()
```
このレシーバの変換を取得します。

戻り値
変換

例外
`JMSException` ― 変換の取得時にエラーが発生した場合

receiveNoData()

```
public void receiveNoData()
```
メッセージを、ユーザーに戻さずに使用します。このコールは、データベースからメッセージをフェッチするときのオーバーヘッドを回避します。このため、JMS クライアントが、キュー・ブラウザを使用するなどしてメッセージをすでに取得した場合に、最適化するために使用できます。

例外
`JMSException` ― エラーのためにメッセージの受信ができなかった場合

receiveNoData(long)

```
public void receiveNoData(long tomeOut)
```
メッセージを、ユーザーに戻さずに使用します。このコールは、データベースからメッセージをフェッチするときのオーバーヘッドを回避します。このため、JMS クライアントが、キュー・ブラウザを使用するなどしてメッセージをすでに取得した場合に、最適化するために使用できます。このコールは、メッセージが到着するまで、またはタイムアウトになるまでブロックします。

パラメータ

`timeout` ― タイムアウト値（ミリ秒）

例外

`JMSException` ― エラーのためにメッセージの受信ができなかった場合

setTransformation(String)

```
public void setTransformation(String transformation)
```
このレシーバの変換を設定します。メッセージをユーザーに戻す前に、変換が適用されます。

パラメータ

`transformation` ― メッセージを戻す前に適用する変換

例外

`JMSException` ― 変換の設定時にエラーが発生した場合

setNavigationMode(int)

```
public void setNavigationMode(int mode)
```
メッセージの受信に使用されるナビゲーション・モードを設定します。

パラメータ

`mode` ― ナビゲーション・モードの新しい値

例外

`JMSException` ― ナビゲーション・モードの取得時にエラーが発生した場合

AQjmsTopicSubscriber

構文

```
public interface AQjmsTopicSubscriber extends javax.jms.TopicSubscriber
```

全スーパーインタフェース

```
javax.jms.MessageConsumer, javax.jms.TopicSubscriber
```

既知の全実装クラス

```
AQjmsConsumer
```

説明

このインタフェースは TopicSubscriber を拡張し、JMS に対する AQ 拡張を定義します。クライアントは、TopicSubscriber を使用して、トピックに公開されているメッセージを受信します。

メンバーの概要	説明
メソッド	-
getNavigationMode()	メッセージの受信に使用されるナビゲーション・モードを取得します。
receiveNoData()	メッセージを、ユーザーに戻さずに使用します。
receiveNoData(long)	メッセージを、ユーザーに戻さずに使用します。
setNavigationMode(int)	メッセージの受信に使用されるナビゲーション・モードを設定します。

継承メンバーの概要

インタフェース javax.jms.TopicSubscriber から継承されるメソッド

getNoLocal、getTopic

インタフェース javax.jms.MessageConsumer から継承されるメソッド

close、getMessageListener、getMessageSelector、receive、receiveNoWait、setMessageListener

メソッド

getNavigationMode()

```
public int getNavigationMode()
```
メッセージの受信に使用されるナビゲーション・モードを取得します。

例外

`JMSException` ― エラーのためにメッセージの受信ができなかった場合

receiveNoData()

```
public void receiveNoData()
```
メッセージを、ユーザーに戻さずに使用します。このコールは、データベースからメッセージをフェッチするときのオーバーヘッドを回避します。このため、JMS クライアントが、キュー・ブラウザを使用するなどしてメッセージをすでに取得した場合に、最適化するために使用できます。

例外

`JMSException` ― エラーのためにメッセージの受信ができなかった場合

receiveNoData(long)

```
public void receiveNoData(long tomeOut)
```
メッセージを、ユーザーに戻さずに使用します。このコールは、データベースからメッセージをフェッチするときのオーバーヘッドを回避します。このため、JMS クライアントが、キュー・ブラウザを使用するなどしてメッセージをすでに取得した場合に、最適化するために使用できます。このコールは、メッセージが到着するまで、またはタイムアウトになるまでブロックします。

パラメータ

`timeout` ― タイムアウト値（ミリ秒）

例外

`JMSException` ― エラーのためにメッセージの受信ができなかった場合

setNavigationMode(int)

```
public void setNavigationMode(int mode)
```
メッセージの受信に使用されるナビゲーション・モードを設定します。

パラメータ

`mode` ― ナビゲーション・モードの新しい値

例外

`JMSException` ― ナビゲーション・モードの取得時にエラーが発生した場合

TopicBrowser

構文

```
public interface TopicBrowser extends javax.jms.MessageConsumer
```

既知の全サブインタフェース

AQjmsTopicBrowser

全スーパーインタフェース

javax.jms.MessageConsumer

説明

このインタフェースは MessageConsumer を拡張します。リモートのサブスクライバは、このインタフェースを使用して、トピック上のメッセージを、そのメッセージを削除せずに参照します。

TopicReceiver

構文

```
public interface TopicReceiver extends javax.jms.MessageConsumer
```

既知の全サブインタフェース

AQjmsTopicReceiver

全スーパーインタフェース

javax.jms.MessageConsumer

説明

このインタフェースは、MessageConsumer を拡張します。（Point-to-Multipoint 通信において）リモートのサブスクライバと明示的に指定したレシピエントは、このインタフェースを使用してメッセージを受信します。

メンバーの概要	説明
メソッド	-
getTopic()	このレシーバに関連付けられているトピックを取得します。

継承メンバーの概要

インタフェース javax.jms.MessageConsumer から継承されるメソッド

close、getMessageListener、getMessageSelector、receive、receiveNoWait、setMessageListener

メソッド

getTopic()

```
public javax.jms.Topic getTopic()
```
このレシーバに関連付けられているトピックを取得します。

戻り値
このサブスクライバのトピック

例外
`JMSException` ― 内部エラーのために、JMS がこのトピック・レシーバに対するトピックの取得に失敗した場合

5

パッケージ oracle.ODCI

この章では、パッケージ oracle.ODCI で提供される Java 言語 ODCI（Oracle Data Cartridge インタフェース）の拡張索引作成機能インタフェースについて説明します。

この章は、次の項で構成されています。

パッケージ oracle.ODCI の説明

Oracle では、リレーショナル・モデルに基づいて順序付けされたデータを効率よく安全に管理し、さらに、オブジェクト・モデルに基づいて編成されたデータをサポートします。オブジェクト型および他の Oracle データベースの機能（ラージ・オブジェクト（LOB）、外部プロシージャ、拡張索引作成機能、問合せの最適化など）を使用すると、データ・カートリッジと呼ばれる再利用可能なサーバーベースのコンポーネントを作成できます。

Oracle Extensibility Architecture のフレームワーク内でのデータ・カートリッジは、Oracle サーバーの機能を拡張するための純粋なオブジェクト指向の機能です。

> **関連項目：** データ・カートリッジの作成と使用に関する詳細は、『Oracle9*i* Data Cartridge Developer's Guide』を参照してください。

ODCI.jar ファイルと CartridgeServices.jar ファイルのインストール

この章で説明する Java クラスを使用するには、`ODCI.jar` ファイルと `CartridgeServices.jar` ファイルを SYS スキーマにインストールする必要があります。

Java オプションをインストールした場合は、ODCI.jar ファイルと CartridgeServices.jar ファイルをインストールする必要があります。Java オプションをインストールしていない場合は、この作業を実行する必要はありません。

ODCI.jar ファイルと CartridgeServices.jar ファイルをインストールするには、コマンドラインから次のコマンドを実行します。

```
loadjava -user sys/PASSWORD -resolve -synonym -grant public
-verbose ORACLE_HOME/vobs/jlib/CartridgeServices.jar

loadjava -user sys/PASSWORD -resolve -synonym -grant public
-verbose ORACLE_HOME/vobs/jlib/ODCI.jar
```

`PASSWORD` の部分を `SYS` のパスワードに置き換え、`ORACLE_HOME` の部分を Oracle ホームのディレクトリに置き換えます。これらのコマンドによってクラスがインストールされ、SYS スキーマにシノニムが作成されます。

パッケージ oracle.ODCI の概要

表 5-1　パッケージ oracle.ODCI の概要

クラス	説明
ODCIArgDesc	引数の説明
ODCIArgDescList	引数の説明のリスト
ODCIArgDescRef	引数説明の参照
ODCIColInfo	列情報
ODCIColInfoList	列情報のリスト
ODCIColInfoRef	列情報の参照
ODCICost	コスト
ODCICostRef	コストの参照
ODCIEnv	環境
ODCIEnvRef	環境の参照
ODCIFuncInfo	機能情報
ODCIFuncInfoRef	機能情報の参照
ODCIIndexCtx	索引コンテキスト
ODCIIndexCtxRef	索引コンテキストの参照
ODCIIndexInfo	索引情報
ODCIQueryInfoRef	問合せ情報の参照
ODCIRidList	Rid リスト
ODCIStatsOptions	統計オプション
ODCIStatsOptionsRef	統計オプションの参照

ODCIArgDesc

```
oracle.ODCI.ODCIArgDesc
public class ODCIArgDesc
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIArgDesc

```
public ODCIArgDesc()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getArgType

```
public java.math.BigDecimal getArgType()
                                 throws java.sql.SQLException
```

setArgType

```
public void setArgType(java.math.BigDecimal ArgType)
               throws java.sql.SQLException
```

getTableName

```
public java.lang.String getTableName()
                               throws java.sql.SQLException
```

setTableName

```
public void setTableName(java.lang.String TableName)
                 throws java.sql.SQLException
```

getTableSchema

```
public java.lang.String getTableSchema()
                                 throws java.sql.SQLException
```

setTableSchema

```
public void setTableSchema(java.lang.String TableSchema)
                   throws java.sql.SQLException
```

getColName

```
public java.lang.String getColName()
                             throws java.sql.SQLException
```

setColName

```
public void setColName(java.lang.String ColName)
               throws java.sql.SQLException
```

getTablePartitionLower

```
public java.lang.String getTablePartitionLower()
                                        throws java.sql.SQLException
```

setTablePartitionLower

```
public void setTablePartitionLower(java.lang.String TablePartitionLower)
                           throws java.sql.SQLException
```

getTablePartitionUpper

```
public java.lang.String getTablePartitionUpper()
                                        throws java.sql.SQLException
```

setTablePartitionUpper

```
public void setTablePartitionUpper(java.lang.String TablePartitionUpper)
                           throws java.sql.SQLException
```

ODCIArgDescList

```
oracle.ODCI.ODCIArgDescList

public class ODCIArgDescList
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIArgDescList

```
public ODCIArgDescList()
```

ODCIArgDescList

```
public ODCIArgDescList(ODCIArgDesc[] a)
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

length

```
public int length()
           throws java.sql.SQLException
```

getBaseType

```
public int getBaseType()
               throws java.sql.SQLException
```

getBaseTypeName

```
public java.lang.String getBaseTypeName()
                                 throws java.sql.SQLException
```

getDescriptor

```
public oracle.sql.ArrayDescriptor getDescriptor()
                                        throws java.sql.SQLException
```

getArray

```
public ODCIArgDesc[] getArray()
                       throws java.sql.SQLException
```

setArray

```
public void setArray(ODCIArgDesc[] a)
              throws java.sql.SQLException
```

getArray

```
public ODCIArgDesc[] getArray(long index,
                              int count)
                       throws java.sql.SQLException
```

setArray

```
public void setArray(ODCIArgDesc[] a,
                     long index)
              throws java.sql.SQLException
```

getElement

```
public ODCIArgDesc getElement(long index)
                      throws java.sql.SQLException
```

setElement

```
public void setElement(ODCIArgDesc a,
                       long index)
               throws java.sql.SQLException
```

ODCIArgDescRef

```
oracle.ODCI.ODCIArgDescRef

public class ODCIArgDescRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIArgDescRef

```
public ODCIArgDescRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCIArgDesc getValue()
                    throws java.sql.SQLException
```

setValue

```
public void setValue(ODCIArgDesc c)
       throws java.sql.SQLException
```

ODCIColInfo

```
oracle.ODCI.ODCIColInfo
public class ODCIColInfo
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIColInfo

```
public ODCIColInfo()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getTableSchema

```
public java.lang.String getTableSchema()
                                throws java.sql.SQLException
```

setTableSchema

```
public void setTableSchema(java.lang.String TableSchema)
                    throws java.sql.SQLException
```

getTableName

```
public java.lang.String getTableName()
                               throws java.sql.SQLException
```

setTableName

```
public void setTableName(java.lang.String TableName)
                  throws java.sql.SQLException
```

getColName

```
public java.lang.String getColName()
                             throws java.sql.SQLException
```

setColName

```
public void setColName(java.lang.String ColName)
                throws java.sql.SQLException
```

getColTypeName

```
public java.lang.String getColTypeName()
                                throws java.sql.SQLException
```

setColTypeName

```
public void setColTypeName(java.lang.String ColTypeName)
                    throws java.sql.SQLException
```

getColTypeSchema

```
public java.lang.String getColTypeSchema()
                                  throws java.sql.SQLException
```

setColTypeSchema

```
public void setColTypeSchema(java.lang.String ColTypeSchema)
                     throws java.sql.SQLException
```

getTablePartition

```
public java.lang.String getTablePartition()
                                   throws java.sql.SQLException
```

setTablePartition

```
public void setTablePartition(java.lang.String TablePartition)
              throws java.sql.SQLException
```

ODCIColInfoList

```
oracle.ODCI.ODCIColInfoList
public class ODCIColInfoList
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIColInfoList

```
public ODCIColInfoList()
```

ODCIColInfoList

```
public ODCIColInfoList(ODCIColInfo[] a)
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

length

```
public int length()
           throws java.sql.SQLException
```

getBaseType

```
public int getBaseType()
               throws java.sql.SQLException
```

getBaseTypeName

```
public java.lang.String getBaseTypeName()
                                 throws java.sql.SQLException
```

getDescriptor

```
public oracle.sql.ArrayDescriptor getDescriptor()
                                        throws java.sql.SQLException
```

getArray

```
public ODCIColInfo[] getArray()
                       throws java.sql.SQLException
```

setArray

```
public void setArray(ODCIColInfo[] a)
              throws java.sql.SQLException
```

getArray

```
public ODCIColInfo[] getArray(long index,
                              int count)
                       throws java.sql.SQLException
```

setArray

```
public void setArray(ODCIColInfo[] a,
                     long index)
              throws java.sql.SQLException
```

getElement

```
public ODCIColInfo getElement(long index)
                      throws java.sql.SQLException
```

setElement

```
public void setElement(ODCIColInfo a,
                       long index)
        throws java.sql.SQLException
```

ODCIColInfoRef

```
oracle.ODCI.ODCIColInfoRef
public class ODCIColInfoRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIColInfoRef

```
public ODCIColInfoRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCIColInfo getValue()
                    throws java.sql.SQLException
```

setValue

```
public void setValue(ODCIColInfo c)
              throws java.sql.SQLException
```

ODCICost

```
oracle.ODCI.ODCICost
public class ODCICost
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCICost

```
public ODCICost()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getCPUcost

```
public java.math.BigDecimal getCPUcost()
                                throws java.sql.SQLException
```

setCPUcost

```
public void setCPUcost(java.math.BigDecimal CPUcost)
               throws java.sql.SQLException
```

getIOcost

```
public java.math.BigDecimal getIOcost()
                              throws java.sql.SQLException
```

setIOcost

```
public void setIOcost(java.math.BigDecimal IOcost)
              throws java.sql.SQLException
```

getNetworkCost

```
public java.math.BigDecimal getNetworkCost()
                                  throws java.sql.SQLException
```

setNetworkCost

```
public void setNetworkCost(java.math.BigDecimal NetworkCost)
                   throws java.sql.SQLException
```

getIndexCostInfo

```
public java.lang.String getIndexCostInfo()
                                throws java.sql.SQLException
```

setIndexCostInfo

```
public void setIndexCostInfo(java.lang.String IndexCostInfo)
            throws java.sql.SQLException
```

ODCICostRef

```
oracle.ODCI.ODCICostRef
public class ODCICostRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCICostRef

```
public ODCICostRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCICost getValue()
                  throws java.sql.SQLException
```

setValue

```
public void setValue(ODCICost c)
       throws java.sql.SQLException
```

ODCIEnv

```
oracle.ODCI.ODCIEnv
public class ODCIEnv
```

フィールド

SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIEnv

```
public ODCIEnv()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getEnvFlags

```
public java.math.BigDecimal getEnvFlags()
                                  throws java.sql.SQLException
```

setEnvFlags

```
public void setEnvFlags(java.math.BigDecimal EnvFlags)
                 throws java.sql.SQLException
```

getCallProperty

```
public java.math.BigDecimal getCallProperty()
                                     throws java.sql.SQLException
```

setCallProperty

```
public void setCallProperty(java.math.BigDecimal CallProperty)
                    throws java.sql.SQLException
```

ODCIEnvRef

```
oracle.ODCI.ODCIEnvRef
public class ODCIEnvRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIEnvRef

```
public ODCIEnvRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCIEnv getValue()
                throws java.sql.SQLException
```

setValue

```
public void setValue(ODCIEnv c)
              throws java.sql.SQLException
```

ODCIFuncInfo

```
oracle.ODCI.ODCIFuncInfo
public class ODCIFuncInfo
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIFuncInfo

```
public ODCIFuncInfo()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getObjectSchema

```
public java.lang.String getObjectSchema()
                            throws java.sql.SQLException
```

setObjectSchema

```
public void setObjectSchema(java.lang.String ObjectSchema)
                      throws java.sql.SQLException
```

getObjectName

```
public java.lang.String getObjectName()
                                throws java.sql.SQLException
```

setObjectName

```
public void setObjectName(java.lang.String ObjectName)
                    throws java.sql.SQLException
```

getMethodName

```
public java.lang.String getMethodName()
                                throws java.sql.SQLException
```

setMethodName

```
public void setMethodName(java.lang.String MethodName)
                    throws java.sql.SQLException
```

getFlags

```
public java.math.BigDecimal getFlags()
                               throws java.sql.SQLException
```

setFlags

```
public void setFlags(java.math.BigDecimal Flags)
               throws java.sql.SQLException
```

ODCIFuncInfoRef

```
oracle.ODCI.ODCIFuncInfoRef
public class ODCIFuncInfoRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIFuncInfoRef

```
public ODCIFuncInfoRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCIFuncInfo getValue()
                      throws java.sql.SQLException
```

setValue

```
public void setValue(ODCIFuncInfo c)
              throws java.sql.SQLException
```

ODCIIndexCtx

```
oracle.ODCI.ODCIIndexCtx
public class ODCIIndexCtx
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIIndexCtx

```
public ODCIIndexCtx()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getIndexInfo

```
public ODCIIndexInfo getIndexInfo()
                            throws java.sql.SQLException
```

setIndexInfo

```
public void setIndexInfo(ODCIIndexInfo IndexInfo)
                   throws java.sql.SQLException
```

getRid

```
public java.lang.String getRid()
                         throws java.sql.SQLException
```

setRid

```
public void setRid(java.lang.String Rid)
       throws java.sql.SQLException
```

ODCIIndexCtxRef

```
oracle.ODCI.ODCIIndexCtxRef
public class ODCIIndexCtxRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIIndexCtxRef

```
public ODCIIndexCtxRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCIIndexCtx getValue()
                      throws java.sql.SQLException
```

setValue

```
public void setValue(ODCIIndexCtx c)
              throws java.sql.SQLException
```

ODCIIndexInfo

```
oracle.ODCI.ODCIIndexInfo
public class ODCIIndexInfo
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIIndexInfo

```
public ODCIIndexInfo()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getIndexSchema

```
public java.lang.String getIndexSchema()
                                  throws java.sql.SQLException
```

setIndexSchema

```
public void setIndexSchema(java.lang.String IndexSchema)
                    throws java.sql.SQLException
```

getIndexName

```
public java.lang.String getIndexName()
                                throws java.sql.SQLException
```

setIndexName

```
public void setIndexName(java.lang.String IndexName)
                  throws java.sql.SQLException
```

getIndexCols

```
public ODCIColInfoList getIndexCols()
                               throws java.sql.SQLException
```

setIndexCols

```
public void setIndexCols(ODCIColInfoList IndexCols)
                  throws java.sql.SQLException
```

getIndexPartition

```
public java.lang.String getIndexPartition()
                                    throws java.sql.SQLException
```

setIndexPartition

```
public void setIndexPartition(java.lang.String IndexPartition)
                       throws java.sql.SQLException
```

getIndexInfoFlags

```
public java.math.BigDecimal getIndexInfoFlags()
                                    throws java.sql.SQLException
```

setIndexInfoFlags

```
public void setIndexInfoFlags(java.math.BigDecimal IndexInfoFlags)
                        throws java.sql.SQLException
```

ODCIIndexInfoRef

```
oracle.ODCI.ODCIIndexInfoRef
public class ODCIIndexInfoRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIIndexInfoRef

```
public ODCIIndexInfoRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCIIndexInfo getValue()
                      throws java.sql.SQLException
```

setValue

```
public void setValue(ODCIIndexInfo c)
              throws java.sql.SQLException
```

ODCIObject

```
oracle.ODCI.ODCIObject
public class ODCIObject
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIObject

```
public ODCIObject()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getObjectSchema

```
public java.lang.String getObjectSchema()
                                  throws java.sql.SQLException
```

setObjectSchema

```
public void setObjectSchema(java.lang.String ObjectSchema)
                    throws java.sql.SQLException
```

getObjectName

```
public java.lang.String getObjectName()
                                throws java.sql.SQLException
```

setObjectName

```
public void setObjectName(java.lang.String ObjectName)
         throws java.sql.SQLException
```

ODCIObjectList

```
oracle.ODCI.ODCIObjectList
public class ODCIObjectList
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIObjectList

```
public ODCIObjectList()
```

ODCIObjectList

```
public ODCIObjectList(ODCIObject[] a)
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

length

```
public int length()
           throws java.sql.SQLException
```

getBaseType

```
public int getBaseType()
               throws java.sql.SQLException
```

getBaseTypeName

```
public java.lang.String getBaseTypeName()
                                 throws java.sql.SQLException
```

getDescriptor

```
public oracle.sql.ArrayDescriptor getDescriptor()
                                        throws java.sql.SQLException
```

getArray

```
public ODCIObject[] getArray()
                     throws java.sql.SQLException
```

setArray

```
public void setArray(ODCIObject[] a)
              throws java.sql.SQLException
```

getArray

```
public ODCIObject[] getArray(long index,
                             int count)
                     throws java.sql.SQLException
```

setArray

```
public void setArray(ODCIObject[] a,
                     long index)
              throws java.sql.SQLException
```

getElement

```
public ODCIObject getElement(long index)
                      throws java.sql.SQLException
```

setElement

```
public void setElement(ODCIObject a,
                       long index)
        throws java.sql.SQLException
```

ODCIObjectRef

```
oracle.ODCI.ODCIObjectRef
public class ODCIObjectRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIObjectRef

```
public ODCIObjectRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCIObject getValue()
                    throws java.sql.SQLException
```

setValue

```
public void setValue(ODCIObject c)
        throws java.sql.SQLException
```

ODCIPartInfo

```
oracle.ODCI.ODCIPartInfo
public class ODCIPartInfo
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIPartInfo

```
public ODCIPartInfo()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getTablePartition

```
public java.lang.String getTablePartition()
                                    throws java.sql.SQLException
```

setTablePartition

```
public void setTablePartition(java.lang.String TablePartition)
                        throws java.sql.SQLException
```

getIndexPartition

```
public java.lang.String getIndexPartition()
                                    throws java.sql.SQLException
```

setIndexPartition

```
public void setIndexPartition(java.lang.String IndexPartition)
                    throws java.sql.SQLException
```

ODCIPartInfoRef

```
oracle.ODCI.ODCIPartInfoRef
public class ODCIPartInfoRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIPartInfoRef

```
public ODCIPartInfoRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCIPartInfo getValue()
                     throws java.sql.SQLException
```

setValue

```
public void setValue(ODCIPartInfo c)
       throws java.sql.SQLException
```

ODCIPredInfo

```
oracle.ODCI.ODCIPredInfo
public class ODCIPredInfo
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIPredInfo

```
public ODCIPredInfo()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getObjectSchema

```
public java.lang.String getObjectSchema()
                                  throws java.sql.SQLException
```

setObjectSchema

```
public void setObjectSchema(java.lang.String ObjectSchema)
                     throws java.sql.SQLException
```

getObjectName

```
public java.lang.String getObjectName()
                                throws java.sql.SQLException
```

setObjectName

```
public void setObjectName(java.lang.String ObjectName)
                   throws java.sql.SQLException
```

getMethodName

```
public java.lang.String getMethodName()
                                throws java.sql.SQLException
```

setMethodName

```
public void setMethodName(java.lang.String MethodName)
                   throws java.sql.SQLException
```

getFlags

```
public java.math.BigDecimal getFlags()
                               throws java.sql.SQLException
```

setFlags

```
public void setFlags(java.math.BigDecimal Flags)
        throws java.sql.SQLException
```

ODCIPredInfoRef

```
oracle.ODCI.ODCIPredInfoRef
public class ODCIPredInfoRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIPredInfoRef

```
public ODCIPredInfoRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCIPredInfo getValue()
                     throws java.sql.SQLException
```

setValue

```
public void setValue(ODCIPredInfo c)
       throws java.sql.SQLException
```

ODCIQueryInfo

```
oracle.ODCI.ODCIQueryInfo
public class ODCIQueryInfo
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIQueryInfo

```
public ODCIQueryInfo()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getFlags

```
public java.math.BigDecimal getFlags()
                              throws java.sql.SQLException
```

setFlags

```
public void setFlags(java.math.BigDecimal Flags)
             throws java.sql.SQLException
```

getAncOps

```
public ODCIObjectList getAncOps()
                        throws java.sql.SQLException
```

setAncOps

```
public void setAncOps(ODCIObjectList AncOps)
                 throws java.sql.SQLException
```

ODCIQueryInfoRef

```
oracle.ODCI.ODCIQueryInfoRef
public class ODCIQueryInfoRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIQueryInfoRef

```
public ODCIQueryInfoRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCIQueryInfo getValue()
                      throws java.sql.SQLException
```

setValue

```
public void setValue(ODCIQueryInfo c)
                throws java.sql.SQLException
```

ODCIRidList

```
oracle.ODCI.ODCIRidList
public class ODCIRidList
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIRidList

```
public ODCIRidList()
```

ODCIRidList

```
public ODCIRidList(java.lang.String[] a)
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
```

```
                                        int sqlType)
                                  throws java.sql.SQLException
```

length

```
public int length()
           throws java.sql.SQLException
```

getBaseType

```
public int getBaseType()
               throws java.sql.SQLException
```

getBaseTypeName

```
public java.lang.String getBaseTypeName()
                                    throws java.sql.SQLException
```

getDescriptor

```
public oracle.sql.ArrayDescriptor getDescriptor()
                                          throws java.sql.SQLException
```

getArray

```
public java.lang.String[] getArray()
                            throws java.sql.SQLException
```

setArray

```
public void setArray(java.lang.String[] a)
              throws java.sql.SQLException
```

getArray

```
public java.lang.String[] getArray(long index,
                                   int count)
                            throws java.sql.SQLException
```

setArray

```
public void setArray(java.lang.String[] a,
                     long index)
```

```
                        throws java.sql.SQLException
```

getElement

```
public java.lang.String getElement(long index)
                            throws java.sql.SQLException
```

setElement

```
public void setElement(java.lang.String a,
                       long index)
                    throws java.sql.SQLException
```

ODCIStatsOptions

```
oracle.ODCI.ODCIStatsOptions
public class ODCIStatsOptions
```

フィールド

_SQL_NAME

```
public static final java.lang.String _SQL_NAME
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIStatsOptions

```
public ODCIStatsOptions()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getSample

```
public java.math.BigDecimal getSample()
                              throws java.sql.SQLException
```

setSample

```
public void setSample(java.math.BigDecimal Sample)
              throws java.sql.SQLException
```

getOptions

```
public java.math.BigDecimal getOptions()
                               throws java.sql.SQLException
```

setOptions

```
public void setOptions(java.math.BigDecimal Options)
               throws java.sql.SQLException
```

getFlags

```
public java.math.BigDecimal getFlags()
                             throws java.sql.SQLException
```

setFlags

```
public void setFlags(java.math.BigDecimal Flags)
                throws java.sql.SQLException
```

ODCIStatsOptionsRef

```
oracle.ODCI.ODCIStatsOptionsRef
public class ODCIStatsOptionsRef
```

フィールド

_SQL_BASETYPE

```
public static final java.lang.String _SQL_BASETYPE
```

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

コンストラクタ

ODCIStatsOptionsRef

```
public ODCIStatsOptionsRef()
```

メソッド

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection c)
                         throws java.sql.SQLException
```

create

```
public oracle.sql.ORAData create(oracle.sql.Datum d,
                                 int sqlType)
                          throws java.sql.SQLException
```

getValue

```
public ODCIStatsOptions getValue()
                          throws java.sql.SQLException
```

setValue

```
public void setValue(ODCIStatsOptions c)
              throws java.sql.SQLException
```

第 II 部

Oracle9*i* XDK for Java 用の Java パッケージ

第 II 部では、Oracle XDK for Java に含まれている Java パッケージについて説明します。Oracle XML Developer's Kit（XDK）には、XML 文書を操作、変換および表示するための基本的なビルディング・ブロックが含まれています。

第 II 部は、次の章で構成されています。

- 第 6 章「パッケージ oracle.xml.classgen」
- 第 7 章「パッケージ oracle.xml.parser.schema」
- 第 8 章「パッケージ oracle.xml.sql.dml」
- 第 9 章「パッケージ oracle.xml.sql.query」
- 第 10 章「パッケージ oracle.xml.util」
- 第 11 章「パッケージ oracle.xml.parser.v2」

このパッケージは、製品版の Oracle XDK を完全にサポートし、市販の再配布ライセンスを備えています。製品ライブラリは、OTN および OTN-J の Web サイトで定期的に更新されます。詳細は、次の OTN および OTN-J の Web サイトで XDK for Java を参照してください。

- Oracle XDK ホーム : `http://otn.oracle.co.jp/tech/xml/xdk/index.html`
- Oracle XML Developer's Kit for Java:
  `http://otn.oracle.co.jp/software/tech/xml/xdk/index.html`
- Oracle XML Developer's Kit for JavaBeans:
  `http://otn.oracle.com/tech/xml/xdk_jbeans/content.html`

6

パッケージ oracle.xml.classgen

この章では、パッケージ oracle.xml.classgen について説明します。このパッケージには、Oracle9*i* XDK for Java の XML Class Generator に対するクラスが含まれています。Class Generator は、入力ファイルを受け入れ、対応する機能を持つ一連の出力クラスを作成するユーティリティです。XML Class Generator の場合の入力ファイルは DTD で、出力は、その DTD に準拠した XML 文書の作成に使用できる一連のクラスです。

この章は、次の項で構成されています。

パッケージ oracle.xml.classgen の説明

XML Class Generator for Java は、XML DTD または XML Schema から Java ソース・ファイルを作成します。この機能は、XML メッセージを互いに同意済みの DTD またはスキーマに基づくアプリケーション間で送信する場合や、あるいは構成する Web フォームと XML 文書のバック・エンドとして役立ちます。これらのクラスを使用すると、Java アプリケーションは、入力の DTD またはスキーマに準拠する XML 文書を作成、検証および出力することができます。Class Generator は、Oracle XML Parser for Java とともに機能します。Oracle XML Parser for Java は、DTD またはスキーマを解析し、解析したドキュメントを Class Generator に渡します。

詳細は、OTN-J の Web サイトで XDK for Java の Oracle リソースを参照してください。

関連項目：

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』

パッケージ oracle.xml.classgen の概要

表 6-1 oracle.xml.classgen の概要

名前	説明
CGDocument クラス	DTD Class Generator によって生成されるクラスのベース・ドキュメント・クラスとしての役割を果たします。
CGNode クラス	DTD Class Generator によって生成される XML 文書のノードに対応するクラスのベース・クラスとしての役割を果たします。
CGXSDElement クラス	Schema Class Generator によって生成される XML Schema に対応する、すべての生成されたクラスのベース・クラスとしての役割を果たします。
DTDClassGenerator クラス	DTD または DTD に基づいた 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。
InvalidContentException クラス	DTD Class Generator および Schema Class Generator クラスによって発生する例外を定義します。
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。
SchemaClassGenerator クラス	XML Schema に対応するクラスを生成します。

CGDocument クラス

CGDocument の説明

このクラスは、DTD Class Generator によって生成されるクラスのベース・ドキュメント・クラスとしての役割を果たします。

CGDocument の構文

```
public abstract class CGDocument extends oracle.xml.classgen.CGNode implements
java.io.Externalizable

oracle.xml.classgen.CGNode
  |
  +--oracle.xml.classgen.CGDocument
```

CGDocument の実装されるインタフェース

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

CGDocument のメソッド

表 6-2 CGDocument のメソッドの概要

メソッド	説明
6-4 ページの「CGDocument()」	DTD のルート要素のコンストラクタです。
6-5 ページの「print()」	作成された XML 文書を出力します。
6-5 ページの「readExternal()」	圧縮されたストリームを読み込み、ルート要素に対応するオブジェクトを作成します。

CGDocument()

説明

DTD のルート要素のコンストラクタです。

構文

```
protected  CGDocument( java.lang.String doctype,
                oracle.xml.parser.v2.DTD dtd);
```

パラメータ

doctype	DTD のルート要素の名前
dtd	クラスの生成に使用される DTD

print()

説明

作成された XML 文書を出力します。ドキュメントの内容が DTD で指定された構文に適合しないと、InvalidContentException が発生します（検証モードを true に設定しておく必要があります）。「DTDClassGenerator クラス」の「setValidationMode()」も参照してください。次の表で、各オプションについて説明します。

構文	説明
protected void print(java.io.OutputStream out);	作成された XML 文書を出力ストリームに出力します。
protected void print(java.io.OutputStream out, java.lang.String enc);	作成された XML 文書を、ユーザー定義のエンコーディングで出力ストリームに出力します。

パラメータ

out	ドキュメントが出力される出力ストリーム
enc	出力ストリームのエンコーディング

readExternal()

説明

圧縮されたストリームを読み込み、ルート要素に対応するオブジェクトを作成します。XML インスタンス文書で生成されたクラスをインスタンス化するために使用します。

構文

```
protected void readExternal( java.io.ObjectInput inArg,
                oracle.xml.comp.CXMLContext cxmlContext);
```

パラメータ

inArg	圧縮されたストリームを読み込むために渡される ObjectInput ストリーム
cxmlContext	圧縮されたストリームのコンテキスト

CGNode クラス

CGNode の説明

このクラスは、DTD Class Generator によって生成される XML 文書のノードに対応するクラスのベース・クラスとしての役割を果たします。

CGNode の構文

```
public abstract class CGNode
```

oracle.xml.classgen.CGNode

CGNode のダイレクト・サブクラス

CGDocument

CGNode のフィールド

isValidating

```
protected boolean isValidating
```
検証モードを示す boolean 型の値

CGNode のメソッド

表 6-3 CGNode のメソッドの概要

メソッド	説明
6-8 ページの「CGNode()」	DOM ツリーの要素のコンストラクタです。
6-9 ページの「addCDATASection()」	要素に CDATA セクションを追加します。
6-9 ページの「addData()」	要素ノードに PCDATA を追加します。
6-9 ページの「addNode()」	要素に子としてノードを追加します。
6-10 ページの「deleteData()」	要素ノードから PCDATA を削除します。
6-10 ページの「getAttribute()」	属性の値を取り出します。
6-10 ページの「getCGDocument()」	ベース・ドキュメントを取り出します。
6-11 ページの「getData()」	要素の PCDATA を取り出します。

表 6-3 CGNode のメソッドの概要（続き）

メソッド	説明
6-11 ページの「getDTDNode()」	ベース・ドキュメントから静的 DTD を取り出します。
6-11 ページの「getElementNode()」	CGNode に対応する XMLElement ノードを取り出します。
6-11 ページの「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 を取り出します。
6-12 ページの「readExternal()」	圧縮されたストリームを読み込み、対応するノードをインスタンス化します。
6-12 ページの「setAttribute()」	属性の値を設定します。
6-13 ページの「setDocument()」	ベース・ドキュメントを設定します。
6-13 ページの「setElementNode()」	CGNode に対応する XMLElement ノードを設定します。
6-13 ページの「storeID()」	ID 識別子の値を格納します。
6-14 ページの「storeIDREF()」	IDREF 識別子の値を格納します。
6-14 ページの「validateContent()」	要素の内容が有効かどうかを、DTD に指定されたコンテンツ・モデルに従ってチェックします。
6-14 ページの「validEntity()」	ENTITY 識別子が有効かどうかをチェックします。
6-15 ページの「validID()」	ID 識別子が有効かどうかをチェックします。
6-15 ページの「validNMTOKEN()」	NMTOKEN 識別子が有効かどうかをチェックします。
6-15 ページの「writeExternal()」	CGNode に対応する圧縮されたストリームを書き込みます。

CGNode()

説明

DOM ツリーの要素のコンストラクタです。

構文

```
protected  CGNode( java.lang.String elementName);
```

パラメータ

elementName	要素の名前

addCDATASection()

説明

要素に CDATA セクションを追加します。theData に無効な文字が含まれている場合は、InvalidContentException が発生します（検証モードを true に設定しておく必要があります）。「DTDClassGenerator クラス」の「setValidationMode()」も参照してください。

構文

```
protected void addCDATASection( java.lang.String theData);
```

パラメータ

theData	CDATA セクションとして要素に追加されるテキスト

addData()

説明

要素ノードに PCDATA を追加します。theData に無効な文字が含まれている場合は、InvalidContentException が発生します（検証モードを true に設定しておく必要があります）。「DTDClassGenerator クラス」の「setValidationMode()」も参照してください。

構文

```
protected void addData( java.lang.String theData);
```

パラメータ

theData	要素に追加されるテキスト

addNode()

説明

要素に子としてノードを追加します。theData に無効な文字が含まれている場合は、InvalidContentException が発生します（検証モードを true に設定しておく必要があります）。「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

theNode	子として追加されるノード

deleteData()

説明

要素ノードから PCDATA を削除します。theData に無効な文字が含まれている場合は、InvalidContentException が発生します（検証モードを true に設定しておく必要があります）。「DTDClassGenerator クラス」の「setValidationMode()」も参照してください。

構文

```
protected void deleteData( java.lang.String theData);
```

パラメータ

theData	要素から削除するテキスト

getAttribute()

説明

属性の値を戻します。

構文

```
protected java.lang.String getAttribute( java.lang.String attName);
```

パラメータ

attName	属性の名前

getCGDocument()

説明

ベース・ドキュメント（ルート要素）を取得します。

構文

```
protected CGDocument getCGDocument();
```

getData()

説明

要素の PCDATA を取得します。データが存在しない場合は、InvalidContentException が発生します。

構文

```
protected java.lang.String getData();
```

getDTDNode()

説明

ベース CGDocument から静的 DTD を取り出します。

構文

```
protected abstract oracle.xml.parser.v2.DTD getDTDNode();
```

getElementNode()

説明

この CGNode に対応する XMLElement ノードを取り出します。

構文

```
protected oracle.xml.parser.v2.XMLElement getElementNode();
```

getNode()

説明

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 を取り出します。

構文

```
protected java.lang.Object getNode(java.lang.String theNode);
```

パラメータ

theNode	戻された CGNode に対応する文字列の名前

readExternal()

説明

圧縮されたストリームを読み込み、対応するノードをインスタンス化します。次の例外が発生します。

IOException	I/O エラーが発生した場合
ClassNotFoundException	対応するクラスをインスタンス化できなかった場合

構文

```
protected void readExternal(oracle.xml.io.XMLObjectInput in,
                oracle.xml.comp.CXMLContext cxmlContext)
```

パラメータ

in	圧縮されたストリームの読込みに使用される XMLObjectInput ストリーム
cxmlContext	圧縮されたストリームのコンテキスト

setAttribute()

説明

属性の値を設定します。

構文

```
protected void setAttribute(java.lang.String attName,
                java.lang.String value);
```

パラメータ

attName	属性の名前
value	属性の値

setDocument()

説明

ベース・ドキュメント（ルート要素）を設定します。

構文

```
public void setDocument( CGDocument d);
```

パラメータ

d	ベース CGDocument

setElementNode()

説明

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

node	XMLElement

storeID()

説明

ID 識別子の値を格納します。この識別子は IDREF 値で検証できます。

構文

```
protected void storeID(java.lang.String attName,
                java.lang.String id);
```

パラメータ

attName	ID 属性の名前
id	ID の値

storeIDREF()

説明

IDREF 識別子の値を格納します。この識別子は対応する ID によって検証できます。

構文

```
protected void storeIDREF( java.lang.String attName,
                java.lang.String idref);
```

パラメータ

attName	IDREF 属性の名前
idref	IDREF の値

validateContent()

説明

要素の内容が有効かどうかを、DTD に指定されたコンテンツ・モデルに従ってチェックします。

構文

```
protected void validateContent();
```

validEntity()

説明

ENTITY 識別子が有効かどうかをチェックします。 ENTITY が有効な場合は `true` が、それ以外の場合は `false` が戻されます。

構文

```
protected boolean validEntity( java.lang.String entity);
```

パラメータ

entity	ENTITY 属性の値

validID()

説明

ID 識別子が有効かどうかをチェックします。ID が有効な場合は `true` が、それ以外の場合は `false` が戻されます。

構文

```
protected boolean validID( java.lang.String name);
```

パラメータ

name	ID 属性の値

validNMTOKEN()

説明

NMTOKEN 識別子が有効かどうかをチェックします。NMTOKEN が有効な場合は `true` が、それ以外の場合は `false` が戻されます。

構文

```
protected boolean validNMTOKEN( java.lang.String name);
```

パラメータ

name	NMTOKEN 属性の値

writeExternal()

説明

CGNode に対応する圧縮された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

out	圧縮されたデータを書き込む ObjectOutput ストリーム
cxmlContext	圧縮されたストリームのコンテキスト

CGXSDElement クラス

CGXSDElement の説明

このクラスは、Schema Class Generator によって生成される XML Schema に対応する、すべての生成されたクラスのベース・クラスとしての役割を果たします。

CGXSDElement の構文

```
public abstract class CGXSDElement extends java.lang.Object

java.lang.Object
  |
  +--oracle.xml.classgen.CGXSDElement
```

CGXSDElement のフィールド

表 6-4 ElementDecl のフィールド

フィールド	構文	説明
type	protected java.lang.Object type	ノードの型情報

CGXSDElement のメソッド

表 6-5 CGXSDElement のメソッドの概要

メソッド	説明
6-18 ページの「CGXSDElement()」	デフォルトのコンストラクタです。
6-18 ページの「addAttribute()」	指定されたノードの属性をハッシュテーブルに追加します。
6-18 ページの「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。
6-19 ページの「getAttributes()」	属性を、属性名と値のハッシュテーブルとして戻します。
6-19 ページの「getChildElements()」	すべてのローカル要素のベクトルを取り出します。
6-19 ページの「getNodeValue()」	ノードの値を戻します。
6-19 ページの「print()」	要素ノードを出力します。

表 6-5 CGXSDElement のメソッドの概要（続き）

メソッド	説明
6-20 ページの「printAttributes()」	属性ノードを出力します。
6-20 ページの「setNodeValue()」	属性のノード値を設定します。

CGXSDElement()

説明
デフォルトのコンストラクタです。

構文
```
public  CGXSDElement();
```

addAttribute()

説明
指定されたノードの属性をハッシュテーブルに追加します。

構文
```
protected void addAttribute(java.lang.String attName,
                java.lang.Object attValue);
```

パラメータ

attName	属性名
attValue	属性値

addElement()

説明
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

elem	追加する必要があるオブジェクト

getAttributes()

説明
属性を、属性名と値のハッシュテーブルとして戻します。

構文
```
public java.util.Hashtable getAttributes();
```

getChildElements()

説明
すべてのローカル要素のベクトルを取り出します。

構文
```
public java.util.Vector getChildElements();
```

getNodeValue()

説明
ノードの値を戻します。

構文
```
public java.lang.String getNodeValue();
```

print()

説明
要素ノードを出力します。 出力ストリームに出力できない場合は、IOException が発生します。

構文
```
public void print( oracle.xml.parser.v2.XMLOutputStream out);
```

パラメータ

out	結果が出力される XMLObjectOutput ストリーム

printAttributes()

説明

属性ノードを出力します。XMLObjectOutput ストリームに出力できない場合は、IOException が発生します。

構文

```
public void printAttributes( oracle.xml.parser.v2.XMLOutputStream out,
                java.lang.String name,
                java.lang.String namespace);
```

パラメータ

out	結果が出力される XMLObjectOutput ストリーム
name	属性名
namespace	名前空間

setNodeValue()

説明

属性のノード値を設定します。

構文

```
protected void setNodeValue( java.lang.String value);
```

パラメータ

value	ノードの値

DTDClassGenerator クラス

DTDClassGenerator の説明

DTD または DTD に基づいた 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。

DTDClassGenerator の構文

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

java.lang.Object
  |
  +--oracle.xml.classgen.DTDClassGenerator
```

DTDClassGenerator のメソッド

表 6-6 DTDClassGenerator のメソッドの概要

メソッド	説明
6-22 ページの「DTDClassGenerator()」	DTDClassGenerator のデフォルトのコンストラクタです。
6-22 ページの「generate()」	ルートとして要素 doctype を使用して DTD を全検索し、Java クラスを生成します。
6-22 ページの「setGenerateComments()」	生成されたクラスに対して javadoc コメントを生成するかどうかを判断するためのスイッチを設定します。
6-23 ページの「setJavaPackage()」	生成されたクラスのパッケージを設定します。
6-23 ページの「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。
6-23 ページの「setSerializationMode()」	DTD をシリアル化オブジェクトとして保存するか、またはテキスト・ファイルとして保存するかを判断するためのスイッチを設定します。
6-24 ページの「setValidationMode()」	生成されたクラスによって XML 文書を検証するかどうかを判断するためのスイッチを設定します。

DTDClassGenerator()

説明

DTDClassGenerator のデフォルトのコンストラクタです。

構文

```
public  DTDClassGenerator();
```

generate()

説明

ルートとして要素 doctype を使用して DTD を全検索し、Java クラスを生成します。

構文

```
public void generate( oracle.xml.parser.v2.DTD dtd,
                java.lang.String doctype);
```

パラメータ

dtd	クラスの生成に使用される DTD
doctype	ルート要素の名前

setGenerateComments()

説明

生成されたクラスに対して javadoc コメントを生成するかどうかを判断するためのスイッチを設定します。デフォルト値は true です。

構文

```
public void setGenerateComments(boolean comments);
```

パラメータ

comments	javadoc コメントを生成するかどうかを切り替える boolean 型のフラグ

setJavaPackage()

説明

生成されたクラスのパッケージを設定します。 デフォルトではパッケージは設定されません。

構文

```
public void setJavaPackage( java.util.Vector packageName);
```

パラメータ

packageName	パッケージの名前

setOutputDirectory()

説明

DTD の Java ソース・コードが生成される出力ディレクトリを設定します。デフォルト値はカレント・ディレクトリです。

構文

```
public void setOutputDirectory( java.lang.String dir);
```

パラメータ

dir	出力ディレクトリ

setSerializationMode()

説明

DTD をシリアル化オブジェクトとして保存するか、またはテキスト・ファイルとして保存するかを判断するためのスイッチを設定します。シリアル化 DTD を使用すると、生成されたクラスを使用して XML ファイルを作成するときの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

yes	DTD をシリアル化オブジェクト（true）として保存するかどうかを切り替えるための boolean 型のフラグ。デフォルトでは、テキスト・ファイル（false）として保存されます。

setValidationMode()

説明

生成されたクラスによって作成中の XML 文書を検証するかどうかを判断するためのスイッチを設定します。デフォルト値は true です。

構文

```
public void setValidationMode( boolean yes);
```

パラメータ

yes	XML 文書を検証するかどうかを切り替えるための boolean 型のフラグ。デフォルトは true です。

InvalidContentException クラス

InvalidContentException の説明

DTD Class Generator および Schema Class Generator クラスによって発生する例外を定義します。

InvalidContentException の構文

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

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--oracle.xml.classgen.InvalidContentException
```

InvalidContentException の実装されるインタフェース

```
java.io.Serializable
```

InvalidContentException のメソッド

InvalidContentException()

説明

コンストラクタです。次の表で、各オプションについて説明します。

構文	説明
public InvalidContentException();	デフォルトのコンストラクタです。
public InvalidContentException(java.lang.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取得します。

パラメータ

s	例外に関する情報を含んだ文字列

oracg クラス

oracg の説明

DTD または XML に対応する Java クラスを生成するためのコマンドライン・インタフェースを提供します。

oracg の構文

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

java.lang.Object
  |
  +--oracle.xml.classgen.oracg
```

oracg のコマンドライン・オプション

表 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・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 <rootNmae>]	入力ファイルは DTD ファイルまたは DTD に基づいた XML ファイルです。
-schema <Schema File>	入力ファイルはスキーマ・ファイルまたはスキーマに基づいた 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。
-package <Package Name>	生成された Java クラスのパッケージ名。
-comment	生成された Java ソース・コードのコメントを生成します。

SchemaClassGenerator クラス

SchemaClassGenerator の説明

このクラスは、XML Schema に対応するクラスを生成します。

SchemaClassGenerator の構文

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

java.lang.Object
  |
  +--oracle.xml.classgen.SchemaClassGenerator
```

SchemaClassGenerator のメソッド

表 6-8 SchemaClassGenerator のメソッドの概要

メソッド	説明
6-28 ページの「SchemaClassGenerator()」	コンストラクタです。
6-28 ページの「generate()」	最上位要素である simpleType 要素と complexType 要素に対応するスキーマ・クラスを生成します。
6-28 ページの「setGenerateComments()」	javadoc コメントを生成するかどうかを判断するためのスイッチを設定します。
6-29 ページの「setJavaPackage()」	各名前空間に、ユーザー定義の Java パッケージ名を割り当てます。
6-29 ページの「setOutputDirectory()」	スキーマ・クラスの Java ソース・コードが生成される出力ディレクトリを設定します。

SchemaClassGenerator()

説明

コンストラクタ。次の表で、各オプションについて説明します。

構文	説明
public SchemaClassGenerator();	Schema Class Generator のデフォルトの空コンストラクタです。
public SchemaClassGenerator(java.lang.String fileName)	このコンストラクタは、XML Schema の説明が含まれている入力文字列を取得します。

パラメータ

fileName	入力 XML Schema

generate()

説明

最上位要素である simpleType 要素と complexType 要素の各ノードで createSchemaClass() をコールして、これらの要素に対応するスキーマ・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

schema	スキーマ・オブジェクト

setGenerateComments()

説明

javadoc コメントを生成するかどうかを判断するためのスイッチを設定します。デフォルトは `true` です。

構文

```
public void setGenerateComments(boolean comments)
```

パラメータ

comments	javadoc コメントを生成するかどうかを切り替えます。デフォルトは `true` です。

setJavaPackage()

説明

各名前空間に、ユーザー定義の Java パッケージ名を割り当てます。スキーマに定義された名前空間の問合せが実行されます。スキーマに定義された名前空間の数と、ユーザーが指定したパッケージ名の数が一致しないと、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

schema	XML Schema
pkgName	コマンドラインを介して指定されたユーザー定義のパッケージ名を含んだベクトル

setOutputDirectory()

説明

スキーマ・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。カレント・ディレクトリがデフォルトです。

構文

```
public void setOutputDirectory( java.lang.String dir);
```

パラメータ

dir	出力ディレクトリ

7

パッケージ oracle.xml.parser.schema

この章では、oracle.xml.parser.schema パッケージについて説明します。パッケージ oracle.xml.parser.schema に含まれているクラスは、Oracle XML Schema Processor for Java を実装します。

クラス参照の他に、この章には次の項が含まれています。

パッケージ oracle.xml.parser.schema の説明

パッケージ oracle.xml.parser.schema に含まれているクラスは、Oracle XML Schema Processor for Java を実装し、World Wide Web Consortium（W3C）の XML Schema 仕様をサポートします。

XML Schema を使用すると、XML 文書のクラスを定義できます。「インスタンス文書」という用語は、特定の XML Schema 定義（XSD）に準拠した XML 文書を意味します。このマニュアルでは、読者に XML Schema の W3C 勧告に関する知識があることを想定しています。W3C 勧告に関する情報は、http://www.w3.org/ を参照してください。

Oracle XML Schema Processor for Java の機能

Oracle XML Schema Processor for Java は、Oracle XML Parser for Java v2 上に構築されており、次の機能があります。

- 組込みデータ型 : XML Schema では、一連の組込みデータ型が指定されます。一部は定義済みで基本データ型と呼ばれ、型システムの基礎を構成します。
- Simple API for XML（SAX）処理のサポート : ストリーム、一定のメモリー使用量および線形の処理時間のサポート。SAX は、XML Parser と XML アプリケーションの間のイベント・ベースの API です。オブジェクト・ベースのインタフェースは、DOM によって提供されます。
- 2000 年 10 月 24 日公開の暫定勧告および 2001 年 5 月 2 日公開の最終勧告での W3C XML Schema 仕様の完全サポート。

XML Schema Processor for Java 実行の要件

XML Schema Processor for Java を実行するには、Java 1.1.x 以上および JDK 1.1.x 以上をサポートするオペレーティング・システムが必要です。

関連項目：

- http://www.w3.org/
- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』

パッケージ oracle.xml.parser.schema の概要

パッケージ oracle.xml.parser.schema に含まれるクラスは、XML Schema と XML Schema 定義のサポートを実装します。

表 7-1 oracle.xml.parser.schema のクラスとインタフェースの概要

クラス / インタフェース	説明
XMLSchema クラス	最上位 XMLSchema ドキュメントの宣言、定義、スキーマの場所およびスキーマのターゲット名前空間を設定します。
XMLSchemaNode クラス	最上位 XMLSchema ドキュメントの宣言と定義、およびスキーマの場所とスキーマのターゲット名前空間を設定します。
XSDAttribute クラス	complexType 属性グループを表します。
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。
XSDComplexType クラス	XML 文書用の complexType に対する XML Schema 定義（XSD）を管理します。
XSDConstants インタフェース	XSDConstantValues インタフェースを実装します。
XSDConstrainingFacet クラス	XSDTypeConstants を実装します。
XSDDataValue クラス	XSDTypeConstants を実装します。
XSDElement クラス	要素の XMLSchema 定義を示します。
XSDException	XMLSchema の妥当性チェック時に例外が発生したことを示します。
XSDGroup クラス	complexType モデル・グループを示します。
XSDIdentity クラス	XSD の識別パラメータを示します。
XSDNode クラス	大部分の XSD クラスのルート・クラスです。
XSDSimpleType クラス	型を派生する XSDTypeContstants を実装します。
XSDTypeConstants インタフェース	XSDTypeConstants のインタフェースを実装します。
XSDValidator クラス	XMLSchema と照合してインスタンス XML 文書を検証します。

XMLSchema クラス

XMLSchema の説明

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 XML Schema が含まれています。XMLSchema オブジェクトは、XSDParser によってインスタンス XML 文書の妥当性チェックに使用されたり、XSDBuilder によって、インポートされたスキーマとして使用されます。

XMLSchema の構文

```
public class XMLSchema extends oracle.xml.parser.schema.XSDNode

oracle.xml.parser.schema.XSDNode
  |
  +--oracle.xml.parser.schema.XMLSchema
```

XMLSchema のメソッド

表 7-2　XMLSchemaNode のメソッド

メソッド	説明
XMLSchema()	クラス・コンストラクタです。
getAllTargetNS()	スキーマのターゲット名前空間を取得します。
getSchemaByTargetNS()	指定した名前空間に対する schemaNode を取得します。
getSchemaTargetNS()	最上位のスキーマのターゲット名前空間を取得します。
getXMLSchemaNodeTable()	XMLSchemaNode 表を取得します。
getXMLSchemaURLS()	スキーマの URL を取得します。
printSchema()	指定したスキーマ情報を出力します。

XMLSchema()

説明

XMLSchema クラスのコンストラクタです。

構文

表 7-3 XMLSchema コンストラクタのバージョン

構文
public XMLSchema() throws XSDException
public XMLSchema(int n) throws XSDException

パラメータ

表 7-4 XMLSchema コンストラクタのパラメータ

パラメータ	説明
n	schemaNode セットの初期サイズ

getAllTargetNS()

説明

スキーマで定義されたすべてのターゲット名前空間を取得します。

構文

```
public java.lang.String[] getAllTargetNS()
```

getSchemaByTargetNS()

説明

指定した名前空間に対する schemaNode を取得します。

構文

```
public XMLSchemaNode getSchemaByTargetNS(java.lang.String namespace)
```

パラメータ
namespace ― スキーマのターゲット名前空間

戻り値
XMLSchemaNode

getSchemaTargetNS()

説明
最上位のスキーマのターゲット名前空間を取得します。最上位のスキーマが複数ある場合は、作成中の最後のスキーマが戻されます。

構文
```
public java.lang.String getSchemaTargetNS()
```

getXMLSchemaNodeTable()

説明
XMLSchemaNode 表を取得します。

構文
```
public java.util.Hashtable getXMLSchemaNodeTable()
```

戻り値
ハッシュテーブル

getXMLSchemaURLS()

説明
XMLSchema の URL を取得します。

構文
```
public java.lang.String[] getXMLSchemaURLS()
```

戻り値
スキーマの URL の配列

printSchema()

説明

スキーマ情報を出力します。

構文

表 7-5 printSchema() のバージョン

構文	説明
public void printSchema()	スキーマ情報を出力します。
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

表 7-6 printSchema のパラメータ

パラメータ	説明
all	すべての情報（組込み情報も含む）の出力を示すフラグ

XMLSchemaNode クラス

XMLSchemaNode の説明

XMLSchemaNode クラスは、最上位 XMLSchema ドキュメントの宣言と定義、およびスキーマの場所とスキーマのターゲット名前空間を設定します。XMLSchema オブジェクトは、XMLSchema ドキュメントを処理した結果として、XSDBuilder によって作成されます。

XMLSchemaNode の構文

```
public class XMLSchemaNode extends oracle.xml.parser.schema.XSDNode

oracle.xml.parser.schema.XSDNode
  |
  +--oracle.xml.parser.schema.XMLSchemaNode
```

XMLSchemaNode のメソッド

表 7-7 XMLSchemaNode のメソッドの概要

メソッド	説明
XMLSchemaNode()	XMLSchema のコンストラクタです。
getAttributeDeclarations()	スキーマ内のすべての最上位属性を取得します。
getComplexTypeSet()	スキーマ内のすべての最上位 complexType 要素を取得します。
getComplexTypeTable()	complexType 定義を取得します。
getElementSet()	スキーマ内のすべての最上位要素を取得します。
getSimpleTypeSet()	スキーマ内のすべての最上位 simpleType 要素を取得します。
getSimpleTypeTable()	simpleType 定義を取得します。
getTargetNS()	スキーマの targetNS を取得します。
getTypeDefinitionTable()	型定義を取得します。

XMLSchemaNode()

説明

XMLSchema のコンストラクタです。

構文

```
public  XMLSchemaNode()
```

getAttributeDeclarations()

説明

スキーマ内のすべての最上位属性を取得します。

構文

```
public XSDAttribute getAttributeDeclarations()
```

戻り値

最上位属性定義の配列

getComplexTypeSet()

説明

スキーマ内のすべての最上位 complexType 要素を取得します。

構文

```
public XSDNode[] getComplexTypeSet()
```

戻り値

最上位 complexType ノードの配列

getComplexTypeTable()

説明

complexType 定義を取得します。

構文

```
public java.util.Hashtable getComplexTypeTable()
```

戻り値

complexTypes のハッシュテーブル

getElementSet()

説明

スキーマ内のすべての最上位要素を取得します。

構文

```
public XSDNode[] getElementSet()
```

戻り値

最上位 XSDNode 要素の配列

getSimpleTypeSet()

説明

スキーマ内のすべての最上位 simpleType 要素を取得します。

構文

```
public XSDNode[] getSimpleTypeSet()
```

戻り値

最上位 simpleType ノードの配列

getSimpleTypeTable()

説明

simpleType 定義を取得します。

構文

```
public java.util.Hashtable getSimpleTypeTable()
```

戻り値

simpleTypes のハッシュテーブル

getTargetNS()

説明

スキーマの targetNS を取得します。クラス `XSDNode` の `XSDNode.getTargetNS()` をオーバーライドします。

構文

```
public java.lang.String getTargetNS()
```

戻り値

値 targetNS

getTypeDefinitionTable()

説明

型定義を取得します。

構文

```
public java.util.Hashtable getTypeDefinitionTable()
```

戻り値

型定義のハッシュテーブル

XSDAttribute クラス

XSDAttribute の説明

XSDAttribute クラス。XMLSchema の complexType 属性グループを表します。

XSDAttribute の構文

```
public class XSDAttribute extends oracle.xml.parser.schema.XSDNode

oracle.xml.parser.schema.XSDNode
  |
  +--oracle.xml.parser.schema.XSDAttribute
```

XSDAttribute のメソッド

表 7-8 XSDAttribute のメソッドの概要

メソッド	説明
getDefaultVal()	要素の場合は、default 属性の値と use 属性に基づいた value 属性の値を取得します。
getFixedVal()	要素の場合は、fixed 属性の値と use 属性に基づいた value 属性の値を取得します。
getName()	ノードの名前を取得します。
getRefLocalname()	解決済み ref 属性のローカル名を取得します。
getRefNamespace()	解決済み ref 属性の名前空間を取得します。
getRefState()	refState を取得します。
getTargetNS()	ターゲット名前空間を取得します。
getType()	ノード・タイプを取得します。
isRequired()	属性が必須かどうかをチェックします。

getDefaultVal()

説明

要素の場合は、default 属性の値と use 属性に基づいた value 属性の値を取得します。

構文

```
public java.lang.String getDefaultVal()
```

戻り値

defaultVal

getFixedVal()

説明

要素の場合は、fixed 属性の値と use 属性に基づいた value 属性の値を取得します。

構文

```
public java.lang.String getFixedVal()
```

戻り値

defaultVal

getName()

説明

ノードの名前を取得します。クラス `XSDNode` の `XSDNode.getName()` をオーバーライドします。

構文

```
public java.lang.String getName()
```

戻り値

ノード名

getRefLocalname()

説明
解決済み ref 属性のローカル名を取得します。

構文
```
public java.lang.String getRefLocalname()
```

戻り値
refLocalname

getRefNamespace()

説明
解決済み ref 属性の名前空間を取得します。

構文
```
public java.lang.String getRefNamespace()
```

戻り値
refNamespace

getRefState()

説明
refState を取得します。戻り値は、TYPE_UNRESOLVED、TYPE_RESOLVED、REF_UNRESOLVED、REF_RESOLVED のいずれかです。

構文
```
public int getRefState()
```

戻り値
refstate 値

getTargetNS()

説明
ターゲット名前空間を取得します。

構文
```
public java.lang.String getTargetNS()
```

オーバーライド
クラス XSDNode の XSDNode.getTargetNS()

getType()

説明
ノード・タイプを取得します。

構文
```
public XSDNode getType()
```

戻り値
nodeType（simpleType または complexType）

isRequired()

説明
属性が必須かどうかをチェックします。

構文
```
public boolean isRequired()
```

XSDBuilder クラス

XSDBuilder の説明

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、最上位スキーマ宣言および定義に対応する一連のオブジェクト（情報セット）です。スキーマ・ドキュメントは、XML 解析され、DOM ツリーに変換されます。このスキーマ DOM ツリーは、次のような順序でスキーマ解析されます。まず、スキーマ・オブジェクトを作成し、参照可能になっているものがあるかどうか。次に、対応する DOM ツリーに置換されているかどうか。最上位宣言および定義が現在のスキーマ情報セットの項目として登録されます。最後に、最上位ツリー要素（情報セットの項目）がスキーマ解析されます。結果として作成される XMLSchema 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）を含んだ SNode タイプのノード / オブジェクトが前に付いた、低レベルの要素、グループ宣言および参照に対応するノードで構成されたツリーです。

XSDBuilder の構文

```
public class XSDBuilder extends oracle.xml.parser.schema.XSDNode

oracle.xml.parser.schema.XSDNode
  |
  +--oracle.xml.parser.schema.XSDBuilder
```

XSDBuilder のメソッド

表 7-9 XSDBuilder のメソッドの概要

メソッド	説明
XSDBuilder()	クラス・コンストラクタです。
build()	XMLSchema オブジェクトまたはドキュメントを作成します。
getObject()	XML Schema オブジェクトを戻します。
setEntityResolver()	import/include を解決するための EntityResolver を設定します。
setError()	XMLError オブジェクトを設定します。
setLocale()	エラー・レポート用のロケールを設定します。

XSDBuilder()

説明

XSDBuilder コンストラクタです。

構文

```
public XSDBuilder() throws XSDException
```

build()

説明

XMLSchema オブジェクト / ドキュメントを作成します。

構文

表 7-10 build() の構文

構文
public Object build(InputStream in, URL baseurl) throws Exception
public Object build(Reader r, URL baseurl) throws Exception
public Object build(String sysId) throws Exception
public Object build(String ns, String sysId) throws Exception
public Object build(String ns, URL sysId) throws Exception
public Object build(URL schemaurl) throws Exception
public Object build(XMLDocument schemaDoc) throws Exception
public Object build(XMLDocument[] schemaDocs, URL baseurl)
public Object build(XMLDocument doc, String fragment, String ns, URL sysId)
public Object build(XMLDocument schemaDoc, URL baseurl)

パラメータ

表 7-11 build() のパラメータ

パラメータ	説明
baseurl	関連する参照の解決に使用される URL。ドキュメントに import/include するために使用されます。
doc	スキーマ要素を含んだ XMLDocument

表 7-11 build() のパラメータ

パラメータ	説明
fragment	スキーマ要素のフラグメント ID
in	スキーマの InputStream
ns	targetNamespace の検証に使用する、スキーマのターゲット名前空間
r	スキーマのリーダー
schemaDoc	XMLDocument
schemaDocs	XMLDocument の配列
sysId	スキーマの位置
url	スキーマの URL

戻り値
XMLSchema オブジェクト

例外
Builder が XMLSchema オブジェクトを作成できない場合は、例外が発生します。

getObject()

説明
XML Schema オブジェクトを戻します。

構文
```
public Object getObject()
```

戻り値
XMLSchema オブジェクト

setEntityResolver()

説明

import/include を解決するための EntityResolver を設定します。
`org.xml.sax.EintityResolver` も参照してください。

構文

```
public void setEntityResolver( org.xml.sax.entityResolver entResolver)
```

パラメータ

表 7-12 setEntityResolver のパラメータ

パラメータ	説明
entResolver	EntityResolver

setError()

説明

XMLError エラー・オブジェクトを設定します。

構文

```
public void setError(XMLError er)
```

パラメータ

表 7-13 setError のパラメータ

パラメータ	説明
er	XMLError オブジェクト

setLocale()

説明

エラー・レポート用のロケールを設定します。

構文

```
public void setLocale(Locale locale)
```

パラメータ

表 7-14 setLocale のパラメータ

パラメータ	説明
locale	ロケール・オブジェクト

XSDComplexType クラス

XSDComplexType の説明

XSDComplexType クラスは、XML 文書用の XML Schema 定義（XSD）に対する complexType を管理します。XML Schema では、インスタンス文書または要素の構造は complexType と呼ばれます。

XSDComplexType の構文

```
public class XSDComplexType extends oracle.xml.parser.schema.XSDNode

oracle.xml.parser.schema.XSDNode
  |
  +--oracle.xml.parser.schema.XSDComplexType
```

XSDComplexType のメソッド

表 7-15 XSDComplexType のメソッドの概要

メソッド	説明
getAttributeDeclarations()	complexType の属性宣言を取得します。属性宣言のワイルド・カード配列は含みません。
getAttributeWildcard()	complexType の属性ワイルド・カードを取得します。
getBaseType()	complexType のベース型を取得します。
getContent()	complexType のコンテンツを取得します。
getDerivationMethod()	型がその親の型から派生された方法を示す数値コードを取得します。
getElementSet()	complexType 要素が別の complexType 要素を拡張する場合に、complexType 要素内のすべてのローカル要素を取得します。
getGroup()	属性グループ、または子と属性グループを取得します。
getRefLocalname()	解決済みの 'base' 属性のローカル名を取得します。
getTypeGroup()	この complexType のグループの種類を取得します。
init()	このグループを初期化します。
isAbstract()	この complexType が抽象であるかどうかを、ブール値 true または false で宣言します。

getAttributeDeclarations()

説明

この complexType の属性宣言を取得します。属性宣言のワイルド・カード配列は含みません。

構文

```
public XSDAttribute getAttributeDeclarations()
```

getAttributeWildcard()

説明

この complexType の属性ワイルド・カードを取得します。

構文

```
public oracle.xml.parser.schema.XSDAny getAttributeWildcard()
```

戻り値

属性ワイルド・カード（この型にある場合）

getBaseType()

説明

この complexType のベース型を取得します。

構文

```
public XSDNode getBaseType()
```

戻り値

XSDNode ― ベース型

getContent()

説明

この complexType のコンテンツを取得します。

構文

```
public int getContent()
```

getDerivationMethod()

説明

この型を構成するために使用された派生の種類を示す数値コードを戻します。

構文

```
public short getDerivationMethod()
```

戻り値

EXTENSION_DERIVATION または RESTRICTION_DERIVATION に対するコード番号

getElementSet()

説明

complexType 要素が別の complexType 要素を拡張する場合に、complexType 要素内のローカル要素をすべて取得します。

構文

```
public XSDNode[] getElementSet()
```

戻り値

ローカル要素の配列

getGroup()

説明

属性グループまたは子と属性グループを取得します。

構文

```
public XSDGroup getGroup()
```

戻り値
グループ

getRefLocalname()

説明
解決済みの base 属性のローカル名を取得します。

構文
```
public java.lang.String getRefLocalname()
```

戻り値
refLocalname

getTypeGroup()

説明
この complexType のグループの種類をモデル・グループまたは属性グループのいずれかで取得します。

構文
```
public XSDGroup getTypeGroup()
```

init()

説明
このグループを初期化します。

構文
```
public static void init()
```

isAbstract()

説明
グループを、ブール値 true または false を使用して抽象または非抽象にします。

構文
```
public boolean isAbstract()
```

XSDConstants インタフェース

XSDConstants の説明

XSDConstantValues インタフェースを実装します。

XSDConstants の構文

```
public class XSDConstants

oracle.xml.parser.schema.XSDConstants
```

XSDConstants のメソッド

XSDConstants()

説明

クラス・コンストラクタです。

構文

```
public XSDConstants()
```

XSDConstrainingFacet クラス

XSDConstrainingFacet の説明

XSDTypeConstants を実装します。XML Schema は、制限を使用したデータ型の派生時に、制約を適用するための 15 個のファセットを定義します。ファセットは、データ型の許可値を制約します。

一部のファセットでは、データ型に関する制限を定義するために値領域が使用されます。値領域は、指定したデータ型に対する値のセットです。字句領域は、データ型に対する有効なリテラルのセットです。列挙は、値領域を指定した値セットに制約します。データ型の値領域内の各値は、その字句領域内の 1 つ以上の字句で示されます。

XSDConstrainingFacet の構文

```
public class XSDConstrainingFacet extends java.lang.Object implements
oracle.xml.parser.schema.XSDTypeConstants

java.lang.Object
  |
  +--oracle.xml.parser.schema.XSDConstrainingFacet
```

XSDConstrainingFacet の実装されるインタフェース

XSDTypeConstants

XSDConstrainingFacet のメソッド

表 7-16 XSDConstrainingFacet のメソッドの概要

メソッド	説明
getFacetId()	ファセットの ID を取得します。
getLexicalEnumeration()	このファセットの値領域を定義する列挙字句の開始と終了のポイントを取得します。
getLexicalValue()	ファセットの字句領域の値を取得します。
getName()	ファセットの名前を取得します。
isFixed(boolean)	ファセットが固定であるかどうかを、ブール値 true または false で宣言します。
validateFacet(XSDDataValue)	データ型と照合してファセットを検証します。

getFacetId()

説明

ファセット ID を取得します。

構文

```
public int getFacetId()
```

getLexicalEnumeration()

説明

このファセットの値領域を定義する列挙字句の開始と終了のポイントを取得します。

構文

```
public java.util.Vector getLexicalEnumeration()
```

getLexicalValue()

説明

ファセットの字句領域の値を取得します。

構文

```
public java.lang.String getLexicalValue()
```

getName()

説明

ファセットの名前を取得します。

構文

```
public java.lang.String getName()
```

isFixed(boolean)

説明

ファセットが固定であるかどうかを、ブール値 true または false で宣言します。

構文

```
public boolean isFixed(boolean fixed)
```

validateFacet(XSDDataValue)

説明

XML Schema 定義と照合してファセットを検証します。

構文

```
public void validateFacet(XSDDataValue value)
```

XSDDataValue クラス

XSDDataValue の説明

XSDTypeConstants を実装します。

XSDDataValue の構文

```
public class XSDDataValue extends java.lang.Object implements
oracle.xml.parser.schema.XSDTypeConstants

java.lang.Object
  |
  +--oracle.xml.parser.schema.XSDDataValue
```

XSDDataValue の実装されるインタフェース

XSDTypeConstants

XSDDataValue のメソッド

表 7-17 XSDDataValue のメソッドの概要

メソッド	説明
compareTo()	2 つの値を比較します。結果は int 型で戻され、小さい場合は -1、等しい場合は 0、大きい場合は 1 が戻されます。
getLength()	STRING/BINARY 値の長さを取得します。
getLexicalValue()	XSDDataValue クラスから LEXICAL 値を取得します。文字列を戻します。
getPrecision()	小数値の精度を取得します。int 型の精度を戻します。
getScale()	小数値のスケールを取得します。int 型のスケールを戻します。

compareTo()

説明

2 つの値を比較します。結果は int 型で戻され、小さい場合は -1、等しい場合は 0、大きい場合は 1 が戻されます。

構文

```
public int compareTo(XSDDataValue val)
```

例外

XSDException ― データ値を比較できない場合

getLength()

説明

STRING/BINARY 値の長さを取得します。int 型の長さを戻します。

構文

```
public int getLength()
```

例外

XSDException ― データ値が String/Binary 型でない場合

getLexicalValue()

説明

XSDDataValue クラスから LEXICAL 値を取得します。文字列を戻します。

構文

```
public java.lang.String getLexicalValue()
```

getPrecision()

説明

小数値の精度を取得します。int 型の精度を戻します。

構文

```
public int getPrecision()
```

例外

XSDException ― データ値が小数型でない場合

getScale()

説明

小数値のスケールを取得します。int 型のスケールを戻します。

構文

```
public int getScale()
```

例外

XSDException ― データ値が小数型でない場合

XSDElement クラス

XSDElement の説明

XSDElement クラス。要素の XMLSchema 定義を表します。

XSDElement の構文

```
public class XSDElement
```

oracle.xml.parser.schema.XSDElement

XSDElement のメソッド

表 7-18 XSDElement のメソッドの概要

メソッド	説明
findEquivClass()	このクラスに対応する等価のクラスを検索します。
getDefaultVal()	要素の場合は、default 属性の値と use 属性に基づいた value 属性の値を取得します。
getEquivClassRef()	解決済み派生クラスのローカル名を取得します。
getFixedVal()	要素の場合は、fixed 属性の値と use 属性に基づいた value 属性の値を取得します。
getIdentities()	識別のセットを戻します。
getMaxOccurs()	maxOccurs を取得します。
getMinOccurs()	minOccurs を取得します。
getName()	名前を取得します。
getRefLocalname()	解決済み ref 属性のローカル名を取得します。
getRefNamespace()	解決済み ref 属性の名前空間を取得します。
getRefState()	refState を取得します。
getSubstitutionGroup()	substitutionGroup を取得します。
getTargetNS()	ターゲット名前空間を取得します。
getType()	ノード・タイプを取得します。
isAbstract()	抽象であるかどうかを true または false で宣言します。

表 7-18 XSDElement のメソッドの概要（続き）

メソッド	説明
isNullable()	null 値が可能であるかどうかを true または false で宣言します。
setMaxOccurs()	maxOccurs を設定します。
setMinOccurs()	minOccurs を設定します。

findEquivClass()

説明

このクラスに対応する等価のクラスを検索します。

構文

```
public XSDElement findEquivClass(java.lang.String ns, java.lang.String nm)
```

パラメータ

表 7-19 findEquivClass のパラメータ

パラメータ	説明
ns	クラスの名前空間
nm	クラスの名前

戻り値

XSDElement

getDefaultVal()

説明

要素の場合は、default 属性の値と use 属性に基づいた value 属性の値を取得します。

構文

```
public String getDefaultVal()
```

戻り値

defaultVal

getEquivClassRef()

説明
解決済み等価クラスのローカル名を取得します。

構文
```
public String getEquivClassRef()
```

戻り値
equivRefLocalname

getFixedVal()

説明
要素の場合は、fixed 属性の値と use 属性に基づいた value 属性の値を取得します。

構文
```
public java.lang.String getFixedVal()
```

戻り値
defaultVal

getIdentities()

説明
識別のセットを戻します。

構文
```
public XSDIdentity[] getIdentities()
```

戻り値
識別の配列

getMaxOccurs()

説明
maxOccurs を取得します。

構文
```
public int getMaxOccurs()
```

戻り値
maxOccurs

getMinOccurs()

説明
minOccurs を取得します。

構文
```
public int getMinOccurs()
```

戻り値
minOccurs 値

getName()

説明
ノードの名前を取得します。

構文
```
public String getName()
```

戻り値
ノード名

getRefLocalname()

説明
解決済み ref 属性のローカル名を取得します。

構文
```
public String getRefLocalname()
```

戻り値
refLocalname

getRefNamespace()

説明
解決済み ref 属性の名前空間を取得します。

構文
```
public String getRefNamespace()
```

戻り値
refNamespace

getRefState()

説明
refState を取得します。戻り値は、TYPE_UNRESOLVED、TYPE_RESOLVED、REF_UNRESOLVED、REF_RESOLVED のいずれかです。

構文
```
public int getRefState()
```

戻り値
refstate 値

getSubstitutionGroup()

説明

substitutionGroup を取得します。

構文

```
public java.util.Vector getSubstitutionGroup()
```

getTargetNS()

説明

ターゲット名前空間を取得します。

構文

```
public java.lang.String getTargetNS()
```

getType()

説明

ノード・タイプを取得します。

構文

```
public XSDNode getType()
```

戻り値

nodeType（simpleType または complexType）

isAbstract()

説明

抽象かどうかを宣言します。

構文

```
public boolean isAbstract()
```

isNullable()

説明

null 値可能かどうかを宣言します。

構文

```
public boolean isNullable()
```

setMaxOccurs()

説明

maxOccurs を設定します。

構文

```
public void setMaxOccurs(int max)
```

パラメータ

表 7-20 setMaxOccurs のパラメータ

パラメータ	説明
max	値

setMinOccurs()

説明

minOccurs を設定します。

構文

```
public void setMinOccurs(int min)
```

パラメータ

表 7-21 setMinOccurs のパラメータ

パラメータ	説明
min	値

XSDException

XSDException の説明

XMLSchema の妥当性チェック時に例外が発生したことを示します。

XSDException の構文

```
java.lang.Object
   |
   +---java.lang.Throwable
           |
           +---java.lang.Exception
                   |
                   +---oracle.xml.parser.schema.XSDException

public class XSDException
extends Exception
```

getMessage()

説明

エラー ID およびエラー・パラメータからエラー・メッセージを作成するために、クラス Throwable の getMessage をオーバーライドします。表 7-22 で、各オプションについて説明します。

表 7-22 getMessage() のバージョン

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られた 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

表 7-23 getMessage() のパラメータ

パラメータ	説明
err	エラー・メッセージの取得に使用される XMLError クラス

XSDGroup クラス

XSDGroup の説明

XSDGroup は、XMLSchema のモデル・グループを表します。モデル・グループには、別のモデル・グループまたは小要素が含まれる場合があります。

XSDGroup の構文

```
public class XSDGroup
```

oracle.xml.parser.schema.XSDGroup

XSDGroup のメソッド

表 7-24 XSDIdentity のメソッドの概要

メソッド	説明
getMaxOccurs()	maxOccurs を取得します。
getMinOccurs()	minOccurs を取得します。
getNodeVector()	nodeVector に格納されたグループの小要素を取得します。
getOrder()	コンポジット型（ALL、SEQUENCE、CHOICE）を取得します。
setMaxOccurs()	maxOccurs を設定します。
setMinOccurs()	minOccurs を設定します。

getMaxOccurs()

説明

maxOccurs を取得します。

構文

```
public int getMaxOccurs()
```

戻り値

maxOccurs

getMinOccurs()

説明
minOccurs を取得します。

構文
```
public int getMinOccurs()
```

戻り値
minOccurs

getNodeVector()

説明
nodeVector に格納されたグループの小要素を取得します。

構文
```
public java.util.Vector getNodeVector()
```

戻り値
nodeVector

getOrder()

説明
コンポジット型（ALL、SEQUENCE または CHOICE）を取得します。

構文
```
public int getOrder()
```

戻り値
順序

setMaxOccurs()

説明

maxOccurs を設定します。

構文

```
public void setMaxOccurs(int max)
```

パラメータ

表 7-25 setMaxOccurs のパラメータ

パラメータ	説明
max	値

setMinOccurs()

説明

minOccurs を設定します。

構文

```
public void setMinOccurs(int min)
```

パラメータ

表 7-26 setMinOccurs のパラメータ

パラメータ	説明
min	値

XSDIdentity クラス

XSDIdentity の説明

XSDIdentity は、XMLSchema に対する XSD の識別パラメータを示します。

XSDIdentity の構文

```
public class XSDIdentity extends oracle.xml.parser.schema.XSDNode

oracle.xml.parser.schema.XSDNode
  |
  +--oracle.xml.parser.schema.XSDIdentity
```

XSDIdentity のメソッド

表 7-27 XSDIdentity のメソッドの概要

メソッド	説明
getFields()	フィールドを取得します。
getNodeType()	ノード・タイプを取得します。
getRefer()	参照キーを取得します。
getSelector()	セレクタを取得します。

getFields()

説明

フィールドを取得します。

構文

```
public java.lang.String[] getFields()
```

戻り値

フィールド

getNodeType()

説明

ノード・タイプを取得します。クラス XSDNode の XSDNode.getNodeType() をオーバーライドします。

構文

```
public int getNodeType()
```

戻り値

nodeType

getRefer()

説明

参照キーを取得します。

構文

```
public java.lang.String getRefer()
```

戻り値

参照キー

getSelector()

説明

セレクタを取得します。

構文

```
public java.lang.String getSelector()
```

戻り値

セレクタ

XSDNode クラス

XSDNode の説明

大部分の XSD クラスのルート・クラスです。XMLSchema 定義属性に対応するフィールドおよびメソッドが含まれます。

XSDNode の構文

```
public class XSDNode

oracle.xml.parser.schema.XSDNode
```

XSDNode のダイレクト・サブクラス

XMLSchema, XMLSchemaNode, XSDAttribute, XSDComplexType, XSDIdentity

XSDNode のメソッド

表 7-28 XSDNode のメソッドの概要

メソッド	説明
getName()	ノードの名前を取得します。
getNamespaceURI()	名前空間の URI を取得します。
getNodeType()	XSDNode のタイプを取得します。
getTargetNS()	ターゲット名前空間を取得します。
isNodeType()	ノードが指定のタイプかどうかをチェックします。

getName()

説明

ノードの名前を取得します。

構文

```
public java.lang.String getName()
```

戻り値

ノード名

getNamespaceURI()

説明
名前空間の URI を取得します。

構文
```
public java.lang.String getNamespaceURI()
```

戻り値
targetNS

getNodeType()

説明
XSDNode のタイプを取得します。

構文
```
public int getNodeType()
```

戻り値
nodeType

getTargetNS()

説明
ターゲット名前空間を取得します。

構文
```
public java.lang.String getTargetNS()
```

戻り値
targetNS

isNodeType()

説明

ノードが指定のタイプかどうかをチェックします。

構文

```
public boolean isNodeType(int type)
```

パラメータ

表 7-29 isNodeType のパラメータ

パラメータ	説明
type	チェックするノードのタイプ

XSDSimpleType クラス

XSDSimpleType の説明

型を派生する XSDTypeContstants を実装します。

XSDSimpleType の構文

```
public class XSDSimpleType implements oracle.xml.parser.schema.XSDTypeConstants
```

oracle.xml.parser.schema.XSDSimpleType

XSDSimpleType の実装されるインタフェース

XSDTypeConstants

XSDSimpleType のメソッド

表 7-30 XSDSimpleType のメソッドの概要

メソッド	説明
XSDSimpleType()	クラス・コンストラクタです。
derivedFrom()	指定したベース型から型を派生します。
getBase()	ベース型を取得します。
getBasicType()	型の派生元の基本型を取得します。
getBuiltInDatatypes()	組込みデータ型を取得します。
getFacets()	ファセットを取得します。
getMaxOccurs()	maxOccurs の値を取得します。
getMinOccurs()	minOccurs の値を取得します。
getVariety()	型の種類を取得します。
isAbstract()	抽象かどうかをブール値 true または false で宣言します。
setFacet()	データ型（内部プライベート API）のファセットを設定します。
setMaxOccurs()	maxOccurs の値を設定します。
setMinOccurs()	minOccurs の値を設定します。

表 7-30 XSDSimpleType のメソッドの概要（続き）

メソッド	説明
setSource()	データ型のベース型を設定します。または集約型の場合は、その集約型のコンポーネントの型を設定します。
validateValue()	この型に定義されたファセットで文字列値を検証します。

XSDSimpleType()

説明

クラス・コンストラクタです。表 7-31 で、各オプションについて説明します。

表 7-31 XSDSimpleType コンストラクタのバージョン

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 nm)	派生されます。

derivedFrom()

説明

指定したベース型から型を派生します。

構文

```
public static XSDSimpleType derivedFrom(XSDSimpleType source, String nm, String var)
```

パラメータ

表 7-32 derivedFrom のパラメータ

パラメータ	説明
source	XSDSimpleType: ベース型
nm	文字列 : 新しい型の名前
var	文字列 : 派生方法

例外

XSDException ― 新しい型を作成できなかった場合

getBase()

説明
この型の派生元のベース型を取得します。

構文
```
public XSDSimpleType getBase()
```

getBasicType()

説明
型の派生元の基本型を取得します。

構文
```
public int getBasicType()
```

戻り値
basicType

getBuiltInDatatypes()

説明
組込みデータ型を取得します。

構文
```
public static Hashtable getBuiltInDatatypes()
```

例外
XSDException ― 型の名前が有効でない場合

getFacets()

説明
このデータ型のファセットを取得します。

構文
```
public XSDConstrainingFacet getFacets()
```

戻り値
ファセット

getMaxOccurs()

説明
maxOccurs の値を取得します。

構文
```
public int getMaxOccurs()
```

戻り値
1

getMinOccurs()

説明
minOccurs の値を取得します。

構文
```
public int getMinOccurs()
```

戻り値
1

getVariety()

説明
型の種類を取得します。

構文
```
public java.lang.String getVariety()
```

戻り値
様々な型

isAbstract()

説明
抽象かどうかをブール値 true または false で宣言します。

構文
```
public boolean isAbstract()
```

setFacet()

説明
データ型（内部プライベート API）のファセットを設定します。

構文
```
public void setFacet(String facetName, String value)
```

パラメータ

表 7-33 setFacet のパラメータ

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

例外
XSDException ― ファセットが無効な場合

setMaxOccurs()

説明

maxOccurs の値を設定します。

構文

```
public void setMaxOccurs(int max)
```

パラメータ

表 7-34 setMaxOccurs のパラメータ

パラメータ	説明
max	最大発生件数

setMinOccurs()

説明

minOccurs の値を設定します。

構文

```
public void setMinOccurs(int min)
```

パラメータ

表 7-35 setMinOccurs のパラメータ

パラメータ	説明
min	最小発生件数

setSource()

説明

データ型のベース型を設定します。または集約型の場合は、集約型のコンポーネントの型を設定します。

構文

```
public void setSource(XSDNode src)
```

パラメータ

表 7-36 setSource のパラメータ

パラメータ	説明
src	XSDNode ソース

例外

XSDException ― ソースの型が有効でない場合

validateValue()

説明

この型に定義されたファセットで文字列値を検証します。

構文

```
public void validateValue(java.lang.String val)
```

パラメータ

表 7-37 validateValue のパラメータ

パラメータ	説明
val	検証する値

例外

XSDException ― 値が有効でない場合

XSDTypeConstants インタフェース

XSDTypeConstants の説明

XSDTypeConstants のインタフェースを実装します。

XSDTypeConstants の構文

```
public interface XSDTypeConstants
```

XSDTypeConstants の実装クラス

```
XSDDataValue, XSDConstrainingFacet, XSDSimpleType
```

XSDTypeConstants のフィールド

表 7-38 XSDTypeConstants のフィールド

フィールド	構文
_atomic	public static final java.lang.String _atomic
_base64	public static final java.lang.String _base64
_collapse	public static final java.lang.String _collapse
_hex	public static final java.lang.String _hex
_preserve	public static final java.lang.String _preserve
_replace	public static final java.lang.String _replace
ANY_SIMPLE	public static final java.lang.String ANY_SIMPLE
ANY_URI	public static final java.lang.String ANY_URI
BASE64_BINARY	public static final java.lang.String BASE64_BINARY
BINARY	public static final java.lang.String BINARY
BOOLEAN	public static final java.lang.String BOOLEAN
BYTE	public static final java.lang.String BYTE
CDATA	public static final java.lang.String CDATA
CENTURY	public static final java.lang.String CENTURY
DATE	public static final java.lang.String DATE
DATE_TIME	public static final java.lang.String DATE_TIME

表 7-38 XSDTypeConstants のフィールド（続き）

フィールド	構文
DECIMAL	public static final java.lang.String DECIMAL
DOUBLE	public static final java.lang.String DOUBLE
DURATION	public static final java.lang.String DURATION
ENCODING	public static final java.lang.String ENCODING
ENTITIES	public static final java.lang.String ENTITIES
ENTITY	public static final java.lang.String ENTITY
ENUMERATION	public static final java.lang.String ENUMERATION
FLOAT	public static final java.lang.String FLOAT
FRACTION_DIGITS	public static final java.lang.String FRACTION_DIGITS
GDAY	public static final java.lang.String GDAY
GMONTH	public static final java.lang.String GMONTH
GMONTH_DAY	public static final java.lang.String GMONTH_DAY
GYEAR	public static final java.lang.String GYEAR
GYEAR_MONTH	public static final java.lang.String GYEAR_MONTH
HEX_BINARY	public static final java.lang.String HEX_BINARY
iANY_SIMPLE	public static final int iANY_SIMPLE
iANY_URI	public static final int iANY_URI
iBASE64_BINARY	public static final int iBASE64_BINARY
iBOOLEAN	public static final int iBOOLEAN
ID	public static final java.lang.String ID
iDATE	public static final int iDATE
iDATE_TIME	public static final int iDATE_TIME
iDECIMAL	public static final int iDECIMAL
iDOUBLE	public static final int iDOUBLE
IDREF	public static final java.lang.String IDREF
IDREFS	public static final java.lang.String IDREFS
iDUMMY	public static final int iDUMMY
iDURATION	public static final int iDURATION

表 7-38 XSDTypeConstants のフィールド（続き）

フィールド	構文
iENUMERATION	public static final int iENUMERATION
iFLOAT	public static final int iFLOAT
iFRACTION_DIGITS	public static final int iFRACTION_DIGITS
iGDAY	public static final int iGDAY
iGMONTH	public static final int iGMONTH
iGMONTH_DAY	public static final int iGMONTH_DAY
iGYEAR	public static final int iGYEAR
iGYEAR_MONTH	public static final int iGYEAR_MONTH
iHEX_BINARY	public static final int iHEX_BINARY
iLENGTH	public static final int iLENGTH
iMAXEXCLUSIVE	public static final int iMAXEXCLUSIVE
iMAXINCLUSIVE	public static final int iMAXINCLUSIVE
iMAXLENGTH	public static final int iMAXLENGTH
iMINEXCLUSIVE	public static final int iMINEXCLUSIVE
iMININCLUSIVE	public static final int iMININCLUSIVE
iMINLENGTH	public static final int iMINLENGTH
iNOTATION	public static final int iNOTATION
INT	public static final java.lang.String INT
INTEGER	public static final java.lang.String INTEGER
iPATTERN	public static final int iPATTERN
iQNAME	public static final int iQNAME
iSTRING	public static final int iSTRING
iTIME	public static final int iTIME
iTOTAL_DIGITS	public static final int iTOTAL_DIGITS
iWHITESPACE	public static final int iWHITESPACE
LANGUAGE	public static final java.lang.String LANGUAGE
LENGTH	public static final java.lang.String LENGTH
LONG	public static final java.lang.String LONG

表 7-38 XSDTypeConstants のフィールド（続き）

フィールド	構文
MAXEXCLUSIVE	public static final java.lang.String MAXEXCLUSIVE
MAXINCLUSIVE	public static final java.lang.String MAXINCLUSIVE
MAXLENGTH	public static final java.lang.String MAXLENGTH
MINEXCLUSIVE	public static final java.lang.String MINEXCLUSIVE
MININCLUSIVE	public static final java.lang.String MININCLUSIVE
MININCLUSIVE	public static final java.lang.String MININCLUSIVE
MINLENGTH	public static final java.lang.String MINLENGTH
MONTH	public static final java.lang.String MONTH
N_STRING	public static final java.lang.String N_STRING
NAME	public static final java.lang.String NAME
NCNAME	public static final java.lang.String NCNAME
NEGATIVE_INTEGER	public static final java.lang.String NEGATIVE_INTEGER
nFacets	public static final int nFacets
NMTOKEN	public static final java.lang.String NMTOKEN
NMTOKENS	public static final java.lang.String NMTOKENS
NON_NEGATIVE_INTEGER	public static final java.lang.String NON_NEGATIVE_INTEGER
NON_POSITIVE_INTEGER	public static final java.lang.String NON_POSITIVE_INTEGER
PATTERN	public static final java.lang.String PATTERN
PERIOD	public static final java.lang.String PERIOD
POSITIVE_INTEGER	public static final java.lang.String POSITIVE_INTEGER
PRECISION	public static final java.lang.String PRECISION
QNAME	public static final java.lang.String QNAME
RECURRING_DATE	public static final java.lang.String RECURRING_DATE
RECURRING_DAY	public static final java.lang.String RECURRING_DAY
RECURRING_DURATION	public static final java.lang.String RECURRING_DURATION
SCALE	public static final java.lang.String SCALE
sFacets	public static final java.lang.String[] sFacets
SHORT	public static final java.lang.String SHORT

表 7-38 XSDTypeConstants のフィールド（続き）

フィールド	構文
SNOTATION	public static final java.lang.String SNOTATION
STRING	public static final java.lang.String STRING
sTypes	public static final java.lang.String[] sTypes
TIME	public static final java.lang.String TIME
TIME_DURATION	public static final java.lang.String TIME_DURATION
TIME_INSTANT	public static final java.lang.String TIME_INSTANT
TIME_PERIOD	public static final java.lang.String TIME_PERIOD
TOKEN	public static final java.lang.String TOKEN
TOTAL_DIGITS	public static final java.lang.String TOTAL_DIGITS
UNSIGNED_BYTE	public static final java.lang.String UNSIGNED_BYTE
UNSIGNED_INT	public static final java.lang.String UNSIGNED_INT
UNSIGNED_LONG	public static final java.lang.String UNSIGNED_LONG
UNSIGNED_SHORT	public static final java.lang.String UNSIGNED_SHORT
URI_REFERENCE	public static final java.lang.String URI_REFERENCE
WHITESPACE	public static final java.lang.String WHITESPACE
YEAR	public static final java.lang.String YEAR

XSDValidator クラス

XSDValidator の説明

XSDValidator は、XMLSchema と照合してインスタンス XML 文書を検証します。

- 登録時に、XSDValidator オブジェクトは、XMLParser と XMLDocument イベント・ハンドラ（SAXHandler または DOMBuilder）の間のパイプライン・ノードとして挿入されます。
- このオブジェクトは、startElement、characters および endElement の 3 つのイベントとともに機能します。定義されている場合は、デフォルトの要素と属性値が（XMLSchema の情報セットへの追加と同様に) イベント・コンテンツに追加され、上位に伝播されます。
- XMLSchema オブジェクトは、`[element(name)] -> [snode(min/maxOccurs)] -> [type(group/simpleType)]` という構造の要素宣言のセットまたはグループです。
- XSDValidator は、スタック・ベース状態のマシンとして実装されます。各状態は、要素タイプ（group または simpleType）を表します。
- （group としての）XMLSchema オブジェクトは、初期状態でロードされます。現行要素（イベント startElement）は、現行の状態グループ要素と照合されます。一致した場合は、要素タイプ、要素名および snode 情報が新規状態としてロードされます。
- グループの場合は、カウンタのベクトル（int 型）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、要素発生のカウントに使用されます。
  - 状態ステータスは次のとおりです。

    `NEW_STATE`: ロードされたのみで、試行されていません。

    `ACCEPTED`: minOccurs が満たされています。要素の発生を受け入れることができます。

    `DONE`: maxOccurs が満たされています。要素の発生を受け入れることはできません。
- テキスト要素の内容（イベント文字）が simpleType（メソッド validateValue）と照合されます。終了要素（イベント endElement）が名前付けされた最後の状態と照合されます。
- XMLSchema 属性は、特別な要素（<_attrTag> attrType ...）のコンテンツを形成するグループ（attrName -> attrType）として表されます。SMLParser は、それに従って属性（イベント startElement）を変換します（メソッド startElement を参照）。
- 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（'any' 要素の XMLSchema 定義を参照）。

- エラーの場合、またはワイルド・カード（'any'）コンテンツがスキップされた場合は、見せかけの状態がロードされます。

XSDValidator の構文

```
public class XSDValidator
```

oracle.xml.parser.schema.XSDValidator

XSDValidator のメソッド

表 7-39 XSDValidator のメソッドの概要

メソッド	説明
XSDValidator()	クラス・コンストラクタです。
characters()	要素内の文字データの通知を伝播します。
endElement()	要素終了の通知を受け取ります。
setDocumentLocator()	ドキュメント・イベントに関するロケータ・オブジェクトを伝播します。
setError()	現行エラーとして XMLError オブジェクトを設定します。
setXMLProperties()	ランタイム・プロパティ用の XML プロパティを設定します。
setXMLProperty()	プロパティを設定します。
startElement()	要素開始の通知を受け取ります。

XSDValidator()

説明

XSDValidator のコンストラクタ

構文

```
public  XSDValidator()
```

characters()

説明

要素内の文字データの通知を伝播します。 org.xml.sax.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

表 7-40 endElement のパラメータ

パラメータ	説明
ch	文字
start	文字配列の開始位置
length	文字配列から使用する文字数

例外

org.xml.sax.SAXException － SAX 例外で、別の例外をラップしている場合もあります。

endElement()

説明

要素終了の通知を受け取ります。

構文

```
public void endElement(String namespaceURI, String localName, String qName)
```

パラメータ

表 7-41 endElement のパラメータ

パラメータ	説明
namespaceURI	名前空間の URI。要素に名前空間の URI がないか、名前空間処理が実行されていない場合は空文字列
localName	ローカル名（接頭辞なし）。名前空間処理が実行されていない場合は空文字列
qName	XML 1.0 修飾名（接頭辞付き）。修飾名が使用できない場合は空文字列

例外

org.xml.sax.SAXException − SAX 例外で、別の例外をラップしている場合もあります。

setDocumentLocator()

説明

ドキュメント・イベントに関するロケータ・オブジェクトを伝播します。

org.xml.sax.DocumentHandler、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

表 7-42 setDocumentLocator のパラメータ

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

setError()

説明

現行エラーとして XMLError オブジェクトを設定します。

構文

```
public void setError(oracle.xml.parser.v2.XMLError he)
```

パラメータ

表 7-43 setError のパラメータ

パラメータ	説明
he	XMLError オブジェクト

例外

SAXException − SAXException が発生する場合があります。

setXMLProperties()

説明

ランタイム・プロパティ用の XML プロパティを設定します。

構文

```
public void setXMLProperties(XMLProperties xmlProp)
```

パラメータ

表 7-44 setXMLProperties のパラメータ

パラメータ	説明
xmlProp	XMLProperties
value	プロパティの値

setXMLProperty()

説明

プロパティを設定します。正常に設定された場合は、プロパティの値が戻されます。プロパティが読取り専用で設定できない、またはプロパティがサポートされていない場合は、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

表 7-45 setXMLProperty のパラメータ

パラメータ	説明
name	プロパティの名前
value	プロパティの値

戻り値

Object	設定されたプロパティ

startElement()

説明

要素開始の通知を受け取ります。

endElement(String, String, String)、org.xml.sax.Attributes も参照してください。

構文

```
public void startElement(String namespaceURI, String localName, String qName,
Attributes atts)
```

パラメータ

表 7-46 startElement のパラメータ

パラメータ	説明
namespaceURI	名前空間の URI。要素に名前空間の URI がないか、名前空間処理が実行されていない場合は空文字列
localName	ローカル名（接頭辞なし）。名前空間処理が実行されていない場合は空文字列
qName	修飾名（接頭辞付き）。修飾名が使用できない場合は空文字列。
atts	要素に連結されている属性。属性がない場合は、空の Attributes オブジェクトです。

例外

org.xml.sax.SAXException − SAX 例外で、別の例外をラップしている場合もあります。

8

パッケージ oracle.xml.sql.dml

この章では、パッケージ oracle.xml.sql.dml に含まれているクラスについて説明します。このクラスは、XML SQL Utility for Java（XSU）に関するデータの操作と変更を処理します。XSU は、Oracle XDK for Java の一部です。XML SQL Utility for Java は、SQL 問合せ、結果セットまたは表から、データベースとの間で XML データの生成と格納を行います。SQL 問合せ結果の XML への標準的なマッピング、およびその逆のマッピングを行うことによって、データ変換を実行します。

この章は、次の項で構成されています。

パッケージ oracle.xml.sql.dml の説明

パッケージ oracle.xml.sql.dml は、Oracle XDK for Java に関するデータ操作と変更の機能を実装します （DML はデータ操作 / 変更言語を表します）。DML 操作用のメソッドは、このパッケージに含まれている OracleXMLSave クラスで提供されます。この OracleXMLSave 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標準的なマッピングをサポートします。

関連項目： 第 9 章「パッケージ oracle.xml.sql.query」

OracleXMLSave クラス

説明

OracleXMLSave 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標準的なマッピングをサポートします。このクラスは、挿入、更新および削除をサポートします。ユーザーはまず、DML 操作を実行する必要がある表の名前を渡して、クラスを作成します。その後、ユーザーはこの表の挿入、更新および削除を自由に使用できます。

このクラスには、更新または削除対象のキー列を識別したり、更新対象の列を制限するために役立つ便利な関数が多数含まれています。

構文

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

java.lang.Object
  |
  +--oracle.xml.sql.dml.OracleXMLSave
```

フィールド

表 8-1 OracleXMLSave のフィールドの概要

フィールド	構文	説明
DATE_FORMAT	public static final java.lang.String DATE_FORMAT	setDateFormat で使用する日付書式
DEFAULT_BATCH_SIZE	public static int DEFAULT_BATCH_SIZE	デフォルトの挿入バッチ・サイズは 17 です。
xDocIsEsc	public boolean xDocIsEsc	XML 文書で SQL から XML へのエスケープが行われたかどうかを示します。

メソッド

表 8-2 OracleXMLSave のメソッドの概要

メソッド	説明
OracleXMLSave()	Save クラスのパブリック・コンストラクタです。
close()	このオブジェクトに対応付けられているすべてのコンテキストをクローズおよび割当て解除します。
deleteXML()	XML 文書に基づいて、表の行を削除します。
getURL()	ファイル名または URL を指定して、URL オブジェクトを戻します。
insertXML()	指定した表に XML 文書を挿入します。
removeXSLTParam()	最上位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()	DML 操作時に使用するバッチ・サイズを変更します。
setCommitBatch()	コミット・バッチ・サイズを設定します。
setDateFormat()	XSU に対して、XML 文書の日付書式を示します。
setIgnoreCase()	XSU に対して、XML 要素とデータベース列または属性との照合は大 / 小文字の区別なしで実行するように指示します。
setKeyColumnList()	更新時または削除時にデータベース表の特定の行を識別するのに使用される列のリストを設定します。
setPreserveWhitespace()	空白を保持するかどうかを XSU に指示します。
setRowTag()	各行値に対応する XML 要素を囲むために XML 文書で使用されるタグ名を指定します。
setSQLToXMLNameEscaping()	SQL オブジェクト名から有効な XML 識別子が作成されない場合に、XML タグのエスケープのオン / オフを切り替えます。
setUpdateColumnList()	更新する列値を設定します。
setXSLT()	生成された XML に適用する XSL 変換を登録します。
setXSLTParam()	最上位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()	XML 文書を指定して表を更新します。

OracleXMLSave()

説明

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,
          java.lang.String tabName);
```

パラメータ

oconn	接続オブジェクト（データベースへの接続）
tableName	更新する必要がある表の名前

close()

説明

このオブジェクトに対応付けられているすべてのコンテキストをクローズおよび割当て解除します。

構文

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

deleteXML()

説明

XML 文書に基づいて、表の行を削除します。処理した XML ROW 要素の数を戻します。XML 文書を通じて選択した行が表の行を一意に識別したかどうかによって、削除されたデータベース行の数と等しくなる場合とならない場合があります。

デフォルトの削除処理では、すべての要素値と対応する列名が照合されます。入力ドキュメントの各 ROW 要素は、表ではそれぞれ別の削除文として処理されます。setKeyColumnList() を使用して、削除する行の識別で照合が必要な列のリストを設定します。他の要素は無視されます。照合を利用できる場合は（削除文がキャッシュに格納されるため）、この方法によって表から複数の行を効率的に削除できます。この方法を使用しない場合は、入力ドキュメントの ROW 要素ごとに新しい削除文を作成する必要があります。次の表で、構文の各オプションについて説明します。

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 文書はリーダー形式です。
public int deleteXML(java.lang.String xmlDoc);	XML 文書は文字列形式です。
public int deleteXML (java.net.URL url);	XML 文書は、URL を使用してアクセスされます。

パラメータ

doc	DOM 形式の XML 文書
xmlStream	ストリーム形式の XML 文書
xmlReader	リーダー形式の XML 文書
xmlDoc	文字列形式の XML 文書
url	表の行を削除するために使用するドキュメントの URL

getURL()

説明

ファイル名または URL を指定して、ターゲット・エンティティを識別する URL オブジェクトを戻します。渡された引数が有効な URL 形式（http://..." や "file://...）でない場合、このメソッドは "file://" を追加して引数を修正します。null または空の文字列が渡された場合には、null を戻します。

構文

```
public static java.net.URL getURL( java.lang.String target);
```

パラメータ

target	ファイル名または URL 文字列

insertXML()

説明

指定された表に XML 文書を挿入します。挿入した行数を戻します。

- 列名と要素名を照合して表に値を挿入し、入力ドキュメントにないすべての要素について null 値を挿入します。setUpdateColumnList() を使用すると、残りの列には、null 値が挿入されず、かわりにデフォルト値が使用されます。
- すべてのキー列のリストを設定するには、setKeyColumnList() を使用します。
- 更新する列のリストを設定するには、setUpdateColumnList() を使用します。

次の表で、各オプションについて説明します。

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。
public int insertXML(java.io.Reader xmlStream);	リーダーから XML 文書を挿入します。
public int insertXML(java.lang.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

doc	表に行を挿入するための DOM
xmlStream	表に行を挿入するために使用されるデータのストリーム
xmlDoc	表に行を挿入するために使用される文字列
url	表に行を挿入するために使用されるドキュメントの URL

removeXSLTParam()

説明

最上位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ません。

構文

```
public void removeXSLTParam( java.lang.String name);
```

パラメータ

name	パラメータ名

setBatchSize()

説明

DML 操作時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ときには、操作をバッチで処理し、I/O サイクルを最小化することをお薦めします。ただし、操作中は、バインド値の格納で多くのキャッシュが必要になります。バッチ処理を使用した場合、コミットはバッチ単位で行われます。バッチ内のいずれかの文の実行が失敗すると、そのバッチ全体がロールバックされます。このような動作を避ける場合は、バッチ・サイズを 1 に設定してください。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

size	すべての DML に使用するバッチ・サイズ

setCommitBatch()

説明

コミット・バッチ・サイズを設定します。このサイズとはレコード数を示すもので、挿入されたレコードがこの数に達すると、コミットが実行されます。size < 1 の場合またはセッションが「自動コミット」モードの場合、XSU は明示的なコミットを行いません。デフォルトのコミット・バッチ・サイズは 0 です。

構文

```
public void setCommitBatch( int size);
```

パラメータ

size	コミット・バッチ・サイズ

setDateFormat()

説明

XSU に対して、XML 文書の日付書式を示します。デフォルトで、OracleXMLSave は日付書式を 'MM/dd/yyyy HH:mm:ss' とみなします。この関数をコールすれば、デフォルトの日付書式をオーバーライドできます。日付書式パターンの構文（つまり、日付マスク）は、java.text.SimpleDateFormat クラスの要件に適合している必要があります。マスクを null または空の文字列に設定すると、デフォルトのマスクである OracleXMLSave.DATE_FORMAT が使用されます。

構文

```
public void setDateFormat( java.lang.String mask);
```

パラメータ

mask	日付マスク

setIgnoreCase()

説明

XSU は、要素名（XML タグ）に基づいてデータベース列または属性に XML 要素をマッピングします。この関数は、大 / 小文字を区別しないで照合を実行するように XSU に指示します。これは、Save オブジェクトの作成時に実行されるメタデータのキャッシュに影響を及ぼすことがあります。

構文

```
public void setIgnoreCase(boolean ignore);
```

パラメータ

ignore	XML 文書のタグの大 / 小文字を無視するかどうかを示すフラグ

setKeyColumnList()

説明

更新時または削除時にデータベース表の特定の行を識別するのに使用される列のリストを設定します。挿入の場合、このコールは無視されます。キー列を設定してから更新を実行する必要があります。削除操作の場合は、オプションです。このキー列を設定すると、更新時または削除時にデータベース行を識別するのに XML 文書のこれらのタグの値が使用されます。現時点では、XML 文書で指定する方法がないため、キー列そのものの値を更新することはできません。

構文

```
public void setKeyColumnList( java.lang.String[] keyColNames);
```

パラメータ

keyColNames	キーとして使用する列のリストの名前

setPreserveWhitespace()

説明

空白を保持するかどうかを XSU に指示します。

構文

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

パラメータ

flag	空白を保持するかどうかを示すフラグ

setRowTag()

説明

各行値に対応する XML 要素を囲むために XML 文書で使用されるタグ名を指定します。この値を null に設定すると、行タグが存在せず、ドキュメントの最上位要素が行そのものに対応することを示します。

構文

```
public void setRowTag( java.lang.String rowTag);
```

パラメータ

rowTag	タグ名

setSQLToXMLNameEscaping()

説明

XML 識別子にマッピングされる SQL オブジェクト名が有効な XML 識別子でない場合に、XML タグのエスケープをオンまたはオフに切り替えます。

構文

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

パラメータ

flag	SQL から XML へのエスケープをオンにするかどうかを示すフラグ

setUpdateColumnList()

説明

更新する列値を設定します。挿入と更新に適用され、削除には適用されません。

- 挿入の場合、デフォルトでは表のすべての列に値が挿入されます。
- 更新の場合、デフォルトでは XML 文書の ROW 要素にあるタグに対応する列のみが更新されます。指定すると、更新文または挿入文で指定された列のみが更新されます。ドキュメント内のその他の要素はすべて無視されます。

構文

```
public void setUpdateColumnList(java.lang.String[] updColNames);
```

パラメータ

updColNames	更新する列の文字列リスト

setXSLT()

説明

生成された XML に適用する XSL 変換を登録します。スタイルシートがすでに登録されている場合は、新しいスタイルシートに置き換わります。スタイルシートの登録を解除するには、`stylesheet` 引数に `null` を渡します。次の表で、各オプションについて説明します。

構文	説明
public void setXSLT(java.io.Reader stylesheet, java.lang.String ref);	スタイルシート・パラメータはデータとして渡されます。
public void setXSLT(java.lang.String stylesheet, java.lang.String ref);	スタイルシート・パラメータは、ドキュメントの URI として渡されます。

パラメータ

stylesheet	スタイルシートの URI
ref	挿入、インポートおよび外部エンティティ用の URL

setXSLTParam()

説明

最上位レベルのスタイルシート・パラメータの値を設定します。パラメータ値は、有効な XPath 式であると予測されます（したがって、文字列リテラル値は明示的に引用する必要があります）。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ません。

構文

```
public void setXSLTParam(java.lang.String name,
                java.lang.String value);
```

パラメータ

name	パラメータ名
value	XPath 式としてのパラメータ値

updateXML()

説明

XML 文書を指定して表を更新します。処理した XML 要素の数を戻します。XML 文書を通じて選択した行が表の行を一意に識別したかどうかによって、変更されたデータベース行の数と等しくなる場合とならない場合があります。

- 更新では、指定した表内の更新する行を一意に識別するためのキー列のリストが必要です。更新では、デフォルトで、キー列のリストを使用して XML 文書の対応する要素の値が照合され、特定の行が識別されます。次に、XML 文書内に同じ要素が存在する表内の列がすべて更新されます。入力ドキュメントの各 ROW 要素は、表に対する別個の更新として処理されます。
- 更新する列のリストを指定すると、目的の列のみが更新されるため、XML 文書内の他の要素を無視できます。入力 XML 文書に複数の行が存在する場合は、更新文そのものがキャッシュに格納され、バッチ処理されるため、これは非常に効率的な方法です。
- 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。

次の表で、各オプションについて説明します。

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(java.lang.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

doc	DOM ツリー形式の XML 文書
xmlStream	ストリーム形式の XML 文書
xmlDoc	文字列形式の XML 文書
url	表の更新に使用するドキュメントの URL

9

パッケージ oracle.xml.sql.query

この章では、oracle.xml.sql.query パッケージに含まれている、XML SQL Utility for Java の Java クラスについて説明します。XML SQL Utility for Java（XSU）は、SQL 問合せから XML を生成し、格納します。

この章は、次の項で構成されています。

パッケージ oracle.xml.sql.query の説明

XML SQL Utility for Java の Java クラスは、oracle.xml.sql.query パッケージに含まれています。XML SQL Utility for Java（XSU）は、SQL 問合せ、結果セットまたは表から、データベースとの間で XML データの生成と格納を行います。SQL 問合せ結果の XML への標準的なマッピング、およびその逆のマッピングを行うことによって、データ変換を実行します。

関連項目： 第 8 章「パッケージ oracle.xml.sql.dml」

OracleXMLQuery クラス

説明

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

java.lang.Object
  |
  +--oracle.xml.sql.query.OracleXMLQuery
```

フィールド

表 9-1　OracleXMLQuery のフィールドの概要

フィールド	構文	説明
DTD	public static final int DTD	DTD の生成を指定します。
ERROR_TAG	public static final java.lang.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	結果にすべての行を含めることを指定します。
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 java.lang.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 java.lang.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 java.lang.String ROWSET_TAG	ドキュメントのデフォルトのタグ名を指定します。
SCHEMA	public static final int SCHEMA	XML Schema が生成されるかどうかを指定します。
SKIPROWS_ALL	public static final int SKIPROWS_ALL	結果ですべての行をスキップすることを指定します。

メソッド

表 9-2　OracleXMLQuery のメソッドの概要

メソッド	説明
9-6 ページの「OracleXMLQuery()」	クラス・コンストラクタです。
9-6 ページの「close()」	Oracle XML エンジンによって作成された、オープンしているリソースをクローズします。
9-6 ページの「getNumRowsProcessed()」	処理する行数を設定します。
9-7 ページの「getXMLDOM()」	コンストラクタに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。
9-7 ページの「getXMLMetaData()」	XML 文書の DTD または XMLSchema を戻します。
9-8 ページの「getXMLSAX()」	コンストラクタに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。
9-8 ページの「getXMLSchema()」	指定された問合せに対応する XMLSchema を生成します。
9-8 ページの「getXMLString()」	コンストラクタに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。
9-9 ページの「keepObjectOpen()」	XML データの取得元であるオブジェクトの永続性のオン / オフを切り替えます。
9-10 ページの「removeXSLTParam()」	最上位レベルのスタイルシート・パラメータの値を削除します。
9-10 ページの「setCollIdAttrName()」	コレクション要素のセパレータ・タグの ID 属性の名前を設定します。
9-10 ページの「setDataHeader()」	XML データ・ヘッダーを設定します。
9-11 ページの「setDateFormat()」	XML 文書の生成された日付の書式を設定します。
9-11 ページの「setEncoding()」	XML 文書のエンコーディング処理命令を設定します。
9-12 ページの「setErrorTag()」	XML エラー・ドキュメントを囲むタグを設定します。
9-12 ページの「setException()」	XSU で処理されるように、ユーザーが例外を渡すことを可能にします。
9-12 ページの「setMaxRows()」	XML に変換する最大行数を設定します。
9-13 ページの「setMetaHeader()」	XML メタ・ヘッダーを設定します。

表 9-2　OracleXMLQuery のメソッドの概要（続き）

メソッド	説明
9-13 ページの「setRaiseException()」	例外が発生した場合に、スローするかどうかを XSU に指示します。
9-13 ページの「setRaiseNoRowsException()」	生成された XML 文書が空のときに、`OracleXMLNoRowsException` を発生させるかどうかを XSU に指示します。
9-14 ページの「setRowIdAttrName()」	行を囲むタグの ID 属性の名前を設定します。
9-14 ページの「setRowIdAttrValue()」	行を囲むタグの ID 属性に割り当てられるスカラー列の値を指定します。
9-14 ページの「setRowsetTag()」	XML データセットを囲むタグを設定します。
9-15 ページの「setSkipRows()」	スキップする行数を設定します。
9-15 ページの「setSQLToXMLNameEscaping()」	マッピングされた SQL オブジェクト名から有効な XML 識別子が作成されない場合に、XML タグのエスケープのオン / オフを切り替えます。
9-16 ページの「setStylesheetHeader()」	スタイルシート・ヘッダーを設定します。
9-16 ページの「setXSLT()」	生成された XML に適用する XSL 変換を登録します。
9-17 ページの「setXSLTParam()」	最上位レベルのスタイルシート・パラメータの値を設定します。
9-17 ページの「useLowerCaseTagNames()」	タグ名を小文字に設定します。
9-17 ページの「useNullAttributeIndicator()」	`null` 値であることを示すために、特別な XML 属性を使用するか、または XML 文書にエンティティを含めないようにするかを指定します。
9-18 ページの「useTypeForCollElemTag()」	コレクション要素のタイプ名を、そのコレクション要素のタグ名として使用するように XSU に指示します。
9-18 ページの「useUpperCaseTagNames()」	タグ名を大文字に設定します。

OracleXMLQuery()

説明

OracleXMLQuery オブジェクトのクラス・コンストラクタ。次の表で、各オプションについて説明します。

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 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, java.lang.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

conn	データベース接続
rset	JDBC 結果セット・オブジェクト
query	SQL 問合せ文字列
dset	データセット

close()

説明

OracleXML エンジンによって作成された、オープンしているリソースをクローズします。ユーザーが提供したインスタンス結果セットはクローズされません。

構文

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

getNumRowsProcessed()

説明

処理する行数を設定します。

構文

```
public long getNumRowsProcessed();
```

getXMLDOM()

説明

コンストラクタに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 DOM 表現を戻します。 次の表で、各オプションについて説明します。

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)	XSU によって XML とともに生成される XML メタデータのタイプを指定するには、引数を使用します。現時点ではこの値は無視され、XML メタデータは生成されません。XML 文書の DOM 表現を戻します。
public org.w3c.dom.Document getXMLDOM(org.w3c.dom.Node root)	`null` 以外の場合、引数は XML 文書のルート要素とみなされます。XML 文書の DOM 表現を戻します。
public org.w3c.dom.Document getXMLDOM(org.w3c.dom.Node root, int metaType)	`null` 以外の場合、`root` 引数は XML 文書のルート要素とみなされます。XSU によって XML とともに生成される XML メタデータのタイプを指定するには、`metaType` 引数を使用します。現時点ではこの値は無視され、XML メタデータは生成されません。XML 文書の DOM 表現を戻します。

パラメータ

metaType	XML メタデータのタイプ（NONE、SCHEMA）
root	新しい XML を追加するルート・ノード

getXMLMetaData()

説明

この関数は、`getXML*()` コール（`getXMLDOM()`、`getXMLSAX()`、`getXMLSchema()`、`getXMLString()` など）によって生成された XML 文書の DTD または XMLSchema を戻します。

構文

```
public java.lang.String getXMLMetaData( int metaType,
                boolean withVer);
```

パラメータ

metaType	生成する XML メタデータのタイプ（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()

説明

コンストラクタに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

sax	登録される ContentHandler オブジェクト

getXMLSchema()

説明

このメソッドは、指定された問合せに対応する XML Schema を生成し、その XML Schema を戻します。

構文

```
public org.w3c.dom.Document[] getXMLSchema();
```

getXMLString()

説明

コンストラクタに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。 次の表で、各オプションについて説明します。

構文	説明
public java.lang.String getXMLString();	引数を使用しません。
public java.lang.String getXMLString(int metaType);	XSU によって XML とともに生成される XML メタデータのタイプを指定するには、`metaType` 引数を使用します。

構文（続き）	説明
public java.lang.String getXMLString(org.w3c.dom.Node root);	null 以外の場合、root 引数は XML 文書のルート要素とみなされます。
public java.lang.String getXMLString(org.w3c.dom.Node root, int metaType)	null 以外の場合、root 引数は XML 文書のルート要素とみなされます。XSU によって XML とともに生成される XML メタデータのタイプを指定するには、metaType 引数を使用します。root 引数が null 以外の場合は、必須であっても DTD は生成されないので注意してください。

パラメータ

metaType	XML メタデータのタイプ（NONE、DTD または SCHEMA、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()

説明

ResultSet オブジェクトで使用されないすべての getXML*() 関数（getXMLDOM()、getXMLSAX()、getXMLSchema()、getXMLString() など）のデフォルト動作は、コールの最後に ResultSet オブジェクトと Statement オブジェクトをクローズすることです。getXML() を繰返しコールして次の行を取得する永続的な機能が必要な場合は、値 true でこの関数をコールして、このデフォルト動作をオフにする必要があります。

OracleXMLQuery で getXML() をコールした後でも、ResultSet オブジェクトと Statement オブジェクトはクローズされなくなります。カーソル状態をクローズするには、close() 関数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

alive	オブジェクトをオープン状態のままにするかどうかを示すパラメータ

removeXSLTParam()

説明

最上位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ません。

構文

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

パラメータ

name	パラメータ名

setCollIdAttrName()

説明

コレクション要素のセパレータ・タグの id 属性の名前を設定します。null または空の文字列を渡すと、row id 属性が省略されます。

構文

```
public void setCollIdAttrName( String attrName);
```

パラメータ

attrName	属性名

setDataHeader()

説明

XML データ・ヘッダーを設定します。このヘッダーは、問合せにより生成された XML エンティティ（つまり、行セット）の先頭に追加される XML エンティティです。2 つのエンティティは、docTag 引数で指定したタグで囲まれます。最後に指定したデータ・ヘッダーが使用されます。header に null を渡すと、パラメータによってデータ・ヘッダーが設定解除されます。

構文

```
public void setDataHeader(  java.io.Reader header,
                java.lang.String docTag);
```

パラメータ

header	ヘッダー
docTag	データ・ヘッダーおよび行セットを囲むタグ

setDateFormat()

説明

XML 文書に生成される日付の書式を設定します。日付書式パターンの構文（つまり、日付マスク）は、java.text.SimpleDateFormat クラスの要件に適合している必要があります。日付マスクを設定解除するには、マスクを null または空の文字列に設定します。

構文

```
public void setDateFormat( java.lang.String mask);
```

パラメータ

mask	日付マスク

setEncoding()

説明

XML 文書にエンコーディング処理命令（PI）を設定します。エンコーディングとして null または空の文字列を指定すると、エンコーディング PI にデフォルトのキャラクタ・セットが指定されます。

構文

```
public void setEncoding(java.lang.String enc)
```

パラメータ

enc	XML 文書のエンコーディング（エンコーディングの IANA 名）

setErrorTag()

説明

XML 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( java.lang.String tag);
```

パラメータ

tag	タグ名

setException()

説明

ユーザーが例外を渡し、XSU で処理することを可能にします。

構文

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

パラメータ

e	XSU によって処理される例外

setMaxRows()

説明

XML に変換する最大行数を設定します。デフォルトでは最大数は設定されていません。最大数の設定を明示的に解除する場合は、MAXROWS_ALL フィールドを指定してください。

構文

```
public void setMaxRows( int rows);
```

パラメータ

rows	生成する行の最大数

setMetaHeader()

説明

XML メタ・ヘッダーを設定します。設定すると、このオブジェクトによって生成された各 XML 文書のメタデータ部分（DTD または XMLSchema）の先頭にヘッダーが挿入されます。最後に指定したメタ・ヘッダーが使用されます。メタ・ヘッダーを設定解除するには、header を null または空の文字列に設定します。

構文

```
public void setMetaHeader( java.io.Reader header);
```

パラメータ

header	ヘッダー

setRaiseException()

説明

例外が発生した場合に、スローするかどうかを XSU に指示します。このコールが実行されないか、flag 引数に false が渡された場合、XSU は SQL 例外を検出し、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

flag	呼び出された例外をスローするかどうかを示すフラグ

setRaiseNoRowsException()

説明

生成された XML 文書が空のときに、OracleXMLNoRowsException を発生させるかどうかを XSU に指示します。デフォルトでは例外は発生しません。

構文

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

パラメータ

flag	データが見つからなかった場合に、OracleXMLNoRowsException を発生させるかどうかを示すフラグ

setRowIdAttrName()

説明

行を囲むタグの ID 属性の名前を設定します。null または空の文字列を渡すと、row id 属性が省略されます。

構文

```
public void setRowIdAttrName( java.lang.String attrName);
```

パラメータ

attrName	属性名

setRowIdAttrValue()

説明

行を囲むタグの ID 属性に割り当てられるスカラー列の値を指定します。null または空の文字列を渡すと、行 ID 属性に行カウント値（0、1、2 など）が割り当てられます。

構文

```
public void setRowIdAttrValue( java.lang.String colName);
```

パラメータ

colName	値が行 ID 属性に割り当てられる列

setRowsetTag()

説明

XML データセットを囲むタグを設定します。

構文

```
public void setRowsetTag( java.lang.String tag);
```

パラメータ

tag	タグ名

setRowTag()

説明

データベース・レコードに該当する XML 要素を囲むタグを設定します。

構文

```
public void setRowTag( java.lang.String tag);
```

パラメータ

tag	タグ名

setSkipRows()

説明

スキップする行数を設定します。デフォルトでは、スキップされる行は 0 です。すべての行をスキップするには、SKIPROWS_ALL を指定します。

構文

```
public void setSkipRows(int rows);
```

パラメータ

rows	スキップする行の数

setSQLToXMLNameEscaping()

説明

XML 識別子にマッピングされる SQL オブジェクト名が有効な XML 識別子でない場合に、XML タグのエスケープのオン / オフを切り替えます。

構文

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

パラメータ

flag	SQL から XML 識別子へのエスケープをオンにするかどうかを示すフラグ

setStylesheetHeader()

説明

生成された XML 文書にスタイルシート・ヘッダー（つまり、スタイルシート処理命令）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・タイプが設定解除されます。次の表で、各オプションについて説明します。

構文	説明
public void setStylesheetHeader(java.lang.String uri);	スタイルシートの URI を使用してスタイルシート・ヘッダーを設定します。
public void setStylesheetHeader(java.lang.String uri, java.lang.String type);	スタイルシートの URI およびスタイルシート・タイプを使用して、スタイルシート・ヘッダーを設定します。

パラメータ

uri	スタイルシートの URI
type	スタイルシートのタイプ。デフォルトは 'text/xsl'

setXSLT()

説明

生成された XML に適用する XSL 変換を登録します。スタイルシートがすでに登録されている場合は、新しいスタイルシートに置き換わります。スタイルシートの登録を解除するには、引数に null 値を渡します。次の表で、各オプションについて説明します。

構文	説明
public void setXSLT(java.io.Reader stylesheet, java.lang.String ref);	スタイルシート・パラメータはデータとして渡されます。
public void setXSLT(j ava.lang.String stylesheet, java.lang.String ref);	スタイルシート・パラメータは、ドキュメントの URI として渡されます。

パラメータ

stylesheet	スタイルシート
ref	挿入、インポートおよび外部エンティティ用の URL

setXSLTParam()

説明

最上位レベルのスタイルシート・パラメータの値を設定します。パラメータ値は、有効な XPath 式であると予測されます、したがって、文字列リテラル値は明示的に引用する必要があります。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ません。

構文

```
public void setXSLTParam( java.lang.String name,
                java.lang.String value);
```

パラメータ

name	パラメータ名
value	XPath 式としてのパラメータ値

useLowerCaseTagNames()

説明

すべてのタグ名の文字を小文字に設定します。目的のタグをすべて設定してから、このコールを実行してください。

構文

```
public void useLowerCaseTagNames();
```

useNullAttributeIndicator()

説明

null 値であることを示すために、特別な XML 属性を使用するか、または XML 文書にエンティティを含めないようにするかを指定します。

構文

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

パラメータ

flag	null 値であることを示すために属性を使用するかどうかを示すフラグ

useTypeForCollElemTag()

説明

デフォルトのコレクションの要素のタグ名は、コレクションのタグ名の後に「_item」が付きます。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タグ名としてコレクション要素のタイプ名を使用します。

構文

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

パラメータ

flag	列要素のタイプを、その列要素のタグ名に使用するかどうかを示すフラグ

useUpperCaseTagNames()

説明

すべてのタグ名を大文字に設定します。目的のタグをすべて設定した後で、このコールを実行してください。

構文

```
public void useUpperCaseTagNames();
```

OracleXMLSQLException クラス

説明

XSU がスローするすべての例外を管理するためのクラスです。

構文

```
public class OracleXMLSQLException extends java.lang.RuntimeException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.lang.RuntimeException
                    |
                    +--oracle.xml.sql.OracleXMLSQLException
```

OracleXMLSQLException のダイレクト・サブクラス

- OracleXMLSQLNoRowsException クラス

OracleXMLSQLException の実装されるインタフェース

- `java.io.Serializable`

OracleXMLSQLException のメソッド

表 9-3 OracleXMLSQLException のメソッドの概要

メソッド	説明
OracleXMLSQLException()	新規 OracleXMLSQLException を作成します。
getErrorCode()	スローされた SQL エラー・コードを戻します。
getParentException()	元の例外があ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。
getXMLErrorString()	指定したエラー・タグ名の XML エラー文字列を出力します。

表 9-3 OracleXMLSQLException のメソッドの概要（続き）

メソッド	説明
getXMLSQLErrorString()	エラー・メッセージに SQL パラメータを出力します。
setErrorTag()	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()

説明

新規 OracleXMLSQLException を作成します。 次の表で、各オプションについて説明します。

構文	説明
public OracleXMLSQLException(java.lang.Exception e);	引数として渡された親例外を設定します。
public OracleXMLSQLException(java.lang.Exception e, java.lang.String errorTagName);	引数として渡されたエラー・タグ名を設定します。
public OracleXMLSQLException(java.lang.String message);	戻されるエラー・メッセージを設定します。
public OracleXMLSQLException(java.lang.String message, java.lang.Exception e);	戻される親例外とエラー・メッセージを設定します。
public OracleXMLSQLException(java.lang.String message, java.lang.Exception e, java.lang.String errorTagName);	使用されるエラー・メッセージ、親例外およびエラー・タグを設定します。
public OracleXMLSQLException(java.lang.String message, int errorCode);	エラー・メッセージと SQL エラー・コードを設定します。
public OracleXMLSQLException(java.lang.String message, int errorCode, java.lang.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(j ava.lang.String message, java.lang.String errorTagName);	使用されるエラー・メッセージとエラー・タグを設定します。

パラメータ

e	例外
errorTagName	エラー・タグ名
message	エラー・メッセージ
errorCode	SQL エラー・コード

getErrorCode()

説明

スローされた SQL エラー・コードを戻します。

構文

```
public int getErrorCode();
```

getParentException()

説明

元の例外がある場合はそれを戻します。それ以外の場合は、null を戻します。

構文

```
public java.lang.Exception getParentException();
```

getXMLErrorString()

説明

指定したエラー・タグ名の XML エラー文字列を出力します。

構文

```
public java.lang.String getXMLErrorString();
```

getXMLSQLErrorString()

説明

エラー・メッセージに SQL パラメータも出力します。

構文

```
public java.lang.String getXMLSQLErrorString();
```

setErrorTag()

説明

エラー・タグ名を設定します。このエラー・タグ名は、getXMLErrorString() および getXMLSQLErrorString() による XML エラー・レポートの生成に使用されます。

構文

```
public void setErrorTag(java.lang.String tagName);
```

パラメータ

tagName	エラーのタグ名

OracleXMLSQLNoRowsException クラス

説明

行が見つからないときにスローされる例外

構文

```
public class OracleXMLSQLNoRowsException extends OracleXMLSQLException

java.lang.Object
 |
 +--java.lang.Throwable
     |
     +--java.lang.Exception
          |
          +--java.lang.RuntimeException
               |
              +--OracleXMLSQLException
                   |
                   +--oracle.xml.sql.OracleXMLSQLNoRowsException
```

OracleXMLSQLNoRowsException の実装されるインタフェース

```
java.io.Serializable
```

メソッド

OracleXMLSQLNoRowsException()

新規 OracleXMLSQLNoRowsException を作成します。次の表で、各オプションについて説明します。

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(java.lang.String errorTag);	引数として渡されたエラー・タグ名を設定します。

パラメータ

errorTag	エラー・タグ

OracleXMLSQLNoRowsException の継承メンバー

表 9-4 OracleCMLSQLNoRowsException の継承メンバーの概要

メンバー	継承元
getErrorCode()	OracleXMLSQLException クラス
getParentException()	OracleXMLSQLException クラス
getXMLErrorString()	OracleXMLSQLException クラス
getXMLSQLErrorString()	OracleXMLSQLException クラス
setErrorTag()	OracleXMLSQLException クラス

10

パッケージ oracle.xml.util

この章では、パッケージ oracle.xml.util について説明します。このパッケージにあるユーティリティ・クラスは、XDK for Java 内の XSLT Processor for Java に対してエラー処理を提供し、拡張機能をサポートします。

XML Parser、DOM および SAX API の完全な機能は oracle.xml.parser.v2 パッケージに含まれています。このパッケージの説明は、このマニュアルの第 11 章「パッケージ oracle.xml.parser.v2」にあります。

この章は、次の項で構成されています。

パッケージ oracle.xml.util の説明

oracle.xml.util に含まれているクラスは、Oracle XDK for Java で提供される、XSLT Processor for Java に対するエラー処理と拡張機能を提供します。このパッケージのクラスは、W3C XML 標準をサポートしています。

Java DOM Parser および XSLT Processor を実装するクラスの説明は、このマニュアルの第 11 章「パッケージ oracle.xml.parser.v2」を参照してください。

関連項目： XDK を使用したアプリケーションの開発に関する情報は、『Oracle9*i* XML Developer's Kit ガイド - XDK』を参照してください。

パッケージ oracle.xml.util の概要

表 10-1 oracle.xml.util パッケージのインタフェースとクラス

インタフェースまたはクラス	説明
NSName	要素名と属性名に対する名前空間サポートを提供するパブリック・インタフェース
XMLError	エラー・メッセージおよびエラーが発生した行番号を保持するクラス
XMLException	XML 文書の処理中に解析例外が発生したことを示すクラス

NSName

説明
パッケージ oracle.xml.util のインタフェース。このインタフェースは、要素名と属性名に対する名前空間サポートを提供します。

構文
```
public interface NSName
```

メソッド

getExpandedName()

説明
この名前の解決済み完全名を取得します。

構文
```
public java.lang.String getExpandedName()
```

戻り値
解決済み完全名

getLocalName()

説明
この名前のローカル名を取得します。

構文
```
public java.lang.String getLocalName()
```

戻り値
ローカル名

getNamespace()

説明
この名前の解決済み名前空間を取得します。

構文
```
public java.lang.String getNamespace()
```

戻り値
解決済み名前空間

getPrefix()

説明
この名前の接頭辞を取得します。

構文
```
public java.lang.String getPrefix()
```

戻り値
接頭辞

getQualifiedName()

説明
修飾名を取得します。

構文
```
public java.lang.String getQualifiedName()
```

戻り値
修飾名

XMLError

説明
このクラスは、エラー・メッセージおよび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

フィールド

表 10-2 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

コンストラクタ

XMLError()

説明
デフォルトのコンストラクタです。

構文
```
public  XMLError()
```

メソッド

表 10-3 oracle.xml.util.XMLError のメソッドの概要

メソッド	説明
error(int, int, String)	新規エラーをベクトルに追加します。
error(int, int, String[])	新規エラーをベクトルに追加します。
error0(int, int)	新規エラーをベクトルに追加します。
error1(int, int, String)	新規エラーをベクトルに追加します。
error2(int, int, String, String)	新規エラーをベクトルに追加します。
error3(int, int, String, String, String)	新規エラーをベクトルに追加します。
flushErrorStream()	すべてのエラーを、デフォルトの出力ストリームまたはエラー・ハンドラにフラッシュします。
formatErrorMesg(int)	
getColumnNumber()	指定されたインデックスにおけるエラーの列番号を取得します。
getException(int)	指定されたインデックスにおけるエラーで発生した例外（存在する場合）を取得します。
getFirstError()	最初のエラーを取得します。
getLineNumber(int)	指定されたインデックスにおけるエラーの行番号を取得します。
getLocator()	登録されたロケータを戻します。
getMessage(int)	指定されたインデックスにおけるエラー・メッセージを取得します。
getMessage(int, String[])	引数が 5 個を超えるエラー・メッセージを取得します。
getMessage0(int)	引数のないエラー・メッセージを取得します。
getMessage1(int, String)	引数が 1 つのエラー・メッセージを取得します。
getMessage3(int, String, String)	引数が 2 つのエラー・メッセージを取得します。
getMessage3(int, String, String, String)	引数が 3 つのエラー・メッセージを取得します。
getMessage4(int, String, String, String, String)	引数が 4 つのエラー・メッセージを取得します。
getMessage5(int, String, String, String, String, String)	引数が 5 つのエラー・メッセージを取得します。

表 10-3 oracle.xml.util.XMLError のメソッドの概要（続き）

メソッド	説明
getMessageType(int)	指定されたインデックスにおけるエラー・メッセージのタイプを取得します。
getNumMessages()	解析中に検出されたエラーまたは警告の合計数を戻します。
getPublicId(int)	指定されたインデックスにおけるエラーが発生したときの入力の公開識別子を取得します。
getSystemId(int)	指定されたインデックスにおけるエラーが発生したときの入力のシステム識別子を取得します。
printErrorListener()	すべての JAXP 1.1 エラーを ErrorListener にフラッシュします。ErrorListener が設定されていない場合は、デフォルトで System.err にフラッシュされます。
reset()	エラー・クラスをリセットします。
setErrorStream(OutputStream)	出力ストリームを登録します。
setErrorStream(OutputStream, String)	出力ストリームを登録します。
setErrorStream(PrintWriter)	出力ストリームを登録します。
setException(Exception)	例外を登録します。
setLocale(Locale)	ロケールを登録します。
setLocator(Locator)	ロケータを登録します。
showWarnings(boolean)	レポートの警告のオン / オフを切り替えます。

error(int, int, String)

説明

新規エラーをベクトルに追加します。

構文

```
public void error(int id, int type, java.lang.String msg)
```

パラメータ

id ― エラー・メッセージの ID

mesg ― エラー・メッセージ（パラメータなし）

type ― エラーのタイプ

error(int, int, String[])

説明

新規エラーをベクトルに追加します。

構文

```
public void error(int id, int type, java.lang.String[] p)
```

パラメータ

id ― エラー・メッセージの ID

type ― エラーのタイプ

p ― パラメータ配列

error0(int, int)

説明

新規エラーをベクトルに追加します。

構文

```
public void error0(int id, int type)
```

パラメータ

id ― エラー・メッセージの ID

type ― エラーのタイプ

error1(int, int, String)

説明

新規エラーをベクトルに追加します。

構文

```
public void error1(int id, int type, java.lang.String p1)
```

パラメータ

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(int, int, String, String)

説明

新規エラーをベクトルに追加します。

構文

```
public void error2(int id, int type, java.lang.String p1, java.lang.String p2)
```

パラメータ

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(int, int, String, String, String)

説明

新規エラーをベクトルに追加します。

構文

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

パラメータ

id ― エラー・メッセージの ID

type ― エラーのタイプ

p1 ― パラメータ 1

p2 ― パラメータ 2

p3 ― パラメータ 3

flushErrorStream()

説明

すべてのエラーを、デフォルトの出力ストリームまたはエラー・ハンドラにフラッシュします。

構文

```
public void flushErrorStream()
```

formatErrorMesg(int)

構文

```
public java.lang.String formatErrorMesg(int index)
```

getColumnNumber(int)

説明

指定されたインデックスにおけるエラーの列番号を取得します。

構文

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

戻り値

列番号

getException(int)

説明

指定されたインデックスにおけるエラーで発生した例外（存在する場合）を取得します。

構文

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

戻り値

例外

getFirstError()

構文

```
public int getFirstError()
```

getLineNumber(int)

説明

指定されたインデックスにおけるエラーの行番号を取得します。

構文

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

戻り値

行番号

getLocator()

説明

登録されたロケータを戻します。

構文

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

戻り値

ロケータ

getMessage(int)

説明

指定されたインデックスにおけるエラー・メッセージを取得します。

構文

```
public java.lang.String getMessage(int i)
```

戻り値

エラー・メッセージ

getMessage(int, String[])

説明

引数が 5 個を超えるエラー・メッセージを取得します。

構文

```
public java.lang.String getMessage(int errId, java.lang.String[] params)
```

getMessage0(int)

説明

引数のないエラー・メッセージを取得します。

構文

```
public java.lang.String getMessage0(int errId)
```

getMessage1(int, String)

説明

引数が 1 つのエラー・メッセージを取得します。

構文

```
public java.lang.String getMessage1(int errId, java.lang.String a1)
```

getMessage2(int, String, String)

説明

引数が 2 つのエラー・メッセージを取得します。

構文

```
public java.lang.String getMessage2(int errId, java.lang.String a1,
java.lang.String a2)
```

getMessage3(int, String, String, String)

説明

引数が 3 つのエラー・メッセージを取得します。

構文

```
public java.lang.String getMessage3(int errId, java.lang.String a1,
java.lang.String a2, java.lang.String a3)
```

getMessage4(int, String, String, String, String)

説明

引数が 4 つのエラー・メッセージを取得します。

構文

```
public java.lang.String getMessage4(int errId, java.lang.String a1,
java.lang.String a2, java.lang.String a3, java.lang.String a4)
```

getMessage5(int, String, String, String, String, String)

説明

引数が 5 つのエラー・メッセージを取得します。

構文

```
public java.lang.String getMessage5(int errId, java.lang.String a1,
java.lang.String a2, java.lang.String a3, java.lang.String a4, java.lang.String a5)
```

getMessageType(int)

説明

指定されたインデックスにおけるエラー・メッセージのタイプを取得します。

構文

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

戻り値

エラー・メッセージのタイプ

getNumMessages()

説明

解析中に検出されたエラーまたは警告の合計数を戻します。

構文

```
public int getNumMessages()
```

戻り値

エラーまたは警告の数

getPublicId(int)

説明

指定されたインデックスにおけるエラーが発生したときの入力の公開識別子を取得します。

構文

```
public java.lang.String getPublicId(int i)
```

戻り値

公開識別子

getSystemId(int)

説明

指定されたインデックスにおけるエラーが発生したときの入力のシステム識別子を取得します。

構文

```
public java.lang.String getSystemId(int i)
```

戻り値

システム識別子

printErrorListener()

説明

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

構文

```
public void printErrorListener()
```

reset()

説明

エラー・クラスをリセットします。

構文

```
public void reset()
```

setErrorStream(OutputStream)

説明

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

out ― エラー / 警告を出力する出力ストリーム

setErrorStream(OutputStream, String)

説明

出力ストリームを登録します。

構文

```
public void setErrorStream(java.io.OutputStream out, java.lang.String enc)
```

パラメータ

out ― エラー / 警告を出力する出力ストリーム

enc ― 出力ストリームのエンコーディング

例外
IOException ― 出力ストリームの初期化でエラーが発生した場合

setErrorStream(PrintWriter)

説明
出力ストリームを登録します。

構文
```
public void setErrorStream(java.io.PrintWriter out)
```

パラメータ
out ― エラー / 警告を出力する PrintWriter

setException(Exception)

説明
例外を登録します。

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

パラメータ
exp ― 最後に発生した例外

setLocale(Locale)

説明
ロケールを登録します。

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

パラメータ
locale ― エラー・レポート用のロケール

setLocator(Locator)

説明

ロケータを登録します。

構文

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

パラメータ

`locator` ― lin/col/sysid/pubid 情報を取得するロケータ

showWarnings(boolean)

説明

レポートの警告のオン / オフを切り替えます。

構文

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

パラメータ

`flag` ― 警告のレポートを制御するフラグ

XMLException

説明

パッケージ oracle.xml.util で、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

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

実装されるインタフェース

```
java.io.Serializable
```

フィールド

表 10-4　AttrDecl のフィールド

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

コンストラクタ

XMLException(String, String, String, int, int, int)

構文

```
public  XMLException(java.lang.String mesg, java.lang.String pubId,
java.lang.String sysId, int line, int col, int type)
```

XMLException(XMLError, Exception)

構文

```
public  XMLException(XMLError err, java.lang.Exception e)
```

XMLException(XMLError, int)

構文

```
public  XMLException(XMLError err, int firsterr)
```

XMLException(XMLError, int, Exception)

構文

```
public  XMLException(XMLError err, int firsterr, java.lang.Exception e)
```

メソッド

表 10-5 oracle.xml.util.XMLException のメソッドの概要

メソッド	説明
formatErrorMessage(int)	指定されたインデックスにおけるエラー・メッセージを取得します。
getColumnNumber(int)	指定されたインデックスにおけるエラーの列番号を取得します。
getException(int)	指定されたインデックスにおけるエラーで発生した例外（存在する場合）を取得します。
getLineNumber(int)	指定されたインデックスにおけるエラーの行番号を取得します。
getMessage(int)	指定されたインデックスにおけるエラー・メッセージを取得します。
getMessageType(int)	指定されたインデックスにおけるエラー・メッセージのタイプを取得します。
getNumMessages(int)	解析中に検出されたエラーまたは警告の合計数を戻します。
getPublicId(int)	指定されたインデックスにおけるエラーが発生したときの入力の公開識別子を取得します。
getSystemId(int)	指定されたインデックスにおけるエラーが発生したときの入力のシステム識別子を取得します。
getXMLError	XMLException 内の XMLError オブジェクトを取得します。
printStackTrace()	この Throwable とそのバックトレースを標準のエラー・ストリームに出力します。
printStackTrace(PrintStream)	この Throwable と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。

表 10-5 oracle.xml.util.XMLException のメソッドの概要（続き）

メソッド	説明
printStackTrace(PrintWriter)	この Throwable とそのバックトレースを指定した出力ライターに出力します。
setException(Exception)	実際の例外（存在する場合）を設定します。
toString()	aReturnsny 埋込み例外をピックアップする toString をオーバーライドします。

formatErrorMessage(int)

説明

指定されたインデックスにおけるエラー・メッセージを取得します。

構文

```
public java.lang.String formatErrorMessage(int i)
```

戻り値

エラー・メッセージ

getColumnNumber(int)

説明

指定されたインデックスにおけるエラーの列番号を取得します。

構文

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

戻り値

列番号

getException(int)

説明

指定されたインデックスにおけるエラーで発生した例外（存在する場合）を取得します。

構文

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

戻り値
例外

getLineNumber(int)

説明
指定されたインデックスにおけるエラーの行番号を取得します。

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

戻り値
行番号

getMessage(int)
指定されたインデックスにおけるエラー・メッセージを取得します。

説明

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

戻り値
エラー・メッセージ

getMessageType(int)

説明
指定されたインデックスにおけるエラー・メッセージのタイプを取得します。

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

戻り値
エラー・メッセージのタイプ

getNumMessages()

解析中に検出されたエラーまたは警告の合計数を戻します。

説明

構文

```
public int getNumMessages()
```

戻り値

エラーまたは警告の数

getPublicId(int)

説明

指定されたインデックスにおけるエラーが発生したときの入力の公開識別子を取得します。

構文

```
public java.lang.String getPublicId(int i)
```

戻り値

公開識別子

getSystemId(int)

説明

指定されたインデックスにおけるエラーが発生したときの入力のシステム識別子を取得します。

構文

```
public java.lang.String getSystemId(int i)
```

戻り値

システム識別子

getXMLError()

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

構文
```
public XMLError getXMLError()
```

戻り値
XMLError オブジェクト

printStackTrace()

説明
この Throwable とそのバックトレースを標準のエラー・ストリームに出力します。

構文
```
public void printStackTrace()
```

オーバーライド
クラス java.lang.Throwable 内の java.lang.Throwable.printStackTrace()

printStackTrace(PrintStream)

説明
この Throwable と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。

構文
```
public void printStackTrace(java.io.PrintStream s)
```

オーバーライド
クラス java.lang.Throwable 内の
java.lang.Throwable.printStackTrace(java.io.PrintStream)

printStackTrace(PrintWriter)

説明

この Throwable とそのバックトレースを指定した出力ライターに出力します。

構文

```
public void printStackTrace(java.io.PrintWriter s)
```

オーバーライド

クラス java.lang.Throwable 内の
java.lang.Throwable.printStackTrace(java.io.PrintWriter)

setException(Exception)

説明

実際の例外（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

ex ― 例外

toString()

説明

aReturnsny 埋込み例外をピックアップする toString をオーバーライドします。

```
public java.lang.String toString()
```

オーバーライド

クラス java.lang.Throwable 内の java.lang.Throwable.toString()

戻り値

この例外の文字列表現

11

パッケージ oracle.xml.parser.v2

この章では、パッケージ oracle.xml.parser.v2 に含まれているクラスについて説明します。このクラスを使用して、Oracle XDK for Java に XML Parser、DOM および SAX の API を実装します （DOM はドキュメント・オブジェクト・モデル、SAX は Simple API for XML の略称です）。oracle.xml.parser.v2 パッケージには、XML Parser、DOM および SAX の API の全機能が含まれています。

この章は、次の項で構成されています。

注意： サポートされているユーティリティ・クラスは、パッケージ oracle.xml.util に含まれています。詳細は、第 10 章「パッケージ oracle.xml.util」で説明されています。

パッケージ oracle.xml.parser.v2 の説明

パッケージ oracle.xml.parser.v2 に含まれるクラスは、XML Parser、DOM および SAX の API を Oracle9*i* XDK for Java に実装します。oracle.xml.parser.v2 パッケージには、XSLT Processor for Java を実装するクラスも含まれています。

DOM と SAX の API の Oracle 実装は、World Wide Web Consortium（W3C）の XML 標準に関する勧告に従っています。

サポートされているユーティリティ・クラスは、パッケージ oracle.xml.util に含まれています。詳細は、第 10 章「パッケージ oracle.xml.util」で説明されています。

関連項目：

- 第 10 章「パッケージ oracle.xml.util」
- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』

パッケージ oracle.xml.parser.v2 の概要

この項の表では、この章で説明する oracle.xml.parser.v2 インタフェースとクラスの概要を説明します。

表 11-1　oracle.xml.parser.v2 パッケージのインタフェース

インタフェース	説明
NSResolver インタフェース	名前空間の解決に関するサポートを提供します。
PrintDriver インタフェース	`PrintDriver` インタフェースは、DOM ツリーとして表される XML 文書の出力で使用するメソッドを定義します。
XMLToken インタフェース	XMLToken の基本インタフェースです。

表 11-2　oracle.xml.parser.v2 パッケージのクラス

クラス	説明
AttrDecl	Document Type Definition（DTD）内の属性リストに宣言されている各属性に関する情報を保持します。
DefaultXMLDocumentHandler	`XMLDocumentHandler` インタフェースのデフォルトの動作を実装します。
DOMParser	World Wide Web Consortium（W3C）の勧告に従って、XML 1.0 パーサーを実装します。
DTD	DOM の DocumentType インタフェースを実装し、XML 文書の Document Type Definition（DTD）情報を保持します。
ElementDecl	Document Type Definition（DTD）内の要素宣言を表します。
NodeFactory	解析時に作成される 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。
oraxml	XML 宣言の処理命令を実装します。
SAXAttrList	SAX の `AttributeList` インタフェースを実装します。また、名前空間サポートも提供します。
SAXParser	World Wide Web Consortium（W3C）の勧告に従って、XML 1.0 SAX パーサーを実装します。
XMLAttr	DOM の Attr インタフェースを実装し、要素の各属性に関する情報を保持します。

表 11-2 oracle.xml.parser.v2 パッケージのクラス（続き）

クラス	説明
XMLCDATA	DOM の CDATASection インタフェースを実装します。
XMLComment	DOM の Comment インタフェースを実装します。
XMLDeclPI	XML 宣言の処理命令を実装します。
XMLDocument	このクラスは、DOM の Document インタフェースを実装し、XML 文書全体を表し、ドキュメント・オブジェクト・モデル・ツリーのルートとして機能します。
XMLDocumentFragment	このクラスは、DOM の DocumentFragment インタフェースを実装します。
XMLDOMImplementation	DOMImplementation を実装します。
XMLElement	このクラスは、DOM の `Element` インタフェースを実装します。
XMLEntityReference	DOM の `EntityReference` インタフェースを実装します。
XMLNode	DOM の `Node` インタフェースを実装し、ドキュメント・オブジェクト・モデル全体のプライマリ・データ型として機能します。
XMLParser	このクラスは、`DOMParser` クラスと `SAXParser` クラスのベース・クラスとして機能します。
XMLPI	このクラスは、DOM の Processing Instruction インタフェースを実装します。
XMLText	このクラスは、DOM の Text インタフェースを実装します。
XMLTokenizer	このクラスは、World Wide Web Consortium（W3C）の勧告に従って、XML 1.0 パーサーを実装します。

表 11-3 oracle.xml.parser.v2 パッケージの例外

例外	説明
XMLDOMException	DOM 操作での例外を示します。
XMLParseException	XML 文書の処理中に解析例外が発生したことを示します。
XMLRangeException	DOM 範囲操作での例外を示します。

表 11-4「パッケージ oracle.xml.parserv2 の XSLT Processor クラスの概要」では、oracle.xml.parser.v2 パッケージに含まれる XSLT Processor クラスの概要を説明します。

表 11-4 パッケージ oracle.xml.parserv2 の XSLT Processor クラスの概要

クラス	説明
oraxsl クラス	複数の XML 文書にスタイルシートを適用するコマンドライン・インタフェースを提供します。
XPathException クラス	XSL 変換時に例外が発生したことを示します。
XSLProcessor クラス	事前に構成された `XSLStylesheet` を使用して、入力 XML 文書を変換するメソッドを提供します。
XSLStylesheet クラス	XSL スタイルシートの情報（テンプレート、キー、変数および属性セットなど）を保持します。

NSResolver インタフェース

NSResolver の構文

```
public interface NSResolver
```

NSResolver の説明

このインタフェースは、名前空間の解決に関するサポートを提供します。

NSResolver の既知の実装クラス

```
XMLElement
```

メソッド

resolveNamespacePrefix(String)

説明

指定された名前空間接頭辞に対して、有効範囲内の名前空間定義を検索します。

構文

```
public java.lang.String resolveNamespacePrefix(java.lang.String prefix)
```

パラメータ

`prefix` ― 解決する名前空間接頭辞

戻り値

解決済み名前空間（接頭辞を解決できなかった場合は null）

PrintDriver インタフェース

PrintDriver の説明

PrintDriver インタフェースは、DOM ツリーとして表される XML 文書の出力で使用するメソッドを定義します。

PrintDriver の構文

```
public interface PrintDriver
```

PrintDriver の実装クラス

XMLPrintDriver

メソッド

表 11-5 PrintDriver のメソッドの概要

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

表 11-5 PrintDriver のメソッドの概要（続き）

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

close()

説明

出力ストリームまたは出力ライターをクローズします。

構文

```
public void close()
```

flush()

説明

出力ストリームまたは出力ライターをフラッシュします。

構文

```
public void flush()
```

printAttribute(XMLAttr)

説明

XMLAttr ノードを出力します。

構文

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

パラメータ

attr − XMLAttr ノード

printAttributeNodes(XMLElement)

説明

XMLElement の属性ごとに出力メソッドをコールします。

構文

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

パラメータ

elem − 属性を出力する要素

printCDATASection(XMLCDATA)

説明

XMLCDATA ノードを出力します。

構文

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

パラメータ

cdata − XMLCDATA ノード

printChildNodes(XMLNode)

説明

XMLNode の子ごとに出力メソッドをコールします。

構文

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

パラメータ

node − 子を出力するノード

printComment(XMLComment)

説明

XMLComment ノードを出力します。

構文

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

パラメータ

comment ― コメント・ノード

printDoctype(DTD)

説明

DTD を出力します。

構文

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

パラメータ

dtd ― 出力する DTD

printDocument(XMLDocument)

説明

XMLDocument を出力します。

構文

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

パラメータ

doc ― 出力するドキュメント

printDocumentFragment(XMLDocumentFragment)

構文

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

説明

空の XMLDocumentFragment オブジェクトを出力します。

パラメータ

dfrag − 出力するドキュメント・フラグメント

printElement(XMLElement)

構文

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

説明

XMLElement を出力します。

パラメータ

elem − 出力する要素

printEntityReference(XMLEntityReference)

説明

XMLEntityReference ノードを出力します。

構文

```
public void printEntityReference(XMLEntityReference en)
```

パラメータ

en − XMLEntityReference ノード

printProcessingInstruction(XMLPI)

説明

XMLPI ノードを出力します。

構文

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

パラメータ

pi − XMLPI ノード

printTextNode(XMLText)

説明

XMLText ノードを出力します。

構文

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

パラメータ

text − テキスト・ノード

setEncoding(String)

説明

出力ドライバのエンコーディングを設定します。

構文

```
public void setEncoding(java.lang.String enc)
```

パラメータ

enc − 出力するドキュメントのエンコーディング

NSName

パッケージ oracle.xml.util に含まれているインタフェースです。

説明
パッケージ oracle.xml.util の一部です。このインタフェースは、Element 名と Attr 名に対する名前空間サポートを提供します。

構文
```
public interface NSName
```

メソッド

getExpandedName()

説明
この名前の解決済み完全名を取得します。

構文
```
public java.lang.String getExpandedName()
```

戻り値
解決済み完全名

getLocalName()

説明
この名前のローカル名を取得します。

構文
```
public java.lang.String getLocalName()
```

戻り値
ローカル名

getNamespace()

説明
この名前の解決済み名前空間を取得します。

構文
```
public java.lang.String getNamespace()
```

戻り値
解決済み名前空間

getPrefix()

説明
この名前の接頭辞を取得します。

構文
```
public java.lang.String getPrefix()
```

戻り値
接頭辞

getQualifiedName()

説明
修飾名を取得します。

構文
```
public java.lang.String getQualifiedName()
```

戻り値
修飾名

AttrDecl

説明

このクラスは、Document Type Definition（DTD）内の属性リストに宣言されている各属性に関する情報を保持します。

構文

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

oracle.xml.parser.v2.AttrDecl

実装されるインタフェース

java.io.Externalizable, java.io.Serializable

フィールド

表 11-6 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－ StringType － CDATA
DEFAULT	public static final int DEFAULT	属性が存在するかどうかを表す型 － デフォルト
ENTITIES	public static final int ENTITIES	属性の型－ TokenizedType － エンティティ（複数）
ENTITY	public static final int ENTITY	属性の型－ TokenizedType － エンティティ
ENUMERATION	public static final int ENUMERATION	属性の型－ EnumeratedType － 一覧
FIXED	public static final int FIXED	属性が存在するかどうかを表す型 － 固定
ID	public static final int ID	属性の型－ TokenizedType － ID
IDREF	public static final int IDREF	属性の型－ TokenizedType － ID 参照
IDREFS	public static final int IDREFS	属性の型－ TokenizedType － ID 参照（複数）
IMPLIED	public static final int IMPLIED	属性が存在するかどうかを表す型 － 暗黙
NMTOKEN	public static final int NMTOKEN	属性の型－ TokenizedType － 名前トークン

表 11-6 AttrDecl のフィールド（続き）

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

コンストラクタ

AttrDecl()

説明

デフォルトのコンストラクタです。このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。

構文

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

メソッド

表 11-7 AttrDecl のメソッドの概要

メソッド	説明
getAttrPresence()	属性が存在するかどうかを表す型を取得します。
getAttrType()	属性の型を取得します。
getDefaultValue	属性のデフォルト値を取得します。
getEnumerationValues()	属性の値を取得します。
getNodeName()	AttrDecl の名前を取得します。
getNodeType()	実際のオブジェクトのタイプを示すコードを取得します。
readExternal(ObjectInput)	writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

表 11-7 AttrDecl のメソッドの概要（続き）

メソッド	説明
typeToString (int)	属性の型を表す文字列を取得します。
writeExternal(ObjectOutput)	このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

getAttrPresence()

説明

属性が存在するかどうかを表す型を取得します。

構文

```
public int getAttrPresence()
```

戻り値

属性が存在するかどうかを表す型

getAttrType()

説明

属性の型を取得します。

構文

```
public int getAttrType()
```

戻り値

属性の型

getDefaultValue()

説明
属性のデフォルト値を取得します。

構文
```
public java.lang.String getDefaultValue()
```

戻り値
属性のデフォルト値

getEnumerationValues()

説明
属性の値を取得します。

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

戻り値
属性の値の一覧

getNodeName()

説明
AttrDecl の名前を取得します。

構文
```
public java.lang.String getNodeName()
```

戻り値
ノードの名前

getNodeType()

説明

実際のオブジェクトのタイプを示すコードを取得します。

構文

```
public short getNodeType()
```

戻り値

ノードのタイプ

readExternal(ObjectInput)

説明

このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

構文

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

指定方法

インタフェース `java.io.Externalizable` 内の
`java.io.Externalizable.readExternal(java.io.ObjectInput)`

パラメータ

`inArg` ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外

`IOException` ― 入力ストリームの読込み時にエラーがあった場合に発生します。

`ClassNotFoundException` ― クラスが検出されない場合に発生します。

typeToString(int)

説明
属性の型を表す文字列を取得します。

構文
```
public static java.lang.String typeToString(int type)
```

戻り値
属性の型を表す文字列

writeExternal(ObjectOutput)

説明
このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文
```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法
インタフェース `java.io.Externalizable` 内の `java.io.Externalizable.writeExternal(java.io.ObjectOutput)`

パラメータ
`outArg` ― シリアル化または圧縮されたストリームの書込みに使用する ObjectOutput ストリーム

例外
`IOException` ― シリアル化または圧縮されたストリームの書込み時に例外があった場合に発生します。

DefaultXMLDocumentHandler

DefaultXMLDocumentHandler の説明

このクラスは、XMLDocumentHandler インタフェースのデフォルトの動作を実装します。

インタフェースの一部のみ実装する必要がある場合、アプリケーション開発者は、このクラスを拡張できます。

DefaultXMLDocumentHandler の構文

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

oracle.xml.parser.v2.DefaultXMLDocumentHandler

DefaultXMLDocumentHandler により実装されるインタフェース

XMLDocumentHandler

コンストラクタ

DefaultXMLDocumentHandler()

説明

デフォルトのドキュメントを構成します。

構文

```
public  DefaultXMLDocumentHandler()
```

メソッド

表 11-8 DefaultXMLDocumentHandler のメソッドの概要

メソッド	説明
cDATASection(char[], int, int)	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。
comment(String)	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、メイン・ドキュメント要素の前後に発生する可能性があることに注意してください。

表 11-8 DefaultXMLDocumentHandler のメソッドの概要（続き）

メソッド	説明
endDoctype()	DTD 終了の通知を受け取ります。
endElement(NSName)	要素終了の通知を受け取ります。
endElement(String, String, String)	要素終了の通知を受け取ります。
endPrefixMapping(String)	URI の接頭辞マッピングの有効範囲を終了します。
getHandler()	次のパイプライン・ノード・ハンドラを取得します。
setDoctype(DTD)	DTD の通知を受け取ります。DTD を設定します。
setError(XMLError)	XMLError ハンドラの通知を受け取ります。
setHandler(XMLDocumentHandler)	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(String, String)	テキスト XML 宣言の通知を受け取ります。パーサーは、テキスト XML 宣言ごとにこのメソッドをコールします。
setXMLDecl(String, String, String)	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。
setXMLSchema(Object)	XMLSchema オブジェクトの通知を受け取ります。
skippedEntity(String)	スキップされたエンティティの通知を受け取ります。
startElement(NSName, SAXAttrList)	要素開始の通知を受け取ります。
startElement(String, String, String, Attributes)	要素開始の通知を受け取ります。
startPrefixMapping(String, String)	URI 名前空間の接頭辞マッピングの有効範囲を開始します。

cDATASection(char[], int, int)

説明

CDATA セクションの通知を受け取ります。パーサーは、CDATA セクションが検出されるたびにこのメソッドをコールします。

構文

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

指定方法

インタフェース XMLDocumentHandler 内の XMLDocumentHandler.cDATASection(char[], int, int)

パラメータ

ch − CDATA セクションの文字

start − 文字配列の開始位置

length − 文字配列から使用する文字数

例外

org.xml.sax.SAXException − SAX 例外で、別の例外をラップしている場合もあります。

comment(String)

説明

コメントの通知を受け取ります。パーサーは、コメントが検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、メイン・ドキュメント要素の前後に発生する可能性があることに注意してください。

構文

```
public void comment(java.lang.String data)
```

指定方法

インタフェース XMLDocumentHandler 内の XMLDocumentHandler.comment(String)

パラメータ

data − コメント・データ、指定されない場合は null

例外
org.xml.sax.SAXException － SAX 例外で、別の例外をラップしている場合もあります。

endDoctype()

説明
DTD 終了の通知を受け取ります。

構文
```
public void endDoctype()
```

指定方法
インタフェース XMLDocumentHandler 内の XMLDocumentHandler.endDoctype()

例外
org.xml.sax.SAXException － SAX 例外で、別の例外をラップしている場合もあります。

endElement(NSName)

説明
要素終了の通知を受け取ります。

構文
```
public void endElement(NSName elem)
```

指定方法
インタフェース XMLDocumentHandler 内の
XMLDocumentHandler.endElement(NSName)

パラメータ
elem － NSName オブジェクト

例外
SAXException － SAXException が発生する場合があります。

関連項目
org.xml.sax.DocumentHandler.endElement

endElement(String, String, String)

説明

要素終了の通知を受け取ります。

構文

```
public void endElement(java.lang.String namespaceURI, java.lang.String localName,
java.lang.String qName)
```

パラメータ

namespaceURI ― 名前空間 URI（要素に名前空間 URI がない場合、または名前空間処理が実行されていない場合は空の文字列）

localName ― ローカル名（接頭辞なし）、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

qName ― XML 1.0 の修飾名（接頭辞付き）、または修飾名が使用できない場合は空の文字列

例外

org.xml.sax.SAXException ― SAX 例外で、別の例外をラップしている場合もあります。

endPrefixMapping(String)

説明

URI の接頭辞マッピングの有効範囲を終了します。

構文

```
public void endPrefixMapping(java.lang.String prefix)
```

パラメータ

prefix ― マッピングされた接頭辞

例外

org.xml.sax.SAXException ― クライアントは処理中に例外をスローできます。

関連項目

startPrefixMapping(String, String)、endElement(NSName)

getHandler()

説明
次のパイプライン・ノード・ハンドラを取得します。

構文
```
public XMLDocumentHandler getHandler()
```

指定方法
インタフェース XMLDocumentHandler 内の XMLDocumentHandler.getHandler()

戻り値
XMLDocumentHandler ノード

setDoctype(DTD)

説明
DTD の通知を受け取ります。DTD を設定します。

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

指定方法
インタフェース XMLDocumentHandler 内の XMLDocumentHandler.setDoctype(DTD)

例外
org.xml.sax.SAXException − SAX 例外で、別の例外をラップしている場合もあります。

setError(XMLError)

説明
XMLError ハンドラの通知を受け取ります。

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

指定方法
インタフェース XMLDocumentHandler 内の
XMLDocumentHandler.setError(XMLError)

パラメータ
err − XMLError オブジェクト

例外
org.xml.sax.SAXException − SAX 例外で、別の例外をラップしている場合もあります。

setHandler(XMLDocumentHandler)

説明
次のパイプライン・ノード・ハンドラの通知を受け取ります。

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

指定方法
インタフェース XMLDocumentHandler 内の
XMLDocumentHandler.setHandler(XMLDocumentHandler)

パラメータ
h − XMLDocumentHandler ノード

例外
org.xml.sax.SAXException − SAX 例外で、別の例外をラップしている場合もあります。

setTextDecl(String, String)

説明
テキスト XML 宣言の通知を受け取ります。パーサーは、テキスト XML 宣言ごとにこのメソッドをコールします。

構文
```
public void setTextDecl(java.lang.String version, java.lang.String encoding)
```

指定方法
インタフェース XMLDocumentHandler 内の
XMLDocumentHandler.setTextDecl(String, String)

パラメータ

version ― バージョン番号（指定されない場合は null）

encoding ― エンコーディング名

例外

org.xml.sax.SAXException ― SAX 例外で、別の例外をラップしている場合もあります。

setXMLDecl(String, String, String)

説明

XML 宣言の通知を受け取ります。 パーサーは、XML 宣言ごとにこのメソッドをコールします。

構文

```
public void setXMLDecl(java.lang.String version, java.lang.String standalone,
java.lang.String encoding)
```

指定方法

インタフェース XMLDocumentHandler 内の
XMLDocumentHandler.setXMLDecl(String, String, String)

パラメータ

version ― バージョン番号

standalone ― スタンドアロン値（指定されない場合は null）

encoding ― エンコーディング名（指定されない場合は null）

例外

org.xml.sax.SAXException ― SAX 例外で、別の例外をラップしている場合もあります。

setXMLSchema(Object)

説明

XMLSchema オブジェクトの通知を受け取ります。

構文

```
public void setXMLSchema(java.lang.Object s)
```

指定方法
インタフェース XMLDocumentHandler 内の
XMLDocumentHandler.setXMLSchema(Object)

パラメータ
s − XMLSchema オブジェクト

例外
org.xml.sax.SAXException − SAX 例外で、別の例外をラップしている場合もあります。

skippedEntity(String)

説明
スキップされたエンティティの通知を受け取ります。

構文
```
public void skippedEntity(java.lang.String name)
```

パラメータ
name − スキップされたエンティティの名前。名前は、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文字列 [dtd] になります。

例外
org.xml.sax.SAXException − SAX 例外で、別の例外をラップしている場合もあります。

startElement(NSName, SAXAttrList)

説明
要素開始の通知を受け取ります。

構文
```
public void startElement(NSName elem, SAXAttrList attrlist)
```

指定方法
インタフェース XMLDocumentHandler 内の
XMLDocumentHandler.startElement(NSName, SAXAttrList)

パラメータ

elem ― NSName オブジェクト

attrlist ― 要素の SAXAttrList

例外

SAXException ― SAXException が発生する場合があります。

関連項目

org.xml.sax.DocumentHandler.startElement

startElement(String, String, String, Attributes)

説明

要素開始の通知を受け取ります。

構文

```
public void startElement(java.lang.String namespaceURI, java.lang.String localName,
java.lang.String qName, org.xml.sax.Attributes atts)
```

パラメータ

namespaceURI ― 名前空間 URI（要素に名前空間 URI がない場合、または名前空間処理が実行されていない場合は空の文字列）

localName ― ローカル名（接頭辞なし）、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

qName ― 修飾名（接頭辞付き）、または修飾名が使用できない場合は空の文字列

atts ― 要素に連結されている属性、または属性がない場合は空の Attributes オブジェクト

例外

org.xml.sax.SAXException ― SAX 例外で、別の例外をラップしている場合もあります。

関連項目

endElement(NSName)、org.xml.sax.Attributes

startPrefixMapping(String, String)

説明

URI 名前空間の接頭辞マッピングの有効範囲を開始します。

構文

```
public void startPrefixMapping(java.lang.String prefix, java.lang.String uri)
```

パラメータ

`prefix` ― 宣言される名前空間の接頭辞

`uri` ― 接頭辞のマップ先の名前空間 URI

例外

`org.xml.sax.SAXException` ― クライアントは処理中に例外をスローできます。

関連項目

`endPrefixMapping(String)`、`startElement(NSName, SAXAttrList)`

DocumentBuilder

DocumentBuilder の構文

```
public class DocumentBuilder
```

oracle.xml.parser.v2.DocumentBuilder

DocumentBuilder の説明

このクラスは、SAX 2.0 イベントから DOM ツリーを作成するための XMLDocumentHandler（使用できません）と ContentHandler を実装します。XMLDocumentHandler イベントは、下位互換性を維持するためにサポートされています。

コンストラクタ

DocumentBuilder()

説明

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

構文

```
public  DocumentBuilder()
```

メソッド

表 11-9 DocumentBuilder のメソッドの概要

メソッド	説明
attributeDecl(String, String, String, String, String)	属性の型の宣言をレポートします。
cDATASection(char[], int, int)	要素内の CDATA セクション・データの通知を受け取ります。
characters(char[], int, int)	要素内の文字データの通知を受け取ります。
comment(char[], int, int)	ドキュメント内にある XML コメントをレポートします。
comment(String)	コメントの通知を受け取ります。
elementDecl(String, String)	要素の型の宣言をレポートします。
endCDATA()	CDATA セクション終了をレポートします。

表 11-9 DocumentBuilder のメソッドの概要（続き）

メソッド	説明
endDoctype()	DTD 終了の通知を受け取ります。
endDocument()	ドキュメント終了の通知を受け取ります。
endDTD()	DTD 宣言終了をレポートします。
endElement(NSName)	要素終了の通知を受け取ります。
endElement(Sring, String, String)	要素終了の通知を受け取ります。
endEntity(String)	エンティティ終了をレポートします。
externalEntityDecl(String, String, String)	解析済み外部エンティティ宣言をレポートします。
getCurrentNode()	作成する現行ノードを取得します。
getDocument()	作成するドキュメントを取得します。
ignorableWhitespace(char[], int, int)	要素の内容にある無視できる空白の通知を受け取ります。
internalEntityDecl(String, String)	内部エンティティ宣言をレポートします。
processingInstruction(String, String)	処理命令の通知を受け取ります。
retainCDATASection(boolean)	フラグを設定して CDATA セクションを保持します。
setDebugMode(boolean)	フラグを設定してドキュメントのデバッグ情報をオンにします。
setDoctype(DTD)	DTD の通知を受け取ります。DTD を設定します。
setDocumentLocator(Locator)	ドキュメント・イベントに関するロケータ・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・ライターは、他のドキュメント・イベントで使用するロケータを格納する場合、サブクラスにあるこのメソッドはオーバーライドできます。
setNodeFactory(NodeFactory)	カスタム DOM ツリーの作成に使用する、オプションの NodeFactory を設定します。
setTextDecl(String, String)	テキスト XML 宣言の通知を受け取ります。パーサーは、テキスト XML 宣言ごとにこのメソッドをコールします。
setXMLDecl(String, String, String)	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()	CDATA セクション開始をレポートします。
startDocument()	ドキュメント開始の通知を受け取ります。
startDTD(String, String, String)	DTD 宣言（存在する場合）開始をレポートします。

表 11-9 DocumentBuilder のメソッドの概要（続き）

メソッド	説明
startElement(NSName, SAXAttrList)	要素開始の通知を受け取ります。
startElement(String, String, String, Attributes)	要素開始の通知を受け取ります。
startEntity(String)	内部と外部の XML エンティティ開始をレポートします。startEntity と endEntity のすべてのイベントが正しくネストされている必要があります。

attributeDecl(String, String, String, String, String)

説明

属性の型の宣言をレポートします。

構文

```
public void attributeDecl(java.lang.String eName, java.lang.String aName,
java.lang.String type, java.lang.String valueDefault, java.lang.String value)
```

パラメータ

eName − 対応する要素の名前

aName − 属性の名前

type − 属性の型を表す文字列

valueDefault − 属性のデフォルトを表す文字列（"#IMPLIED"、"#REQUIRED" または "#FIXED"）、これらの文字列が適用されない場合は null

value − 属性のデフォルト値を表す文字列、またはデフォルト値がない場合は null

例外

SAXException − アプリケーションで例外が発生する場合があります。

cDATASection(char[ ], int, int)

説明

要素内の CDATA セクション・データの通知を受け取ります。

構文

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

パラメータ

ch − CDATA セクションの文字列を保持する配列

start − 文字配列の開始位置

length − 文字配列から使用する文字数

例外

org.xml.sax.SAXException − SAX 例外で、別の例外をラップしている場合もあります。

関連項目

org.xml.sax.DocumentHandler.characters

characters(char[], int, int)

説明

要素内の文字データの通知を受け取ります。

構文

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

パラメータ

ch − CDATA セクションの文字列を保持する配列

start − 文字配列の開始位置

length − 文字配列から使用する文字数

例外

org.xml.sax.SAXException − SAX 例外で、別の例外をラップしている場合もあります。

関連項目

org.xml.sax.DocumentHandler.characters

comment(char[], int, int)

説明
ドキュメント内にある XML コメントをレポートします。

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

パラメータ
ch ― コメント内の文字を保持する配列

start ― 配列の開始位置

length ― 配列から使用する文字数

例外
SAXException ― アプリケーションで例外が発生する場合があります。

comment(String)

説明
コメントの通知を受け取ります。

構文
```
public void comment(java.lang.String data)
```

パラメータ
data ― コメント・データ、指定されない場合は null

例外
org.xml.sax.SAXException ― SAX 例外で、別の例外をラップしている場合もあります。

elementDecl(String, String)

説明
要素の型の宣言をレポートします。

構文
```
public void elementDecl(java.lang.String name, java.lang.String model)
```

パラメータ

name ― 要素型名

model ― 正規化された文字列としてのコンテンツ・モデル

例外

org.xml.sax.SAXException ― アプリケーションで例外が発生する場合があります。

endCDATA()

説明

CDATA セクション終了をレポートします。

構文

```
public void endCDATA()
```

例外

org.xml.sax.SAXException ― アプリケーションで例外が発生する場合があります。

関連項目

startCDATA()

endDoctype()

説明

DTD 終了の通知を受け取ります。

構文

```
public void endDoctype()
```

例外

org.xml.sax.SAXException ― SAX 例外で、別の例外をラップしている場合もあります。

endDocument()

説明
ドキュメント終了の通知を受け取ります。

構文
```
public void endDocument()
```

例外
`org.xml.sax.SAXException` − SAX 例外で、別の例外をラップしている場合もあります。

関連項目
`org.xml.sax.DocumentHandler.endDocument`

endDTD()

説明
DTD 宣言終了をレポートします。

構文
```
public void endDTD()
```

例外
`org.xml.sax.SAXException` − アプリケーションで例外が発生する場合があります。

関連項目
`startDTD(String, String, String)`

endElement(NSName)

説明
要素終了の通知を受け取ります。

構文
```
public void endElement(NSName elem)
```

パラメータ
`elem` − NSName オブジェクト

例外

org.xml.sax.SAXException － SAXException が発生する場合があります。

関連項目

org.xml.sax.DocumentHandler.endElement

endElement(String, String, String)

説明

要素終了の通知を受け取ります。

構文

```
public void endElement(java.lang.String namespaceURI, java.lang.String localName,
java.lang.String qName)
```

パラメータ

namespaceURI － 名前空間 URI（要素に名前空間 URI がない場合、または名前空間処理が実行されていない場合は空の文字列）

localName － ローカル名（接頭辞なし）、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

qName － XML 1.0 の修飾名（接頭辞付き）、または修飾名が使用できない場合は空の文字列

例外

org.xml.sax.SAXException － SAX 例外で、別の例外をラップしている場合もあります。

endEntity(String)

説明

エンティティ終了をレポートします。

構文

```
public void endEntity(java.lang.String name)
```

パラメータ

name － 終了するエンティティの名前

例外
org.xml.sax.SAXException ー アプリケーションで例外が発生する場合があります。

関連項目
startEntity(String)

externalEntityDecl(String, String, String)

説明
解析済み外部エンティティ宣言をレポートします。

構文
```
public void externalEntityDecl(java.lang.String name, java.lang.String publicId,
java.lang.String systemId)
```

パラメータ
name ー エンティティの名前。パラメータ・エンティティの場合、名前は % で始まります。

publicId ー エンティティの宣言済み公開識別子、宣言済み公開識別子がない場合は null。

systemId ー エンティティの宣言済みシステム識別子。

例外
org.xml.sax.SAXException ー アプリケーションで例外が発生する場合があります。

関連項目
internalEntityDecl(String, String)、
org.xml.sax.DTDHandler.unparsedEntityDecl

getCurrentNode()

説明
作成する現行ノードを取得します。

構文
```
public XMLNode getCurrentNode()
```

戻り値
XMLNode

getDocument()

説明
作成するドキュメントを取得します。

構文
```
public XMLDocument getDocument()
```

戻り値
XMLDocument

ignorableWhitespace(char[], int, int)

説明
要素の内容にある無視できる空白の通知を受け取ります。

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

パラメータ
ch ― 空白文字

start ― 文字配列の開始位置

length ― 文字配列から使用する文字数

例外
org.xml.sax.SAXException ― SAX 例外で、別の例外をラップしている場合もあります。

関連項目
org.xml.sax.DocumentHandler.ignorableWhitespace

internalEntityDecl(String, String)

説明
内部エンティティ宣言をレポートします。

構文
```
public void internalEntityDecl(java.lang.String name, java.lang.String value)
```

パラメータ

name ― エンティティの名前。パラメータ・エンティティの場合、名前は % で始まります。

value ― エンティティの置換テキスト。

例外

org.xml.sax.SAXException ― アプリケーションで例外が発生する場合があります。

関連項目

externalEntityDecl(String, String, String)、
org.xml.sax.DTDHandler.unparsedEntityDecl

processingInstruction(String, String)

説明

処理命令の通知を受け取ります。

構文

```
public void processingInstruction(java.lang.String target, java.lang.String data)
```

パラメータ

target ― 処理命令のターゲット

data ― 処理命令データ、指定されない場合は null

例外

org.xml.sax.SAXException ― SAX 例外で、別の例外をラップしている場合もあります。

関連項目

org.xml.sax.DocumentHandler.processingInstruction

retainCDATASection(boolean)

説明
フラグを設定して CDATA セクションを保持します。

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

パラメータ
`flag` − CDATA セクションを保持するかどうかを決定します。

setDebugMode(boolean)

説明
フラグを設定してドキュメントのデバッグ情報をオンにします。

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

パラメータ
`flag` − デバッグ情報を有効にするかどうかを決定します。

setDoctype(DTD)

説明
DTD の通知を受け取ります。DTD を設定します。

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

パラメータ
`dtd` − ドキュメントの DTD を設定します。

例外
`org.xml.sax.SAXException` − SAX 例外で、別の例外をラップしている場合もあります。

setDocumentLocator(Locator)

説明

ドキュメント・イベントに関するロケータ・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。他のドキュメント・イベントで使用するロケータを格納する場合、アプリケーション開発者はサブクラスのこのメソッドを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

`locator` ― すべての SAX ドキュメント・イベントに関するロケータ

関連項目

`org.xml.sax.DocumentHandler.setDocumentLocator`、`org.xml.sax.Locator`

setNodeFactory(NodeFactory)

説明

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

構文

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

パラメータ

`f` ― NodeFactory

setTextDecl(String, String)

説明

テキスト XML 宣言の通知を受け取ります。パーサーは、テキスト XML 宣言ごとにこのメソッドをコールします。

構文

```
public void setTextDecl(java.lang.String version, java.lang.String encoding)
```

パラメータ

`version` ― バージョン番号（指定されない場合は null）

`encoding` ― エンコーディング名

例外

org.xml.sax.SAXException ― SAX 例外で、別の例外をラップしている場合もあります。

setXMLDecl(String, String, String)

説明

XML 宣言の通知を受け取ります。パーサーは、XML 宣言ごとにこのメソッドをコールします。

構文

```
public void setXMLDecl(java.lang.String version, java.lang.String standalone,
java.lang.String encoding)
```

パラメータ

version ― バージョン番号

standalone ― スタンドアロン値（指定されない場合は null）

encoding ― エンコーディング名（指定されない場合は null）

例外

org.xml.sax.SAXException ― SAX 例外で、別の例外をラップしている場合もあります。

startCDATA()

説明

CDATA セクション開始をレポートします。

構文

```
public void startCDATA()
```

例外

org.xml.sax.SAXException ― アプリケーションで例外が発生する場合があります。

関連項目

endCDATA()

startDocument()

説明

ドキュメント開始の通知を受け取ります。

構文

```
public void startDocument()
```

例外

`org.xml.sax.SAXException` － SAX 例外で、別の例外をラップしている場合もあります。

関連項目

`org.xml.sax.DocumentHandler.startDocument`

startDTD(String, String, String)

説明

DTD 宣言（存在する場合）開始をレポートします。

構文

```
public void startDTD(java.lang.String name, java.lang.String publicId,
java.lang.String systemId)
```

パラメータ

`name` － ドキュメント・タイプ名

`publicId` － 外部 DTD サブセットの宣言済み公開識別子、宣言済み公開識別子がない場合は null

`systemId` － 外部 DTD サブセットの宣言済みシステム識別子、宣言済みシステム識別子がない場合は null

例外

`org.xml.sax.SAXException` － アプリケーションで例外が発生する場合があります。

関連項目

`endDTD()`、`startEntity(String)`

startElement(NSName, SAXAttrList)

説明

要素開始の通知を受け取ります。

構文

```
public void startElement(NSName elem, SAXAttrList attrlist)
```

パラメータ

elem ― NSName オブジェクト

attrlist ― 要素の SAXAttrList

例外

org.xml.sax.SAXException ― SAXException が発生する場合があります。

関連項目

org.xml.sax.DocumentHandler.startElement

startElement(String, String, String, Attributes)

説明

要素開始の通知を受け取ります。

構文

```
public void startElement(java.lang.String namespaceURI, java.lang.String localName,
java.lang.String qName, org.xml.sax.Attributes atts)
```

パラメータ

namespaceURI ― 名前空間 URI（要素に名前空間 URI がない場合、または名前空間処理が実行されていない場合は空の文字列）

localName ― ローカル名（接頭辞なし）、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

qName ― 修飾名（接頭辞付き）、または修飾名が使用できない場合は空の文字列

atts ― 要素に連結されている属性、または属性がない場合は空の Attributes オブジェクト

例外

org.xml.sax.SAXException ― SAX 例外で、別の例外をラップしている場合もあります。

関連項目

endElement(NSName)、org.xml.sax.Attributes

startEntity(String)

説明

内部と外部の XML エンティティ開始をレポートします。startEntity と endEntity のすべてのイベントが正しくネストされている必要があります。

構文

```
public void startEntity(java.lang.String name)
```

パラメータ

name ― エンティティの名前。名前は、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。

例外

org.xml.sax.SAXException ― アプリケーションで例外が発生する場合があります。

関連項目

endEntity(String)、org.xml.sax.ext.DeclHandler、
org.xml.sax.ext.DeclHandler

DOMParser

DOMParser の構文

```
public class DOMParser
```

oracle.xml.parser.v2.DOMParser

DOMParser の説明

このクラスは、World Wide Web Consortium（W3C）の勧告に従って、XML 1.0 パーサーを実装して XML 文書を解析し、DOM ツリーを作成します。

DOMParser のフィールド

表 11-10 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・モードを Boolean.TRUE または Boolean.FALSE に設定します。
ERROR_ ENCODING	public static final java.lang.String ERROR_ENCODING	エラー・ストリーム（ERROR_STREAM が設定されている場合のみ）を使用したエラー・レポートのエンコーディング。
ERROR_STREAM	public static final java.lang.String ERROR_STREAM	エラー・レポート用のエラー・ストリーム。オブジェクトは OutputStream または PrintWriter です。ErrorHandler が設定されている場合、この属性は無視されます。
NODE_FACTORY	public static final java.lang.String NODE_FACTORY	NodeFactory を設定してカスタム・ノードを作成します。
SHOW_ WARNINGS	public static final java.lang.String SHOW_WARNINGS	警告を無視するかどうかを示すブール値。Boolean.TRUE または Boolean.FALSE に設定します。

DOMParser のコンストラクタ

DOMParser()

説明

新しいパーサー・オブジェクトを作成します。

構文

```
public  DOMParser()
```

DOMParser のメソッド

表 11-11 DOMParser のメソッドの概要

メソッド	説明
getAttribute(String)	実際の実装の特定の属性を取り出すことができます。
getDoctype()	DTD を取得します。
getDocument()	ドキュメントを取得します。
parseDTD(InputSource, String)	指定された入力ソースから XML 外部 DTD を解析します。
parseDTD(InputStream, String)	指定された入力ストリームから XML 外部 DTD を解析します。外部エンティティと DTD を解決するために、ベース URL を設定する必要があります。
parseDTD(Reader, String)	指定された入力ストリームから XML 外部 DTD を解析します。外部エンティティと DTD を解決するために、ベース URL を設定する必要があります。
parseDTD(String, String)	指定された URL から XML 外部 DTD を解析します。
parseDTD(URL, String)	指定された URL が指す XML 外部 DTD ドキュメントを解析し、対応する XML 文書階層を作成します。
reset()	パーサーの状態をリセットします。
retainCDATASection(boolean)	CDATA セクションを保持するかどうかを切り替えます。
setAttribute(String, Object)	実際の実装に特定の属性を設定できます。
setDebugMode(boolean)	フラグを設定してドキュメントのデバッグ情報をオンにします。
setErrorStream(OutputStream)	エラーと警告の出力に使用する出力ストリームを設定します。エラー用の出力ストリームが指定されない場合、パーサーは、エラーと警告の出力に標準のエラー出力ストリーム `System.err` を使用します。

表 11-11 DOMParser のメソッドの概要（続き）

メソッド	説明
setErrorStream(OutputStream, String)	エラーと警告の出力に使用する出力ストリームを設定します。エラー用の出力ストリームが指定されない場合、パーサーは、エラーと警告の出力に標準のエラー出力ストリーム System.err を使用します。また、指定し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。
setErrorStream(PrintWriter)	エラーと警告の出力に使用する出力ストリームを設定します。エラー用の出力ストリームが指定されない場合、パーサーは、エラーと警告の出力に標準のエラー出力ストリーム System.err を使用します。
setNodeFactory(NodeFactory)	ノード・ファクトリを設定します。アプリケーションで NodeFactory を拡張し、このメソッドを使用して登録できます。パーサーは、ユーザーが提供した NodeFactory を使用して、DOM ツリーのノードを作成します。
showWarnings(boolean)	警告の内容を出力するかどうかを切り替えます。

getAttribute(String)

説明

実際の実装の特定の属性を取り出すことができます。

構文

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

パラメータ

name ― 属性の名前

戻り値

属性の値

例外

IllegalArgumentException ― 実際の実装で属性を認識できない場合に発生します。

getDoctype()

説明
DTD を取得します。

構文
```
public DTD getDoctype()
```

戻り値
DTD

getDocument()

説明
ドキュメントを取得します。

構文
```
public XMLDocument getDocument()
```

戻り値
解析対象のドキュメント

parseDTD(InputSource, String)

説明
指定された入力ソースから XML 外部 DTD を解析します。

構文
```
public final void parseDTD(org.xml.sax.InputSource in, java.lang.String rootName)
```

パラメータ
in − 解析する org.xml.sax.InputSource

rootName − ルート要素として使用される要素

例外
XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

parseDTD(InputStream, String)

説明

指定された入力ストリームから XML 外部 DTD を解析します。外部エンティティと DTD を解決するために、ベース URL を設定する必要があります。

構文

```
public final void parseDTD(java.io.InputStream in, java.lang.String rootName)
```

パラメータ

in ― 解析する XML データを含んだ InputStream

rootName ― ルート要素として使用される要素

例外

XMLParseException ― 構文エラーまたはその他のエラーが発生した場合。

SAXException ― SAX 例外。他の例外をラップしている場合もあります。

IOException ― I/O エラー。

関連項目

XMLParser.setBaseURL(URL)

parseDTD(Reader, String)

説明

指定された入力ストリームから XML 外部 DTD を解析します。外部エンティティと DTD を解決するために、ベース URL を設定する必要があります。

構文

```
public final void parseDTD(java.io.Reader r, java.lang.String rootName)
```

パラメータ

r ― 解析する XML データを含んだ Reader

rootName ― ルート要素として使用される要素

例外

XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

関連項目

XMLParser.setBaseURL(URL)

parseDTD(String, String)

説明

指定された URL から XML 外部 DTD を解析します。

構文

```
public final void parseDTD(java.lang.String in, java.lang.String rootName)
```

パラメータ

in − 解析する URL を含んだ String

rootName − ルート要素として使用される要素

例外

XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

parseDTD(URL, String)

説明

指定された URL が指す XML 外部 DTD ドキュメントを解析し、対応する XML 文書階層を作成します。

構文

```
public final void parseDTD(java.net.URL url, java.lang.String rootName)
```

パラメータ

url − 解析する XML 文書を指し示す URL

rootName − ルート要素として使用される要素

例外

XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

reset()

説明

パーサーの状態をリセットします。

構文

```
public void reset()
```

retainCDATASection(boolean)

説明

CDATA セクションを保持するかどうかを切り替えます。

構文

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

パラメータ

flag − true（デフォルト）の場合は CDATA セクションを保持し、false の場合は CDATA セクションを Text ノードに変換します。

setAttribute(String, Object)

説明

実際の実装に特定の属性を設定できます。

構文

```
public void setAttribute(java.lang.String name, java.lang.Object value)
```

パラメータ

name − 属性の名前

value − 属性の値

例外

IllegalArgumentException ― 実際の実装で属性を認識できない場合に発生します。

setDebugMode(boolean)

説明

フラグを設定してドキュメントのデバッグ情報をオンにします。

構文

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

パラメータ

flag ― デバッグ情報を有効にするかどうかを決定します。

setErrorStream(OutputStream)

説明

エラーと警告の出力に使用する出力ストリームを設定します。エラー用の出力ストリームが指定されない場合、パーサーは、エラーと警告の出力に標準のエラー出力ストリーム System.err を使用します。

構文

```
public final void setErrorStream(java.io.OutputStream out)
```

パラメータ

out ― エラーと警告に使用する出力ストリーム

setErrorStream(OutputStream, String)

説明

エラーと警告の出力に使用する出力ストリームを設定します。エラー用の出力ストリームが指定されない場合、パーサーは、エラーと警告の出力に標準のエラー出力ストリーム System.err を使用します。また、指定し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。

構文

```
public final void setErrorStream(java.io.OutputStream out, java.lang.String enc)
```

パラメータ

out ― エラーと警告に使用する出力ストリーム

enc ― 使用するエンコーディング

例外

IOException ― サポートされていないエンコーディングが指定された場合

setErrorStream(PrintWriter)

説明

エラーと警告の出力に使用する出力ストリームを設定します。エラー用の出力ストリームが指定されない場合、パーサーは、エラーと警告の出力に標準のエラー出力ストリーム System.err を使用します。

構文

```
public final void setErrorStream(java.io.PrintWriter out)
```

パラメータ

out ― エラーと警告に使用する PrintWriter

例外

IOException ― エラー・ストリームの設定時に I/O エラーが発生した場合

setNodeFactory(NodeFactory)

説明

ノード・ファクトリを設定します。アプリケーションで NodeFactory を拡張し、このメソッドを使用して登録できます。パーサーは、ユーザーが提供した NodeFactory を使用して、DOM ツリーのノードを作成します。

構文

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

パラメータ

factory ― 設定する NodeFactory

例外

XMLParseException ― 無効なファクトリが設定された場合

関連項目

NodeFactory

showWarnings(boolean)

説明

警告の内容を出力するかどうかを切り替えます。

構文

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

パラメータ

`flag` ― 警告を表示するかどうかの決定

DTD

DTD の説明

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

DTD の構文

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

oracle.xml.parser.v2.DTD

DTD により実装されるインタフェース

java.io.Externalizable, java.io.Serializable

DTD のコンストラクタ

DTD()

説明

デフォルトのコンストラクタです。このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。

構文

```
public  DTD()
```

DTD のメソッド

表 11-12 DTD のメソッドの概要

メソッド	説明
findElementDecl(String)	指定されたタグ名の要素宣言を検索します。
findEntity(String, boolean)	DTD 内の名前付きエンティティを検索します。
findNotation(String)	DTD から名前表記法を取り出します。

表 11-12 DTD のメソッドの概要（続き）

メソッド	説明
getChildNodes()	このノードのすべての子ノードを含んだ NodeList を取得します。子がない場合は、ノードを含まない NodeList が戻されます。戻される NodeList の内容は「生きている」といえます。たとえば、ノード・オブジェクトの子に加えた変更は、NodeList アクセッサが戻すノードに即時に反映されます。つまり、ノードの内容の静的なスナップショットではありません。これは、getElementsByTagName メソッドが戻すものも含め、すべての NodeList に対して同様です。
getElementDecls()	DTD 内の要素宣言を含んだ NamedNodeMap を取得します。このマップ内のノードはすべて ElementDecl オブジェクトです。
getEntities()	DTD 内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んだ NamedNodeMap を取得します。重複部分は廃棄されます。たとえば、<!DOCTYPE ex SYSTEM "ex.dtd" [<!ENTITY foo "foo"> <!ENTITY bar "bar"> <!ENTITY % baz "baz">]> <ex/> では、インタフェースは foo と bar へのアクセスは提供しますが、baz へのアクセスは提供しません。このマップ内のすべてのノードは、Entity インタフェースも実装しています。DOM レベル 1 は、エンティティの編集をサポートしません。したがって、エンティティはこの方法では変更できません。
getInternalSubset()	DTD の内部サブセットを取得します。
getName()	DTD の名前を取得します。つまり DOCTYPE キーワードの直後の名前を取得します。
getNodeName()	DTD の名前を取得します。つまり DOCTYPE キーワードの直後の名前を取得します。
getNodeType()	実際のオブジェクトのタイプを示すコードを取得します。
getNotations()	DTD 内で宣言されている表記法を含んだ NamedNodeMap を取得します。重複部分は廃棄されます。このマップ内のノードはすべて、Notation インタフェースも実装しています。DOM レベル 1 は表記法の編集をサポートしません。したがって、表記法は、この方法では変更できません。
getOwnerImplementation()	DTD 実装の所有者を取得します。
getPublicId()	DTD に関連付けられている公開識別子を取得します（指定されている場合）。公開識別子が指定されていない場合は、null が戻されます。
getRootTag()	DTD のルート・タグを取得します。

表 11-12 DTD のメソッドの概要（続き）

メソッド	説明
getSystemId()	DTD に関連付けられているシステム識別子を取得します（指定されている場合）。システム識別子が指定されていない場合は、null が戻されます。
hasChildNodes()	ノードに子があるかどうかを簡単に判断できる便利なメソッドです。DTD が XMLNode のオーバーライド・メソッドを持てない場合は、常に false が戻されます。
normailize()	DTD を正規化します。
printExternalDTD(OutputStream)	このドキュメントの内容を、指定された出力ストリームに書き込みます。
printExternalDTD(OutputStream, String)	外部 DTD の内容を、指定された出力ストリームに書き込みます。
printExternalDTD(PrintWriter)	このドキュメントの内容を、指定された出力ストリームに書き込みます。
readExternal(ObjectInput)	このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。
setRootTag(String)	DTD のルート・タグを設定します。
writeExternal(ObjectOutput)	このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

findElementDecl(String)

説明

指定されたタグ名の要素宣言を検索します。

構文

```
public final ElementDecl findElementDecl(java.lang.String name)
```

パラメータ

name ― タグ名

戻り値

要素宣言オブジェクト

findEntity(String, boolean)

説明

DTD 内の名前付きエンティティを検索します。

構文

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

パラメータ

n ― エンティティの名前

par ― エンティティがパラメータ・エンティティかどうかを示すブール値

戻り値

指定された Entity オブジェクト。検出されない場合は null。

findNotation(String)

説明

DTD から名前表記法を取り出します。

構文

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

パラメータ

name ― 表記法の名前

戻り値

Notation オブジェクト。検出されない場合は null。

getChildNodes()

説明

ノードのすべての子ノードを含んだ NodeList を取得します。子がない場合は、ノードを含まない NodeList が戻されます。戻される NodeList の内容は「最新」といえます。たとえば、ノード・オブジェクトの子に加えた変更は、NodeList アクセッサが戻すノードに即時に反映されます。つまり、ノードの内容の静的なスナップショットではありません。これは、getElementsByTagName メソッドが戻すものも含め、すべての NodeList に対して同様です。

構文

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

戻り値

このノードの子

getElementDecls()

説明

DTD 内の要素宣言を含んだ NamedNodeMap を取得します。このマップ内のノードはすべて ElementDecl オブジェクトです。

構文

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

戻り値

DTD 内の要素宣言。DOM レベル 1 は、DTD 内の要素宣言の編集をサポートしません。したがって、DTD 内の要素宣言はこの方法では変更できません。

getEntities()

説明

DTD 内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んだ NamedNodeMap を取得します。重複部分は廃棄されます。たとえば、<!DOCTYPE ex SYSTEM "ex.dtd" [<!ENTITY foo "foo"> <!ENTITY bar "bar"> <!ENTITY % baz "baz">]> <ex/> では、インタフェースは foo と bar へのアクセスは提供しますが、baz へのアクセスは提供しません。このマップ内のすべてのノードは、Entity インタフェースも実装しています。DOM レベル 1 は、エンティティの編集をサポートしません。したがって、エンティティはこの方法では変更できません。

構文

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

戻り値

DTD 内で宣言されているエンティティ

getInternalSubset()

説明
DTD の内部サブセットを取得します。

構文
```
public java.lang.String getInternalSubset()
```

戻り値
内部サブセット宣言（文字列）

getName()

説明
DTD の名前を取得します。つまり DOCTYPE キーワードの直後の名前を取得します。

構文
```
public java.lang.String getName()
```

戻り値
DTD の名前

getNodeName()

説明
DTD の名前を取得します。つまり DOCTYPE キーワードの直後の名前を取得します。

構文
```
public java.lang.String getNodeName()
```

戻り値
DTD の名前

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

getNotations()

説明
DTD 内で宣言されている表記法を含んだ NamedNodeMap を取得します。重複部分は廃棄されます。このマップ内のノードはすべて、Notation インタフェースも実装しています。DOM レベル 1 は表記法の編集をサポートしません。したがって、表記法は、この方法では変更できません。

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

戻り値
DTD 内で宣言されている表記法

getOwnerImplementation()

説明
DTD 実装の所有者を取得します。

構文
```
public XMLDOMImplementation getOwnerImplementation()
```

戻り値
DTD が作成された実装

適用対象
DOM レベル 2

getPublicId()

説明

DTD に関連付けられている公開識別子を取得します（指定されている場合）。公開識別子が指定されていない場合は、null が戻されます。

構文

```
public java.lang.String getPublicId()
```

戻り値

DTD に関連付けられている公開識別子

getRootTag()

説明

DTD のルート・タグを取得します。

構文

```
public java.lang.String getRootTag()
```

戻り値

ルート・タグ

getSystemId()

説明

DTD に関連付けられているシステム識別子を取得します（指定されている場合）。システム識別子が指定されていない場合は、null が戻されます。

構文

```
public java.lang.String getSystemId()
```

戻り値

DTD に関連付けられているシステム識別子

hasChildNodes()

説明

ノードに子があるかどうかを簡単に判断できる便利なメソッドです。DTD が XMLNode のオーバーライド・メソッドを持てない場合は、常に false が戻されます。

構文

```
public boolean hasChildNodes()
```

戻り値

DTD が子を持てない場合は `false`

normalize()

説明

DTD を正規化します。

構文

```
public void normalize()
```

適用対象

DOM レベル 2

printExternalDTD(OutputStream)

説明

このドキュメントの内容を、指定された出力ストリームに書き込みます。

構文

```
public void printExternalDTD(java.io.OutputStream out)
```

パラメータ

`out` ― 書込み先の `OutputStream`

例外

`IOException` ― エラーが発生した場合

printExternalDTD(OutputStream, String)

説明

外部 DTD の内容を、指定された出力ストリームに書き込みます。

構文

```
public void printExternalDTD(java.io.OutputStream out, java.lang.String enc)
```

パラメータ

out ― 書込み先の OutputStream

enc ― 出力に使用するエンコーディング

例外

IOException ― 無効なエンコーディングが指定された場合、またはその他のエラーが発生した場合

printExternalDTD(PrintWriter)

説明

このドキュメントの内容を、指定された出力ストリームに書き込みます。

構文

```
public void printExternalDTD(java.io.PrintWriter out)
```

パラメータ

out ― 書込み先の PrintWriter

例外

IOException ― エラーが発生した場合

readExternal(ObjectInput)

説明

このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

構文

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

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ

`inArg` ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外

`IOException` ― 入力ストリームの読込み時にエラーがあった場合に発生します。

`ClassNotFoundException` ― クラスが検出されない場合に発生します。

setRootTag(String)

説明

DTD のルート・タグを設定します。

構文

```
public void setRootTag(java.lang.String root)
```

パラメータ

`root` ― ルート・タグ

writeExternal(ObjectOutput)

説明

このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文

```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

パラメータ

outArg ― シリアル化または圧縮されたストリームの書込みに使用する ObjectOutput ストリーム

例外

IOException ― シリアル化または圧縮されたストリームの書込み時に例外があった場合に発生します。

ElementDecl

ElementDecl の説明

このクラスは、DTD 内の要素宣言を表します。

ElementDecl の構文

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

oracle.xml.parser.v2.ElementDecl

ElementDecl により実装されるインタフェース

java.io.Externalizable, java.io.Serializable

ElementDecl のフィールド

表 11-13 ElementDecl のフィールド

フィールド	構文	説明
ANY	public static final byte ANY	要素のコンテンツ型 ー 子はいずれの要素でもかまいません。
ASTERISK	public static final int ASTERISK	ContentModelParseTreeNode タイプ ー 「*」ノード（子を 1 つ持ちます）。
COMMA	public static final int COMMA	ContentModelParseTreeNode タイプ ー 「,」ノード（子を 2 つ持ちます）。
ELEMENT	public static final int ELEMENT	ContentModelParseTreeNode タイプ ー 「leaf」ノード（子を持ちません）。
ELEMENT_ DECLARED	ublic static final int ELEMENT_ DECLARED	DTD の要素の宣言を表すノード・フラグ
ELEMENTS	public static final byte ELEMENTS	要素のコンテンツ型 ー 子をコンテンツ・モデルに応じた要素にできます。
EMPTY	public static final byte EMPTY	要素のコンテンツ型 ー 子なし。
ID_ATTR_DECL	public static final int ID_ATTR_ DECL	DTD の ID 属性の宣言を表すノード・フラグ
MIXED	public static final byte MIXED	要素のコンテンツ型 ー 子を PCDATA およびコンテンツ・モデルに応じた要素にできます。

表 11-13 ElementDecl のフィールド（続き）

フィールド	構文	説明
OR	public static final int OR	ContentModelParseTreeNode タイプ － 「|」ノード（子を 2 つ持ちます）。
PLUS	public static final int PLUS	ContentModelParseTreeNode タイプ － 「+」ノード（子を 1 つ持ちます）。
QMARK	public static final int QMARK	ContentModelParseTreeNode タイプ － 「?」ノード（子を 1 つ持ちます）。

ElementDecl のコンストラクタ

ElementDecl()

説明

デフォルトのコンストラクタです。このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。

構文

```
public  ElementDecl()
```

ElementDecl のメソッド

表 11-14 ElementDecl のメソッドの概要

メソッド	説明
cloneNode(boolean)	このノードの複製を戻します。つまり、ノードの汎用コピー・コンストラクタとして機能します。複製ノードには親はありません（parentNode は null を戻します）。Element をクローニングすると、属性とその値がすべて（デフォルトの属性を示すために XML プロセッサによって生成された属性とその値も含む）コピーされますが、このメソッドでは、ディープ・クローンでないかぎり、それに含まれるテキストをコピーしません。これは、テキストが子の Text ノードに含まれているためです。他のタイプのノードをクローニングすると、単純にこのノードのコピーが戻されます。
expectedElements(Element)	要素に付加できる要素名のベクトルを戻します。

表 11-14 ElementDecl のメソッドの概要（続き）

メソッド	説明
findAttrDecl(String)	属性宣言オブジェクトを取得します。または検出されない場合は null が戻されます。
getAttrDecls()	属性宣言の一覧を取得します。
getContentElements()	この要素に追加できる要素のベクトルを戻します。
getContentType()	要素のコンテンツ・モデルを戻します。
getNodeName()	Element Decl の名前を取得します。
getNodeType()	実際のオブジェクトのタイプを示すコードを取得します。
getParseTree()	コンテンツ・モデル解析ツリーのルート・ノードを戻します。Node.getFirstChild() および Node.getLastChild() は、解析ツリーの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・タイプとノード名を戻します。
readExternal(ObjectInput)	このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。
validateContent(Element)	要素ノードの内容の妥当性をチェックします。
writeExternal(ObjectOutput)	このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

cloneNode(boolean)

説明

このノードの複製を戻します。つまり、ノードの汎用コピー・コンストラクタとして機能します。複製ノードには親はありません（parentNode は null を戻します）。Element をクローニングすると、属性とその値がすべて（デフォルトの属性を示すために XML プロセッサによって生成された属性とその値も含む）コピーされますが、このメソッドでは、ディープ・クローンでないかぎり、それに含まれるテキストをコピーしません。これは、テキストが子の Text ノードに含まれているためです。他のタイプのノードをクローニングすると、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
deep ― true の場合は、指定したノードの下位サブツリーが再帰的にクローニングされ、false の場合は、そのノード（Element の場合はその属性）のみクローニングされます。

戻り値
複製ノード

expectedElements(Element)

説明
要素に付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
e ― 要素

戻り値
名前のベクトル

findAttrDecl(String)

説明
属性宣言オブジェクトを取得します。または検出されない場合は null が戻されます。

構文
```
public final AttrDecl findAttrDecl(java.lang.String name)
```

パラメータ
name ― 検索する属性宣言

戻り値
AttrDecl オブジェクト、または検出されない場合は null

getAttrDecls()

説明
属性宣言の一覧を取得します。

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

戻り値
属性宣言の一覧

getContentElements()

説明
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()
```

戻り値
要素名を含んだベクトル

getContentType()

説明
要素のコンテンツ・モデルを戻します。

構文
```
public int getContentType()
```

戻り値
要素宣言のタイプ

getNodeName()

説明
Element Decl の名前を取得します。

構文
```
public java.lang.String getNodeName()
```

戻り値
ノードの名前

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

getParseTree()

説明
コンテンツ・モデル解析ツリーのルート・ノードを戻します。Node.getFirstChild() および Node.getLastChild() は、解析ツリーの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・タイプとノード名を戻します。

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

戻り値
コンテンツ・モデル解析ツリーのルート・ノードを含んだノード

readExternal(ObjectInput)

説明

このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

構文

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

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ

`inArg` ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外

`IOException` ― 入力ストリームの読込み時にエラーがあった場合に発生します。

`ClassNotFoundException` ― クラスが検出されない場合に発生します。

validateContent(Element)

説明

要素ノードの内容の妥当性をチェックします。

構文

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

戻り値

有効な場合は true、それ以外は false

writeExternal(ObjectOutput)

説明

このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文

```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

パラメータ

`outArg` ― シリアル化または圧縮されたストリームの書込みに使用する ObjectOutput ストリーム

例外

`IOException` ― シリアル化または圧縮されたストリームの書込み時に例外があった場合に発生します。

NodeFactory

NodeFactory の説明

このクラスは、解析時に作成される 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、解析時に DOM ツリーに追加される独自のカスタム・クラスを作成できます。その場合、DOMParser の setNodeFactory() メソッドを使用して、独自の NodeFactory 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、DOM ツリーにノードは追加されません。

NodeFactory の構文

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

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

NodeFactory により実装されるインタフェース

```
java.io.Serializable
```

関連項目

```
DOMParser.setNodeFactory(NodeFactory)
```

NodeFactory のコンストラクタ

NodeFactory()

構文

```
public  NodeFactory()
```

NodeFactory のメソッド

表 11-15 NodeFactory のメソッドの概要

メソッド	説明
createAttribute(String, String)	指定されたタグとテキストで属性ノードを作成します。
createAttribute(String, String, String, String)	指定されたタグとテキストで属性ノードを作成します。
createCDATASection(String)	指定されたテキストで CDATA ノードを作成します。
createComment(String)	指定されたテキストでコメント・ノードを作成します。
createDocument()	ドキュメント・ノードを作成します。このメソッドでは、null ポインタを戻すことができません。
createDocumentFragment()	ドキュメント・フラグメント・ノードを作成します。
createElement(String)	指定されたタグで要素ノードを作成します。
createElementNS(String, String, String)	指定されたローカル名、接頭辞および名前空間 URI で要素ノードを作成します。
createEntityReference(String)	指定されたタグでエンティティ参照ノードを作成します。
createProcessingInstruction(String, String)	指定されたタグとテキストで PI ノードを作成します。
createTextNode(String)	指定されたテキストでテキスト・ノードを作成します。

createAttribute(String, String)

説明

指定されたタグとテキストで属性ノードを作成します。

構文

```
public XMLAttr createAttribute(java.lang.String tag, java.lang.String text)
```

パラメータ

`tag` − ノードの名前

`text` − ノードに関連付けるテキスト

戻り値

作成された属性ノード

createAttribute(String, String, String, String)

説明
指定されたタグとテキストで属性ノードを作成します。

構文
```
public XMLAttr createAttribute(java.lang.String localName, java.lang.String prefix,
java.lang.String namespaceURI, java.lang.String value)
```

パラメータ
`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` − ノードに関連付ける値

戻り値
作成された属性ノード

createCDATASection(String)

説明
指定されたテキストで CDATA ノードを作成します。

構文
```
public XMLCDATA createCDATASection(java.lang.String text)
```

パラメータ
`text` − ノードに関連付けるテキスト

戻り値
作成された CDATA ノード

createComment(String)

説明
指定されたテキストでコメント・ノードを作成します。

構文
```
public XMLComment createComment(java.lang.String text)
```

パラメータ
text ― ノードに関連付けるテキスト

戻り値
作成されたコメント・ノード

createDocument()

説明
ドキュメント・ノードを作成します。このメソッドでは、null ポインタを戻すことができません。

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

戻り値
作成された要素

createDocumentFragment()

説明
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(String)

説明
指定されたタグで要素ノードを作成します。

構文
```
public XMLElement createElement(java.lang.String tag)
```

パラメータ
tag ― 要素の名前

戻り値
作成された要素

createElementNS(String, String, String)

説明
指定されたローカル名、接頭辞および名前空間 URI で要素ノードを作成します。

構文
```
public XMLElement createElementNS(java.lang.String localName, java.lang.String
prefix, java.lang.String namespaceURI)
```

パラメータ
localName ― 要素の名前

prefix ― 要素の接頭辞

namespaceURI ― 要素の名前空間

戻り値
作成された要素

createEntityReference(String)

説明
指定されたタグでエンティティ参照ノードを作成します。

構文
```
public XMLEntityReference createEntityReference(java.lang.String tag)
```

パラメータ
tag ― ノードの名前

戻り値
作成されたエンティティ参照ノード

createProcessingInstruction(String, String)

説明
指定されたタグとテキストで PI ノードを作成します。

構文
```
public XMLPI createProcessingInstruction(java.lang.String tag, java.lang.String
text)
```

パラメータ
tag ― ノードの名前

text ― ノードに関連付けるテキスト

戻り値
作成された PI ノード

createTextNode(String)

説明
指定されたテキストでテキスト・ノードを作成します。

構文
```
public XMLText createTextNode(java.lang.String text)
```

パラメータ
text ― ノードに関連付けるテキスト

戻り値
作成されたテキスト・ノード

oraxml

oraxml の説明

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

表 11-16 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、正しい形式であることをチェックします。
-dtd	DTD 妥当性チェックを使用して入力ファイルの妥当性をチェックします。
-schema	スキーマ妥当性チェックを使用して入力ファイルの妥当性をチェックします。
-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	圧縮された入力ファイルを解凍します。
-enc	入力ファイルのエンコーディングを出力します。
-warning	警告を表示します。

oraxml の構文

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

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

oraxml のコンストラクタ

oraxml()

構文

```
public  oraxml()
```

oraxml のメソッド

main(String[])

構文

```
public static void main(java.lang.String[] args)
```

SAXAttrList

SAXAttrList の説明

このクラスは、SAX AttributeList インタフェースを実装します。また、名前空間サポートも提供します。名前空間サポートを必要とするアプリケーションでは、Oracle パーサー・クラスによって戻された属性リストを明示的に SAXAttrList にキャストでき、ここで説明するメソッドを使用できます。このクラスは、Attributes（SAX 2.0）インタフェースも実装します。

SAXAttrList の構文

```
public class SAXAttrList

oracle.xml.parser.v2.SAXAttrList
```

SAXAttrList に関するコメント

このインタフェースを使用して属性リストにアクセスするには、次の 3 つの方法があります。

- 属性のインデックスを指定
- 名前空間と修飾名の組合せを指定
- 修飾名（接頭辞付き）を指定

このリストには、#IMPLIED として宣言されているが、開始タグに指定されていない属性は含まれません。また、http://xml.org/sax/features/namespace-prefixes 機能が true に設定されていないかぎり（デフォルトは false）、名前空間宣言（xmlns*）として使用する属性は含まれません。

前述の namespace-prefixes 機能が false の場合、修飾名によるアクセスは使用できません。http://xml.org/sax/features/namespaces 機能が false の場合、名前空間と修飾名の組合せによるアクセスは使用できません。

このインタフェースによって、現在は使用できない SAX1 インタフェース（名前空間をサポートしない）が置換されます。また、名前空間サポート以外に、getIndex メソッド（後述）も追加されます。

リスト内の属性の順序は指定されていないため、実装によって異なります。

SAXAttrList のコンストラクタ

SAXAttrList(int)

構文

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

SAXAttrList のメソッド

addAttr(String, String, String, String, boolean, int)

説明

属性を親要素ノードに追加します。

構文

```
public void addAttr(java.lang.String pfx, java.lang.String lname, java.lang.String
tag, java.lang.String value, boolean spec, int type)
```

パラメータ

pfx ― 属性の接頭辞

lname ― 属性のローカル名

tag ― 属性の修飾名

value ― 属性の値

spec ― 指定したフラグ

type ― 属性の型

addAttr(String, String, String, String, boolean, int, String)

説明

属性を親要素ノードに追加します。

構文

```
public void addAttr(java.lang.String pfx, java.lang.String lname, java.lang.String
tag, java.lang.String value, boolean spec, int type, java.lang.String nmsp)
```

パラメータ

pfx ― 属性の接頭辞

lname ― 属性のローカル名

tag ― 属性の修飾名

value ― 属性の値

spec ― 指定したフラグ

type ― 属性の型

nmsp ― 属性の名前空間

getExpandedName(int)

説明

リスト内の属性に対する拡張名を取得します（位置を指定）。

構文

```
public java.lang.String getExpandedName(int i)
```

パラメータ

i ― リスト内の属性のインデックス

戻り値

属性の拡張名

getIndex(String)

説明

XML 1.0 の修飾名を指定して、属性のインデックスを検索します。

構文

```
public int getIndex(java.lang.String qName)
```

パラメータ

qName ― 修飾名（接頭辞付き）

戻り値

属性のインデックス、リストに表示されない場合は -1

getIndex(String, String)

説明
名前空間名を指定して、属性のインデックスを検索します。

構文
```
public int getIndex(java.lang.String uri, java.lang.String localPart)
```

パラメータ
uri ― 名前空間 URI、または名前に名前空間 URI がない場合は空の文字列

localPart ― 属性のローカル名

戻り値
属性のインデックス、リストに表示されない場合は -1

適用対象
SAX2

getLength()

説明
このリスト内の属性の数を戻します。

構文
```
public int getLength()
```

コメント
SAX パーサーは、属性の宣言順または指定順に関係なく、任意の順序で属性を提供します。属性の数は 0（ゼロ）の場合もあります。

戻り値
リスト内の属性の数

getLocalName(int)

説明
インデックスを指定して属性のローカル名を検索します。

構文
```
public java.lang.String getLocalName(int index)
```

パラメータ
index ― 属性のインデックス（0（ゼロ）から開始）

戻り値
ローカル名、空の文字列（名前空間処理が実行されていない場合）または null（インデックスが有効範囲外の場合）

適用対象
SAX2

関連項目
getLength()

getPrefix(int)

説明
リスト内の属性に対する名前空間接頭辞を取得します（位置を指定）。

構文
```
public java.lang.String getPrefix(int i)
```

パラメータ
i ― リスト内の属性のインデックス

戻り値
属性に対する名前空間接頭辞

getQName(int)

説明
インデックスを指定して属性の XML 1.0 の修飾名を検索します。

構文
```
public java.lang.String getQName(int index)
```

パラメータ
index ― 属性のインデックス（0（ゼロ）から開始）

戻り値
XML 1.0 の修飾名、空の文字列（使用可能な修飾名がない場合）または null（インデックスが有効範囲外の場合）

適用対象
SAX2

関連項目
```
getLength()
```

getType(int)

説明
インデックスを指定して属性の型を検索します。

構文
```
public java.lang.String getType(int index)
```

コメント
属性の型は、"CDATA"、"ID"、"IDREF"、"IDREFS"、"NMTOKEN"、"NMTOKENS"、"ENTITY"、"ENTITIES" または "NOTATION"（常に大文字）のいずれかの文字列です。

パーサーが属性の宣言を読み込んでいない場合または属性の型をレポートしない場合、そのパーサーは、XML 1.0 の勧告（3.3.3 節「Attribute-Value Normalization」）に従って、"CDATA" 値を戻す必要があります。

列挙された属性が表記法以外の場合、パーサーはタイプとして "NMTOKEN" をレポートします。

パラメータ
index ― 属性のインデックス（0（ゼロ）から開始）

戻り値
属性の型（文字列）、またはインデックスが有効範囲外の場合は null

関連項目
getLength()

getType(String)

説明
XML 1.0 の修飾名を指定して属性の型を検索します。

構文
```
public java.lang.String getType(java.lang.String qName)
```

コメント
可能なタイプの説明は、getType(int) を参照してください。

パラメータ
qName ― XML 1.0 の修飾名

戻り値
属性の型（文字列）または null（属性がリストにない場合、または修飾名が使用できない場合）

getType(String, String)

説明
名前空間名を指定して属性の型を検索します。

構文
```
public java.lang.String getType(java.lang.String uri, java.lang.String localName)
```

コメント
可能なタイプの説明は、getType(int) を参照してください。

パラメータ
uri ― 名前空間 URI、または名前に名前空間 URI がない場合は空の文字列

localName ― 属性のローカル名

戻り値
属性の型（文字列）または null（属性がリストにない場合、または名前空間処理が実行されていない場合）

適用対象
SAX2

getURI(int)

説明
インデックスを指定して属性の名前空間 URI を検索します。

構文
```
public java.lang.String getURI(int index)
```

パラメータ
index ― 属性のインデックス（0（ゼロ）から開始）

戻り値
名前空間 URI、空の文字列（使用可能な名前空間 URI がない場合）または null（インデックスが有効範囲外の場合）

適用対象
SAX2

関連項目
getLength()

getValue(int)

説明
インデックスを指定して属性の値を検索します。

構文
```
public java.lang.String getValue(int index)
```

説明
属性値がトークン（IDREFS、ENTITIES または NMTOKENS）のリストである場合、各トークンは単一の空白で区切られて単一の文字列に連結されます。

パラメータ
index ― 属性のインデックス（0（ゼロ）から開始）

戻り値
属性の値（文字列）、またはインデックスが有効範囲外の場合は null

関連項目
getLength()

getValue(String)

説明
XML 1.0 の修飾名を指定して属性の値を検索します。

構文
```
public java.lang.String getValue(java.lang.String qName)
```

コメント
可能な値の説明は、getValue(int) を参照してください。

パラメータ
qName ― XML 1.0 の修飾名

戻り値
属性値（文字列）または null（属性がリストにない場合、または修飾名が使用できない場合）

getValue(String, String)

説明
名前空間名を指定して属性の値を検索します。

構文
```
public java.lang.String getValue(java.lang.String uri, java.lang.String localName)
```

コメント
可能な値の説明は、`getValue(int)` を参照してください。

パラメータ
`uri` ― 名前空間 URI、または名前に名前空間 URI がない場合は空の文字列

`localName` ― 属性のローカル名

戻り値
属性値（文字列）、または属性がリストにない場合は null

適用対象
SAX2

reset()

説明
SAXAttrList をリセットします。

構文
```
public void reset()
```

SAXParser

構文

```
public class SAXParser
```

oracle.xml.parser.v2.SAXParser

説明

このクラスは、World Wide Web Consortium（W3C）の勧告に従って、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 ハンドラを登録すると、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 ドライバで実装する必要が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、パーサーの機能とプロパティの設定および問合せを行い、ドキュメント処理用のイベント・ハンドラを登録して、ドキュメント解析を開始できます。

すべての SAX インタフェースは、同期しているとみなされます。解析メソッドは解析が完了するまで戻すことができないため、Reader は、イベント・ハンドラのコールバックが戻るまで待機してから次のイベントをレポートする必要があります。

このインタフェースによって、SAX 1.0 パーサー・インタフェース（現在は使用できません）が置換されます。XMLReader インタフェースには、古いパーサー・インタフェースにかわって、次の重要な拡張機能が含まれています。

- 機能とプロパティの問合せと設定を行う標準的な方法が追加されました。
- 上位レベルの多くの XML 標準に必要な名前空間サポートが追加されました。

コンストラクタ

SAXParser()

説明

新しいパーサー・オブジェクトを作成します。

構文

```
public  SAXParser()
```

メソッド

getContentHandler()

説明
現行のコンテンツ・ハンドラを戻します。

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

戻り値
現行のコンテンツ・ハンドラ、またはコンテンツ・ハンドラが登録されていない場合は null

適用対象
SAX 2.0

関連項目
`setContentHandler(ContentHandler)`

getDTDHandler()

説明
現行の DTD ハンドラを戻します。

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

戻り値
現行の DTD ハンドラ、または DTD ハンドラが登録されていない場合は null

適用対象
SAX 2.0

関連項目
`setDTDHandler(DTDHandler)`

getFeature(String)

説明

機能の値を検索します。

構文

```
public boolean getFeature(java.lang.String name)
```

コメント

機能名は、完全に修飾された URI です。XMLReader は、機能名を認識できますが、その値を戻すことはできません。これは、SAX1 パーサーのアダプタの場合も同様で、実際のパーサーが妥当性チェックを実行しているのか、外部エンティティを拡張しているのかを判別する方法はありません。

すべての XMLReader では、`http://xml.org/sax/features/namespaces` および `http://xml.org/sax/features/namespace-prefixes` の機能名を認識する必要があります。

特定のコンテキスト（解析前、解析中、解析後など）でのみ使用可能な機能値もあります。

一般的な使用方法を次に示します。

```
XMLReader r = new MySAXDriver();

                        // try to activate validation
try {
  r.setFeature("http://xml.org/sax/features/validation", true);
} catch (SAXException e) {
  System.err.println("Cannot activate validation.");
}

                        // register event handlers
r.setContentHandler(new MyContentHandler());
r.setErrorHandler(new MyErrorHandler());

                        // parse the first document
try {
  r.parse("http://www.foo.com/mydoc.xml");
} catch (IOException e) {
  System.err.println("I/O exception reading XML document");
} catch (SAXException e) {
  System.err.println("XML exception reading document.");
}
```

機能 "http://xml.org/sax/features/validation" は入力値がバイナリであるため、制御の対象は DTD 妥当性チェックのみです。値が true の場合は、DTD 妥当性チェックが true に設定

されます。この機能を使用して XML Schema に基づいた妥当性チェックを制御することはできません。

実装者は、独自の URI で作成した名前を使用して、独自の機能を自由に開発できます。

パラメータ

`feature` ― 機能名（完全に修飾された URI）

`version` ― 機能のバージョン

戻り値

機能の現在の状態（true または false）

例外

`org.xml.sax.SAXNotRecognizedException` ― XMLReader が機能名を認識しない場合

`org.xml.sax.SAXNotSupportedException` ― XMLReader は機能名を認識しているが、その値をすぐに判断できない場合

関連項目

`setFeature(String, boolean)`

getProperty(String)

説明

プロパティの値を検索します。

構文

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

コメント

プロパティ名は、完全に修飾された URI です。XMLReader は、プロパティ名を認識できますが、その状態を戻すことはできません。これは、SAX1 パーサーのアダプタの場合も同様です。

初期のコア・セットは SAX2 用に記述されていますが、XMLReader による特定プロパティ名の認識は不要です。

特定のコンテキスト（解析前、解析中、解析後など）でのみ使用可能なプロパティ値もあります。

実装者は、独自の URI 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できます。

パラメータ
name ― プロパティ名（完全に修飾された URI）

戻り値
プロパティの現行値

例外
org.xml.sax.SAXNotRecognizedException ― XMLReader がプロパティ名を認識しない場合

org.xml.sax.SAXNotSupportedException ― XMLReader はプロパティ名を認識しているが、その値をすぐに判断できない場合

関連項目
setProperty(String, Object)

setContentHandler(ContentHandler)

説明
アプリケーションでコンテンツ・イベント・ハンドラを登録できます。

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

コメント
アプリケーションによるコンテンツ・ハンドラの登録がない場合、SAX パーサーでレポートされるすべてのコンテンツ・イベントは無視されます。

アプリケーションは解析中に新規または別のハンドラを登録でき、SAX パーサーはその新規ハンドラの使用を即時に開始する必要があります。

パラメータ
handler ― コンテンツ・ハンドラ

例外
java.lang.NullPointerException ― ハンドラの引数が null の場合

適用対象
SAX 2.0

関連項目
`getContentHandler()`

setDTDHandler(DTDHandler)

説明
アプリケーションで DTD イベント・ハンドラを登録できます。

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

コメント
アプリケーションによる DTD ハンドラの登録がない場合、SAX パーサーでレポートされるすべての DTD イベントは無視されます。

アプリケーションは解析中に新規または別のハンドラを登録でき、SAX パーサーはその新規ハンドラの使用を即時に開始する必要があります。

パラメータ
`handler` − DTD ハンドラ

例外
`java.lang.NullPointerException` − ハンドラの引数が null の場合

関連項目
`getDTDHandler()`

setFeature(String, boolean)

説明
機能の状態を設定します。

構文
```
public void setFeature(java.lang.String name, boolean state)
```

コメント

機能名は、完全に修飾された URI です。XMLReader は、機能名を認識できますが、その値を設定することはできません。これは、SAX1 パーサーのアダプタの場合も同様で、実際のパーサーが、次のような妥当性チェックを実行するかどうかについて影響を与える方法はありません。

すべての XMLReader は、`http://xml.org/sax/features/namespaces` の true への設定、`http://xml.org/sax/features/namespace-prefixes` の false への設定をサポートする必要があります。

特定のコンテキスト（解析前、解析中、解析後など）でのみ変更可能または変更不可の機能値もあります。

機能 "http://xml.org/sax/features/validation" は入力値がバイナリであるため、制御の対象は DTD 妥当性チェックのみです。値が true の場合は、DTD 妥当性チェックが true に設定されます。この機能を使用して XML Schema に基づいた妥当性チェックを制御することはできません。

パラメータ

`name` ― 機能名（完全に修飾された URI）

`state` ― 機能の要求された状態（true または false）

例外

`org.xml.sax.SAXNotRecognizedException` ― XMLReader が機能名を認識しない場合

`org.xml.sax.SAXNotSupportedException` ― XMLReader は機能名を認識しているが、要求された値を設定できない場合

関連項目

`getFeature(String)`

setProperty(String, Object)

説明

プロパティの値を設定します。

構文

```
public void setProperty(java.lang.String name, java.lang.Object value)
```

コメント

プロパティ名は、完全に修飾された URI です。XMLReader は、プロパティ名を認識できますが、その値を設定することはできません。これは、SAX1 パーサーのアダプタの場合も同様です。

初期のコア・セットは SAX2 で提供されていますが、XMLReader で特定プロパティ名の設定を認識する必要はありません。

特定のコンテキスト（解析前、解析中、解析後など）でのみ変更可能または変更不可のプロパティ値もあります。

このメソッドは、拡張ハンドラを設定するための標準的な機能でもあります。

パラメータ

`name` ― プロパティ名（完全に修飾された URI）

`value` ― 要求されたプロパティの値

例外

`org.xml.sax.SAXNotRecognizedException` ― XMLReader がプロパティ名を認識しない場合

`org.xml.sax.SAXNotSupportedException` ― XMLReader はプロパティ名を認識しているが、要求された値を設定できない場合

XMLAttr

XMLAttr の構文

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

oracle.xml.parser.v2.XMLAttr

XMLAttr により実装されるインタフェース

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

説明

このクラスは、DOM の Attr インタフェースを実装し、要素の各属性に関する情報を保持します。

関連項目

Attr、NodeFactory、DOMParser.setNodeFactory(NodeFactory)

XMLAttr()

説明

デフォルトのコンストラクタです。

構文

```
public  XMLAttr()
```

コメント

使用できません。XMLDocument の createAttribute(String) または createAttributeNS(String, String) を使用してください。

このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。通常の XMLAttr を作成する場合は、XMLDocument の createAttribute(String) または createAttributeNS(String, String) を使用してください。

XMLAttr(String, String)

説明

指定された名前と値で属性を構成します。

構文

```
public  XMLAttr(java.lang.String n, java.lang.String v)
```

コメント

使用できません。XMLDocument の createAttribute(String) メソッドを使用してください。

パラメータ

n ― 属性の名前

v ― 属性の値

XMLAttr(String, String, String, String)

説明

名前空間サポート

構文

```
public  XMLAttr(java.lang.String name, java.lang.String prefix, java.lang.String
namespace, java.lang.String v)
```

コメント

使用できません。XMLDocument の createAttributeNS(String, String) メソッドを使用してください。

パラメータ

name ― 属性のローカル名

prefix ― 属性の接頭辞

namespace ― 属性の名前空間

v ― 属性の値

XMLAttr(String, String, String, String, String)

説明

プライベート・コンストラクタです（名前の制限はありません）。

構文

```
public  XMLAttr(java.lang.String name, java.lang.String prefix, java.lang.String
qname, java.lang.String namespace, java.lang.String v)
```

コメント

使用できません。XMLDocument の crateAttribute(String) または createAttributeNS(String, String) を使用してください。

パラメータ

name ― 属性のローカル名

prefix ― 属性の接頭辞

qname ― 属性の修飾名

namespace ― 属性の名前空間

v ― 属性の値

メソッド

addText(String)

説明

XML Node にテキストを追加します。

構文

```
public XMLNode addText(java.lang.String str)
```

cloneNode(boolean)

説明

このノードの複製を戻します。つまり、ノードの汎用コピー・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)
```

コメント
複製ノードには親はありません（parentNode は null を戻します）。Element をクローニングすると、属性とその値がすべて（デフォルトの属性を示すために XML プロセッサによって生成された属性とその値も含む）コピーされますが、このメソッドでは、ディープ・クローンでないかぎり、それに含まれるテキストをコピーしません。これは、テキストが子の Text ノードに含まれているためです。他のタイプのノードをクローニングすると、単純にこのノードのコピーが戻されます。

パラメータ
deep ― true の場合は、指定したノードの下位サブツリーが再帰的にクローニングされ、false の場合は、そのノード（Element の場合はその属性）のみクローニングされます。

戻り値
複製ノード

getExpandedName()

説明
この属性の解決済み完全名を取得します。

構文
```
public java.lang.String getExpandedName()
```

指定方法
インタフェース oracle.xml.util.NSName 内の oracle.xml.util.NSName.getExpandedName()

戻り値
解決済み完全名

getLocalName()

説明
この属性のローカル名を取得します。

構文
```
public java.lang.String getLocalName()
```

指定方法
インタフェース oracle.xml.util.NSName 内の oracle.xml.util.NSName.getLocalName()

戻り値
属性のローカル名

適用対象
DOM レベル 2

getName()

説明
属性名を取得します。

構文
```
public java.lang.String getName()
```

戻り値
属性名

getNamespaceURI()

説明
属性の名前空間を取得します。

構文
```
public java.lang.String getNamespaceURI()
```

戻り値
この属性に関連付けられた名前空間 URI

適用対象
DOM レベル 2

getNextAttribute()

説明
次の属性（存在する場合）を取得します。

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

戻り値
次の属性

getNextSibling()

説明
次の兄弟関係（存在する場合）を取得します。

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

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

getNodeValue()

説明
このノードの値を、そのタイプに基づいて取得します。

構文
```
public java.lang.String getNodeValue()
```

戻り値
このノードの値

例外
DOMException − NO_MODIFICATION_ALLOWED_ERR: ノードが読取り専用の場合に発生します。DOMSTRING_SIZE_ERR: 実装プラットフォームにおける DOMString 変数に収まらない文字が戻された場合に発生します。

getOwnerElement()

説明
この属性を所有する要素を取得します。

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

戻り値
この属性を所有する要素ノード

適用対象
DOM レベル 2

getParentNode()

説明
このノードの親を取得します。

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

コメント
Document、DocumentFragment および Attr を除くすべてのノードに親が存在する可能性があります。ただし、ノードが作成されたばかりでまだ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。

戻り値
このノードの親

getPrefix()

説明
要素の名前空間接頭辞を取得します。

構文
```
public java.lang.String getPrefix()
```

指定方法
インタフェース oracle.xml.util.NSName 内の oracle.xml.util.NSName.getPrefix()

戻り値
この属性の名前空間接頭辞

適用対象
DOM レベル 2

getPreviousSibling()

説明
前の兄弟関係（存在する場合）を取得します。

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

getSpecified()

説明
要素内に属性が明示的に指定されている場合は true を戻します。

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

戻り値
属性が明示的に指定されている場合は `true`、明示的に指定されていない場合は `false`

getValue()

説明
属性値を取得します。

構文
```
public java.lang.String getValue()
```

戻り値
属性値

readExternal(ObjectInput)

説明
このメソッドは、writeExternal によって書き込まれた情報をリストアします。

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

指定方法
インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ
`inArg` ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外
`IOException` ― 圧縮ストリームの読込み時に例外があった場合に発生します。

`ClassNotFoundException` ― クラスが検出されない場合に発生します。

setNodeValue(String)

説明
このノードの値を、そのタイプに基づいて設定します。

構文
```
public void setNodeValue(java.lang.String nodeValue)
```

パラメータ
nodeValue − 設定するノードの値

例外
DOMException − NO_MODIFICATION_ALLOWED_ERR: ノードが読取り専用の場合に発生します。DOMSTRING_SIZE_ERR: 実装プラットフォームにおける DOMString 変数に収まらない文字が戻された場合に発生します。

setValue(String)

説明
値を設定します。

構文
```
public void setValue(java.lang.String arg)
```

パラメータ
arg − 設定する値

writeExternal(ObjectOutput)

説明
このメソッドは、オブジェクトの状態を保存します。オブジェクト情報は、バイナリの圧縮ストリームに保存されます。

構文
```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法
インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

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

例外
IOException − 圧縮ストリームの書込み時に例外があった場合に発生します。

XMLCDATA

説明
このクラスは、DOM の CDATASection インタフェースを実装します。

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

oracle.xml.parser.v2.XMLCDATA

実装されるインタフェース
java.io.Externalizable, java.io.Serializable

関連項目
CDATASection、NodeFactory、DOMParser.setNodeFactory(NodeFactory)

コンストラクタ

XMLCDATA()

説明
デフォルトのコンストラクタです。

構文
```
public  XMLCDATA()
```

コメント
このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。

メソッド

getNodeName()

説明
ノードの名前を取得します。

構文
```
public java.lang.String getNodeName()
```

戻り値
ノードの名前

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

readExternal(ObjectInput)

説明
このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

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

指定方法
インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ

inArg ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外

IOException ― 入力ストリームの読込み時にエラーがあった場合に発生します。

ClassNotFoundException ― クラスが検出されない場合に発生します。

writeExternal(ObjectOutput)

説明

このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文

```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

パラメータ

outArg ― 圧縮ストリームの書込みに使用する ObjectOutput ストリーム

例外

IOException ― 圧縮ストリームの書込み時に例外があった場合に発生します。

XMLComment

説明
このクラスは、DOM の Comment インタフェースを実装します。

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

oracle.xml.parser.v2.XMLComment

実装されるインタフェース
java.io.Externalizable, java.io.Serializable

関連項目
Comment、NodeFactory、DOMParser.setNodeFactory(NodeFactory)

コンストラクタ

XMLComment()

説明
デフォルトのコンストラクタです。

構文
```
public  XMLComment()
```

コメント
このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。

メソッド

addText(String)

説明
コメント・テキストを追加します。

構文
```
public XMLNode addText(java.lang.String str)
```

パラメータ
str ― コメント・テキスト

getNodeName()

説明
ノードの名前を取得します。

構文
```
public java.lang.String getNodeName()
```

戻り値
ノードの名前

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

readExternal(ObjectInput)

説明

このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

構文

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

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ

`inArg` − 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外

`IOException` − 入力ストリームの読込み時にエラーがあった場合に発生します。

`ClassNotFoundException` − クラスが検出されない場合に発生します。

reportSAXEvents(ContentHandler)

説明

DOM ツリーから SAX イベントをレポートします。

構文

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

パラメータ

`cntHandler` − ContentHandler

例外

`SAXException` − SAX コールバック関数によって発生します。

writeExternal(ObjectOutput)

説明

このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文

```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

パラメータ

`outArg` ― 圧縮ストリームの書込みに使用する ObjectOutput ストリーム

例外

`IOException` ― 圧縮ストリームの書込み時に例外があった場合に発生します。

XMLDeclPI

説明
このクラスは、XML 宣言の処理命令を実装します。

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

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

実装されるインタフェース
```
java.io.Externalizable, java.io.Serializable
```

関連項目
```
ProcessingInstruction
```

コンストラクタ

XMLDeclPI()

説明
デフォルトのコンストラクタです。

構文
```
public  XMLDeclPI()
```

XMLDeclPI(String, String, String, boolean)

説明

構文
```
public  XMLDeclPI(java.lang.String version, java.lang.String encoding,
java.lang.String standalone, boolean textDecl)
```

メソッド

cloneNode(boolean)

説明
このノードの複製を戻します。つまり、汎用コピーとして機能します。

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

戻り値
複製ノード

getData()

説明
完全に構成された文字列 'version=1.0 ....' を戻します。

構文
```
public java.lang.String getData()
```

戻り値
ノードの値

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

getEncoding()

説明
文字エンコーディング情報を取り出します。

構文
```
public final java.lang.String getEncoding()
```

戻り値
<?xml ...?> タグ内に格納されているエンコーディング情報、または最近設定された場合は、そのユーザー定義の出力エンコーディング

getNodeValue()

説明
このノードの値を取得します。

構文
```
public java.lang.String getNodeValue()
```

戻り値
ノードの値

例外
DOMException − DOMSTRING_SIZE_ERR: 実装における DOMString 変数に収まらない文字が戻された場合に発生します。

getStandalone()

説明
スタンドアロン情報を取り出します。

構文
```
public final java.lang.String getStandalone()
```

戻り値
<?xml ...?> タグ内に格納されているスタンドアロン属性

getVersion()

説明
バージョン情報を取り出します。

構文
```
public final java.lang.String getVersion()
```

戻り値
<?xml ...?> タグ内に格納されているバージョン番号

readExternal(ObjectInput)

説明

このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

構文

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

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

オーバーライド

クラス XMLPI の XMLPI.readExternal(ObjectInput)

パラメータ

inArg ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外

IOException ― 入力ストリームの読込み時にエラーがあった場合に発生します。

ClassNotFoundException ― クラスが検出されない場合に発生します。

setEncoding(String)

説明

出力用の文字エンコーディングを設定します。

構文

```
public final void setEncoding(java.lang.String encoding)
```

コメント

<?xml ...?> タグ内に格納されるエンコーディングを結果的に設定しますが、ドキュメントを保存するまでは設定されません。ドキュメントがロードされるまでは、このメソッドをコールしないでください。

パラメータ

encoding ― 設定する文字エンコーディング

setStandalone(String)

説明

<?xml ...?> タグ内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(java.lang.String value)
```

パラメータ

value ― 属性値（'YES' または 'NO'）

setVersion(String)

説明

<?xml ...?> タグ内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(java.lang.String version)
```

パラメータ

version ― 設定するバージョン情報

writeExternal(ObjectOutput)

説明

このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文

```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

オーバーライド

クラス XMLPI の XMLPI.writeExternal(ObjectOutput)

パラメータ

`outArg` ― 圧縮ストリームの書込みに使用する ObjectOutput ストリーム

例外

`IOException` ― 圧縮ストリームの書込み時に例外があった場合に発生します。

XMLDocument

説明

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

構文

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

oracle.xml.parser.v2.XMLDocument

実装されるインタフェース

java.io.Externalizable, java.io.Serializable

コメント

XML 仕様に従って、ツリーのルートはコメントと処理命令との組合せで構成されますが、ルート要素は 1 つのみです。getDocumentElement は、ルート要素を検索するショート・カットとして提供されている便利なメソッドです。

コンストラクタ

XMLDocument()

説明

空のドキュメントを作成します。

構文

```
public  XMLDocument()
```

メソッド

addID(String, XMLElement)

説明
このドキュメントに関連付けられた ID 要素を追加します。

構文
```
public void addID(java.lang.String name, XMLElement e)
```

パラメータ
name ― 文字列 - ID の値

e ― ID に関連付けられた XMLElement

adoptNode(Node)

説明
ノードを別のドキュメントからこのドキュメントに適用します。

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

コメント
戻されるノードに親はありません（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ます。

パラメータ
srcNode ― 適用されるノード

戻り値
ドキュメントとの関連が更新されたノード

例外
DOMException ― NOT_SUPPORTED_ERR: 適用されるノードのタイプがサポートされていない場合に発生します。

適用対象
DOM レベル 2

appendChild(Node)

説明
新しいノードをドキュメントに追加します。

構文
```
public org.w3c.dom.Node appendChild(org.w3c.dom.Node elem)
```

パラメータ
elem − 追加される新しいノード

戻り値
ドキュメントに追加された後のノード

例外
DOMException − HIERARCHY_REQUEST_ERR: このノードが、elem ノードのタイプの子を許可しないタイプの場合に発生します。WRONG_DOCUMENT_ERR: elem がこのドキュメントとは異なるドキュメントから作成された場合に発生します。

cloneNode(boolean)

説明
このノードの複製を戻します。つまり、ノードの汎用コピー・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)
```

コメント
複製ノードには親はありません（parentNode は null を戻します）。Element をクローニングすると、属性とその値がすべて（デフォルトの属性を示すために XML プロセッサによって生成された属性とその値も含む）コピーされますが、このメソッドでは、ディープ・クローンでないかぎり、それに含まれるテキストをコピーしません。これは、テキストが子の Text ノードに含まれているためです。他のタイプのノードをクローニングすると、単純にこのノードのコピーが戻されます。

パラメータ

deep ― true の場合は、指定したノードの下位サブツリーが再帰的にクローニングされ、false の場合は、そのノード（Element の場合はその属性）のみクローニングされます。

戻り値

複製ノード

createAttribute(String)

説明

指定された名前の Attr を作成します。

構文

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

コメント

作成後、Attr インスタンスを、setAttribute メソッドを使用して Element に設定できます。

パラメータ

name ― 属性の名前

戻り値

新しい Attr オブジェクト

例外

DOMException ― INVALID_CHARACTER_ERR: 指定した名前に無効な文字が含まれている場合に発生します。

createAttributeNS(String, String)

説明

指定された修飾名および名前空間 URI で属性を作成します。

構文

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

パラメータ
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

戻り値
指定された修飾名および名前空間 URI で作成された要素ノード

例外
DOMException − INVALID_CHARACTER_ERR: 指定した修飾名に無効な文字が含まれている場合に発生します。NAMESPACE_ERR: 修飾名の形式が誤っている場合、修飾名に接頭辞があり名前空間 URI が null または空の文字列の場合、または qualifiedName に接頭辞 "xml" があり名前空間 URI が "http://www.w3.org/2000/xmlns/" でない場合に発生します。

適用対象
DOM レベル 2

createCDATASection(String)

説明
値が指定された文字列の CDATASection ノードを作成します。

構文
```
public org.w3c.dom.CDATASection createCDATASection(java.lang.String data)
```

パラメータ
data − CDATASection コンテンツ用のデータ

戻り値
新しい CDATASection オブジェクト

例外
DOMException − DOMException が発生する場合があります。

createComment(String)

説明
指定された文字列を持つ Comment ノードを作成します。

構文
```
public org.w3c.dom.Comment createComment(java.lang.String data)
```

パラメータ
data ― ノード用のデータ

戻り値
新しい Comment オブジェクト

createDocumentFragment()

説明
空の DocumentFragment オブジェクトを作成します。

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

戻り値
新しい DocumentFragment

createElement(String)

説明
指定されたタイプの要素を作成します。

構文
```
public org.w3c.dom.Element createElement(java.lang.String tagName)
```

コメント
戻されるインスタンスは Element インタフェースを実装しているため、戻されたオブジェクトに属性を直接指定できることに注意してください。

パラメータ
tagName ― インスタンス化する要素タイプの名前。名前は大 / 小文字が区別されます。

戻り値
新しい Element オブジェクト

例外
DOMException − INVALID_CHARACTER_ERR: 指定した名前に無効な文字が含まれている場合に発生します。

createElementNS(String, String)

説明
指定された修飾名および名前空間 URI の要素を作成します。

構文
```
public org.w3c.dom.Element createElementNS(java.lang.String namespaceURI,
java.lang.String qualifiedName)
```

パラメータ
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

戻り値
指定された修飾名および名前空間 URI で作成された要素ノード

例外
DOMException − INVALID_CHARACTER_ERR: 指定した修飾名に無効な文字が含まれている場合に発生します。NAMESPACE_ERR: 修飾名の形式が誤っている場合、修飾名に接頭辞があり名前空間 URI が null または空の文字列の場合、または qualifiedName に接頭辞 "xml" があり名前空間 URI が "http://www.w3.org/XML/1998/namespace" でない場合に発生します。

適用対象
DOM レベル 2

createEntityReference(String)

説明
EntityReference オブジェクトを作成します。

構文
```
public org.w3c.dom.EntityReference createEntityReference(java.lang.String name)
```

パラメータ
name ― 参照するエンティティの名前

戻り値
新しい EntityReference オブジェクト

例外
DOMException ― INVALID_CHARACTER_ERR: 指定した名前に無効な文字が含まれている場合に発生します。

createEvent(String)

説明
指定されたタイプのイベント・オブジェクトを作成します。

構文
```
public org.w3c.dom.events.Event createEvent(java.lang.String type)
```

パラメータ
type ― イベントのタイプ

戻り値
指定したタイプのイベント・オブジェクト

createMutationEvent(String)

説明
指定したタイプの変更イベント・オブジェクトを作成します。

構文
```
public org.w3c.dom.events.MutationEvent createMutationEvent(java.lang.String type)
```

パラメータ
type ー 変更イベントのタイプ

戻り値
指定したタイプのイベント・オブジェクト

createNodeIterator(Node, int, NodeFilter, boolean)

説明
ルート・ノードを指定して、ノードイテレータを作成します。論理ビューに含める必要があるノードのタイプを制御するフラグ、ノードのフィルタ処理に使用するフィルタ、エンティティ参照とその子を含めることができるかどうかを判断するフラグを指定します。

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
root ー イテレータのルート・ノード

whatToShow ー イテレータ / ツリー・ウォーカーに含めるノードのタイプを示すフラグ

filter ー イテレータ / ツリー・ウォーカーから不要なノードをフィルタ処理するための NodeFilter

expandEntityReferences ー エンティティ参照の走査を示すフラグ

戻り値
このドキュメントで作成された NodeIterator インタフェースを実装するオブジェクト

例外
DOMException ー NOT_SUPPORTED_ERR: 指定したルートで NodeIterator が作成できない場合

createProcessingInstruction(String, String)

説明
指定された名前とデータ文字列の ProcessingInstruction ノードを作成します。

構文
```
public org.w3c.dom.ProcessingInstruction
```

```
createProcessingInstruction(java.lang.String target, java.lang.String data)
```

パラメータ

target ― 処理命令のターゲット部分

data ― ノード用のデータ

戻り値

新しい ProcessingInstruction オブジェクト

例外

DOMException ― INVALID_CHARACTER_ERR: 指定した名前に無効な文字が含まれている場合に発生します。

createRange()

説明

ドキュメントの始めに開始と終了の境界ポイントがある新しい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()
```

戻り値

ドキュメントの始めに開始と終了の境界ポイントがある新しいドキュメント範囲オブジェクト

createRangeEvent(String)

説明

指定したタイプの範囲イベント・オブジェクトを作成します。

構文

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

パラメータ

type ― イベントのタイプ

戻り値

指定したタイプのイベント・オブジェクト

createTextNode(String)

説明
指定された文字列の Text ノードを作成します。

構文
```
public org.w3c.dom.Text createTextNode(java.lang.String data)
```

パラメータ
data ― ノード用のデータ

戻り値
新しい Text オブジェクト

createTraversalEvent(String)

説明
指定したタイプの走査イベント・オブジェクトを作成します。

構文
```
public org.w3c.dom.events.Event createTraversalEvent(java.lang.String type)
```

パラメータ
type ― イベントのタイプ

戻り値
指定したタイプのイベント・オブジェクト

createTreeWalker(Node, int, NodeFilter, boolean)

説明

ルート・ノードを指定して、ノードイテレータを作成します。論理ビューに含める必要があるノードのタイプを制御するフラグ、ノードのフィルタ処理に使用するフィルタ、エンティティ参照とその子を含めることができるかどうかを判断するフラグを指定します。

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

`root` ― イテレータのルート・ノード

`whatToShow` ― イテレータ / ツリー・ウォーカーに含めるノードのタイプを示すフラグ

`filter` ― イテレータ / ツリー・ウォーカーから不要なノードをフィルタ処理するための NodeFilter

`expandEntityReference` ― エンティティ参照の走査を示すフラグ

戻り値

このドキュメントで作成された TreeWalker インタフェースを実装するオブジェクト

例外

`DOMException` ― NOT_SUPPORTED_ERR: 指定したルートで NodeIterator が作成できない場合

expectedElements(Element)

説明

要素に付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
e ー 要素

戻り値
名前のベクトル

getColumnNumber()

説明
列番号のデバッグ情報を取得します。

構文
```
public int getColumnNumber()
```

戻り値
列番号

getDebugMode()

説明
デバッグ・フラグを取得します。

構文
```
public boolean getDebugMode()
```

戻り値
ブール値のフラグ

getDoctype()

説明
このドキュメントに関連付けられているドキュメント・タイプ宣言（DTD）。DTD のない XML 文書の場合は null が戻されます。

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

コメント
DOM レベル 1 の仕様は DTD の編集をサポートしていないことに注意してください。

戻り値
関連付けられている DTD

関連項目
`org.w3c.dom.DocumentType`

getDocumentElement()

説明
ドキュメントのルート要素である子ノードに直接アクセスできる便利なメソッドです。

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

戻り値
ルート要素

getElementById(String)

説明
elementId で ID が指定された要素を戻します。該当する要素が存在しない場合は、null を戻します。該当の ID を持つ要素が複数ある場合の動作は定義されていません。

構文
```
public org.w3c.dom.Element getElementById(java.lang.String elementId)
```

パラメータ
`elementID` ― 一致する ID 要素の取得に使用する String

戻り値
一致する ID 要素が存在する場合はその ID 要素、存在しない場合は null

適用対象
DOM レベル 2

getElementsByTagName(String)

説明

指定されたタグ名を持つすべての Element を含んだ NodeList を、Document ツリーの先行順走査で検出された順序で戻します。

構文

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

パラメータ

tagname ― 照合するタグの名前。特殊値「*」はすべてのタグと一致します。

戻り値

一致した Element をすべて含んだ新しい NodeList オブジェクト

getElementsByTagNameNS(String, String)

説明

指定されたローカル名と名前空間 URI を持つ要素すべての NodeList を、Document ツリーの先行順走査で検出された順序で戻します。

構文

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

パラメータ

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

戻り値

一致する要素の Nodelist

適用対象

DOM レベル 2

getEncoding()

説明

文字エンコーディング情報を取り出します。

構文

```
public final java.lang.String getEncoding()
```

戻り値

<?xml ...?> タグ内に格納されているエンコーディング情報、または最近設定された場合は、そのユーザー定義の出力エンコーディング

getIDHashtable()

説明

XML DOM ツリー内の ID 要素のハッシュテーブルを取得します。

構文

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

戻り値

XMLDocument に関連付けられたハッシュテーブル

getImplementation()

説明

このドキュメントを処理する `DOMImplementation` オブジェクト。DOM アプリケーションは、複数の実装からオブジェクトを使用する場合があります。

構文

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

戻り値

関連付けられている DOM 実装

getLineNumber()

説明
行番号のデバッグ情報を取得します。

構文
```
public int getLineNumber()
```

戻り値
行番号

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

getOwnerDocument()

説明
このノードに関連付けられている Document オブジェクト。このノードが Document の場合は、null です。

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

戻り値
null

getStandalone()

説明
スタンドアロン情報を取り出します。

構文
```
public final java.lang.String getStandalone()
```

戻り値
<?xml ...?> タグ内に格納されているスタンドアロン属性

getSystemId()

説明
このノードを含むエンティティのシステム識別子を取得します。

構文
```
public java.lang.String getSystemId()
```

戻り値
システム識別子

getText()

説明
この要素によって挿入された、マークアップされていないテキストを戻します。

構文
```
public java.lang.String getText()
```

コメント
テキスト要素の場合は、ロー・データです。子ノードを持つ要素の場合、このメソッドはサブツリー全体を走査し、末端のテキスト要素ごとにテキストを追加して、サブツリーの XML マークアップを効率的に削除します。たとえば、XML 文書に "William Shakespeare" が含まれている場合、`XMLDocument.getText` は "William Shakespeare" を戻します。

戻り値
この要素に含まれる、マークアップされていないテキスト

getVersion()

説明
バージョン情報を取り出します。

構文
```
public final java.lang.String getVersion()
```

戻り値
<?xml ...?> タグ内に格納されているバージョン番号

importNode(Node, boolean)

説明
ノードを別のドキュメントからこのドキュメントにインポートします。

構文
```
public org.w3c.dom.Node importNode(org.w3c.dom.Node importedNode, boolean deep)
```

コメント
戻されるノードに親はありません（parentNode は null です）。ソース・ノードは、元のドキュメントから変更されたり削除されることはありません。このメソッドは、ソース・ノードの新しいコピーを作成します。すべてのノードについて、ノードをインポートすることで、インポート元のドキュメントが所有するノード・オブジェクトが作成されます。その属性値は、ソース・ノードの nodeName と nodeType、および名前空間に関連する属性（接頭辞、localName および namespaceURI）と同じです。ノード上で行う cloneNode 操作と同様に、ソース・ノードは変更されません。

パラメータ
`importedNode` − インポートされるノード。このノードの子がインポートされるかどうかを示すブール変数です。

`deep` − `true` の場合は、指定したノードの下位サブツリーが再帰的にクローニングされ、`false` の場合は、そのノード（`Element` の場合はその属性）のみがクローニングされます。

戻り値
現行ドキュメントには連結されているが、ドキュメント・ツリーには含まれない importedNode のコピー

例外

DOMException ― NOT_SUPPORTED_ERR: インポートされるノードのタイプがサポートされていない場合に発生します。

適用対象

DOM レベル 2

insertBefore(Node, Node)

説明

ノード newChild を、既存の子ノード refChild の前に挿入します。

構文

```
public org.w3c.dom.Node insertBefore(org.w3c.dom.Node newChild, org.w3c.dom.Node
refChild)
```

コメント

refChild が null の場合、newChild は子のリストの最後に挿入されます。newChild が DocumentFragment オブジェクトの場合、その子すべてが同じ順序で refChild の前に挿入されます。newChild がすでにツリーに存在している場合は、最初に削除されます。

パラメータ

newChild ― 挿入するノード

refChild ― 参照ノード（新規ノードを挿入する位置の後ろのノード）

戻り値

挿入されたノード

例外

DOMException ― 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の場合、または挿入するノードがこのノードにおける上位ノードの 1 つである場合に発生します。WRONG_DOCUMENT_ERR:newChild がこのノードを作成したドキュメントと異なるドキュメントから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の場合に発生します。NOT_FOUND_ERR:refChild がこのノードの子ではない場合に発生します。

print(OutputStream)

説明
このドキュメントの内容を、指定された出力ストリームに書き込みます。

構文
```
public void print(java.io.OutputStream out)
```

パラメータ
out ― 書込み先の OutputStream

例外
IOException ― エラーが発生した場合

print(OutputStream, String)

説明
このドキュメントの内容を、指定された出力ストリームに書き込みます。

構文
```
public void print(java.io.OutputStream out, java.lang.String enc)
```

パラメータ
out ― 書込み先の OutputStream

enc ― 出力に使用するエンコーディング

例外
IOException ― 無効なエンコーディングが指定された場合、またはその他のエラーが発生した場合

print(PrintDriver)

説明
このドキュメントの内容を、指定された出力ストリームに書き込みます。

構文
```
public void print(PrintDriver pd)
```

パラメータ
pd − 各ノードを書き込むために使用する PrintDriver

例外
IOException − エラーが発生した場合

print(PrintWriter)

説明
このドキュメントの内容を、指定された出力ストリームに書き込みます。

構文
```
public void print(java.io.PrintWriter out)
```

パラメータ
out − 書込み先の PrintWriter

例外
IOException − エラーが発生した場合

printExternalDTD(OutputStream)

説明
このドキュメントの内容を、指定された出力ストリームに書き込みます。

構文
```
public void printExternalDTD(java.io.OutputStream out)
```

パラメータ
out − 書込み先の OutputStream

例外
IOException − エラーが発生した場合

printExternalDTD(OutputStream, String)

説明

外部 DTD の内容を、指定された出力ストリームに書き込みます。

構文

```
public void printExternalDTD(java.io.OutputStream out, java.lang.String enc)
```

パラメータ

out ― 書込み先の OutputStream

enc ― 出力に使用するエンコーディング

例外

IOException ― 無効なエンコーディングが指定された場合、またはその他のエラーが発生した場合

printExternalDTD(PrintWriter)

説明

このドキュメントの内容を、指定された出力ストリームに書き込みます。

構文

```
public void printExternalDTD(java.io.PrintWriter out)
```

パラメータ

out ― 書込み先の PrintWriter

例外

IOException ― エラーが発生した場合

readExternal(ObjectInput)

説明

このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

構文

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

指定方法
インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

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

例外
IOException − 入力ストリームの読込み時にエラーがあった場合に発生します。

ClassNotFoundException − クラスが検出されない場合に発生します。

removeChild(Node)

説明
子ノードのドキュメント・リストから要素を削除します。

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

パラメータ
elem − 削除するノード

戻り値
ドキュメントから削除した後のノード

例外
DOMException − NO_MODIFICATION_ALLOWED_ERR: このドキュメントが読取り専用の場合に発生します。NOT_FOUND_ERR: oldChild がこのノードの子ではない場合に発生します。

replaceChild(Node, Node)

説明
子のリストにおいて、子ノード oldChild を newChild で置換し、oldChild ノードを戻します。

構文
```
public org.w3c.dom.Node replaceChild(org.w3c.dom.Node newChild, org.w3c.dom.Node
oldChild)
```

コメント
newChild がすでにツリーに存在している場合は、最初に削除されます。これは、XMLNode.removeChild メソッドのオーバーライドです。

パラメータ
newChild − 子リストに入れる新しいノード

oldChild − リスト内の置換対象のノード

戻り値
置換されたノード

例外
DOMException − 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の場合に発生します。WRONG_DOCUMENT_ERR:newChild がこのドキュメントとは異なるドキュメントから作成された場合に発生します。
NOT_FOUND_ERR: oldChild がこのノードの子ではない場合に発生します。

reportSAXEvents(ContentHandler)

説明
DOM ツリーから SAX イベントをレポートします。

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

パラメータ
ContentHandler − cntHandler

例外
SAXException − SAX コールバック関数によって発生します。

setDoctype(String, String, String)

説明
ドキュメントの doctype URI を設定します。

構文
```
public void setDoctype(java.lang.String rootname, java.lang.String sysid,
java.lang.String pubid)
```

パラメータ

root ― ルート要素の名前

sysid ― doctype のシステム識別子

pubid ― doctype の公開識別子（null も可能です）

setEncoding(String)

説明

出力用の文字エンコーディングを設定します。<?xml ...?> タグ内に格納されるエンコーディングを結果的に設定しますが、ドキュメントを保存するまでは設定されません。

構文

```
public final void setEncoding(java.lang.String encoding)
```

コメント

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。

パラメータ

encoding ― 設定する文字エンコーディング

setLocale(Locale)

説明

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

locale ― エラー・レポート用のロケール

setNodeContext(NodeContext)

説明

ノードのコンテキストを設定します。

構文

```
public void setNodeContext(oracle.xml.util.NodeContext nctx)
```

setParsedDoctype(String, String, String)

説明

sysid を解析して doctype オブジェクトを設定します。

構文

```
public void setParsedDoctype(java.lang.String rootname, java.lang.String sysid,
java.lang.String pubid)
```

パラメータ

rootmane ― ルート要素の名前

sysid ― doctype のシステム識別子

pubid ― doctype の公開識別子（null も可能です）

setStandalone(String)

説明

<?xml ...?> タグ内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(java.lang.String value)
```

パラメータ

value ― 属性値（'YES' または 'NO'）

setVersion(String)

説明

<?xml ...?> タグ内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(java.lang.String version)
```

パラメータ

version ― 設定するバージョン情報

validateElementContent(Element)

説明
要素ノードの内容の妥当性をチェックします。

構文
```
public boolean validateElementContent(org.w3c.dom.Element e)
```

パラメータ
e ― 妥当性チェックする要素

戻り値
有効な場合は true、それ以外は false

writeExternal(ObjectOutput)

説明
このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文
```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法
インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

パラメータ
outArg ― シリアル化または圧縮されたストリームの書込みに使用する ObjectOutput ストリーム

例外
IOException ― シリアル化または圧縮されたストリームの書込み時に例外があった場合に発生します。

XMLDocumentFragment

説明
このクラスは、DOM の DocumentFragment インタフェースを実装します。

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

oracle.xml.parser.v2.XMLDocumentFragment

実装されるインタフェース
```
java.io.Serializable
```

コメント
XMLNode ではなく XMLElement の拡張です。したがって、要素として処理できるため便利です。

関連項目
```
DocumentFragment、NodeFactory、DOMParser.setNodeFactory(NodeFactory)
```

コンストラクタ

XMLDocumentFragment()

説明
空のドキュメント・フラグメントを作成します。

構文
```
public  XMLDocumentFragment()
```

コメント
使用できません。XMLDocument の createElement(String) メソッドを使用してください。

このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。通常の XMLElement を作成する場合は、XMLDocument の createElement(String) を使用してください。

メソッド

getAttributes()

説明
XMLDocumentFragment の属性を取得します。

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

戻り値
空の NamedNodeMap

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

getParentNode()

説明
このノードの親を取得します。

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

戻り値
このノードの親（常に null）

XMLDOMException

説明
このクラスを使用して、DOM 例外をスローします。

構文
```
public class XMLDOMException
```

oracle.xml.parser.v2.XMLDOMException

コンストラクタ

XMLDOMException(short)

説明
指定したコードを使用して XMLDOMException 例外を構成します。

構文
```
public  XMLDOMException(short code)
```

パラメータ
code − DOM インタフェースで示されるコード。

XMLDOMException(short, String)

説明
指定したメッセージとコードを使用して XMLDOMException 例外を構成します。

構文
```
public  XMLDOMException(short code, java.lang.String s)
```

パラメータ
code − DOM インタフェースで示されるコード。デフォルト・メッセージを使用します。

s − エラー・メッセージ

XMLDOMImplementation

説明

このクラスは、DOMImplementation を実装します。

構文

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

oracle.xml.parser.v2.XMLDOMImplementation

実装されるインタフェース

java.io.Serializable

コンストラクタ

XMLDOMImplementation()

説明

XML DOM Implementation の新しいインスタンスを作成します。

構文

```
public  XMLDOMImplementation()
```

メソッド

createDocument(String, String, DocumentType)

説明

指定した DocumentType ノード、指定した名前のルート要素、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。

構文

```
public org.w3c.dom.Document createDocument(java.lang.String namespaceURI,
java.lang.String qualifiedName, org.w3c.dom.DocumentType doctype)
```

パラメータ

namespaceURI − ドキュメント内のルート要素の名前空間

qualifiedName − ドキュメント内のルート要素の修飾名

doctype − ドキュメントに関連付けられた DocumentType（DTD）

戻り値

作成された Document オブジェクト

例外

INVALID_CHARACTER_ERR: 指定した修飾名に無効な文字が含まれている場合に発生します。NAMESPACE_ERR: qualifiedName の形式が誤っている場合、qualifiedName に接頭辞があり namespaceURI が null または空の文字列の場合、または qualifiedName に接頭辞 "xml" があり namespaceURI が "http://www.w3.org/XML/1998/namespace" でない場合に発生します。WRONG_DOCUMENT_ERR: doctype がすでに別のドキュメントで使用されている場合、または doctype が別の実装から作成されている場合に発生します。

createDocumentType(String, String, String)

説明

ルート要素名およびシステム / 公開識別子を持つ空の DocumentType ノードを作成します。

構文

```
public org.w3c.dom.DocumentType createDocumentType(java.lang.String qualifiedName,
java.lang.String publicId, java.lang.String systemId)
```

パラメータ

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

戻り値

作成された DocumentType オブジェクト

例外

DOMException − INVALID_CHARACTER_ERR: 指定した修飾名に無効な文字が含まれている場合に発生します。NAMESPACE_ERR: qualifiedName の形式が誤っている場合に発生します。

hasFeature(String, String)

説明

DOM 実装で特定の機能が実装されているかどうかをテストします。

構文

```
public boolean hasFeature(java.lang.String feature, java.lang.String version)
```

戻り値

機能が実装されている場合は true、それ以外の場合は false

setFeature(String)

説明

指定した機能を設定します。

構文

```
public void setFeature(java.lang.String feature)
```

パラメータ

`feature` − DOM の機能

例外

`DOMException` − 機能が設定できない場合

XMLElement

構文

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

oracle.xml.parser.v2.XMLElement

実装されるインタフェース

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

説明

このクラスは、DOM の Element インタフェースを実装します。

コンストラクタ

XMLElement()

説明

デフォルトのコンストラクタです。

構文

```
public  XMLElement()
```

コメント

このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。通常の XMLElement を作成する場合は、XMLDocument の createElement(String) を使用してください。

メソッド

cloneNode(boolean)

説明

このノードの複製を戻します。つまり、ノードの汎用コピー・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)
```

コメント

複製ノードには親はありません（parentNode は null を戻します）。Element をクローニングすると、属性とその値がすべて（デフォルトの属性を示すために XML プロセッサによって生成された属性とその値も含む）コピーされますが、このメソッドでは、ディープ・クローンでないかぎり、それに含まれるテキストをコピーしません。これは、テキストが子の Text ノードに含まれているためです。他のタイプのノードをクローニングすると、単純にこのノードのコピーが戻されます。

パラメータ

deep ― true の場合は、指定したノードの下位サブツリーが再帰的にクローニングされ、false の場合は、そのノード（Element の場合はその属性）のみがクローニングされます。

戻り値

複製ノード

getAttribute(String)

説明

属性値を名前で取り出します。

構文

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

パラメータ

name ― 取り出す属性の名前

戻り値

Attr 値（文字列）、またはその属性に値が指定されていないかデフォルト値の場合は空の文字列

getAttributeNode(String)

説明

Attr ノードを名前を指定して取り出します。

構文

```
public org.w3c.dom.Attr getAttributeNode(java.lang.String name)
```

パラメータ

name ー 取り出す属性の名前

戻り値

指定された属性名を持つ Attr ノード、または該当する属性がない場合は null

getAttributeNodeNS(String, String)

説明

ローカル名と名前空間 URI を使用して Attr ノードを取り出します。

構文

```
public org.w3c.dom.Attr getAttributeNodeNS(java.lang.String namespaceURI,
java.lang.String localName)
```

パラメータ

namespaceURI ー名前空間 URI

localName ー取り出す属性の名前

戻り値

指定した namespaceURI および localName（存在する場合）を持つ属性、存在しない場合は null

適用対象

DOM レベル 2

getAttributeNS(String, String)

説明

ローカル名と名前空間 URI を使用して属性値を取り出します。

構文

```
public java.lang.String getAttributeNS(java.lang.String namespaceURI,
java.lang.String localName)
```

パラメータ

`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

戻り値

前述の namespaceURI および localName（存在する場合）を持つ属性の値、存在しない場合は null

適用対象

DOM レベル 2

getAttributes()

説明

このノードの属性を含んだ `NamedNodeMap` を取得します（`Element` の場合）、それ以外の場合は `null`。

構文

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

戻り値

この要素の属性リスト

getChildrenByTagName(String)

説明

指定されたタグ名を持つ直近の子すべての `NodeList` を戻します。

構文

```
public org.w3c.dom.NodeList getChildrenByTagName(java.lang.String name)
```

パラメータ
name ー 照合するタグの名前

戻り値
一致する子のリスト

getChildrenByTagName(String, String)

説明
指定されたタグ名と名前空間を持つ直近の子すべての NodeList を戻します。

構文
```
public org.w3c.dom.NodeList getChildrenByTagName(java.lang.String name,
java.lang.String ns)
```

パラメータ
name ー 照合するタグの名前 （ローカル名を指定してください）

ns ー 名前空間

戻り値
一致する子のリスト

getElementsByTagName(String)

説明
指定されたタグ名を持つすべての Element を含んだ NodeList を、Document ツリーの先行順走査で検出された順序で戻します。

構文
```
public org.w3c.dom.NodeList getElementsByTagName(java.lang.String tagname)
```

パラメータ
tagname ー 照合するタグの名前。特殊値「*」はすべてのタグと一致します。

戻り値
一致した Element をすべて含んだ新しい NodeList オブジェクト

getElementsByTagNameNS(String, String)

説明

指定されたローカル名と名前空間 URI を持つ子要素すべての NodeList を、この Element ツリーの先行順走査で検出された順序で戻します。

構文

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

パラメータ

`namespaceURI` ― 要素の名前空間

`localName` ― 要素のローカル名

適用対象

DOM レベル 2

getExpandedName()

説明

この要素の解決済み完全名を取得します。

構文

```
public java.lang.String getExpandedName()
```

指定方法

インタフェース oracle.xml.util.NSName 内の oracle.xml.util.NSName.getExpandedName()

戻り値

解決済み完全名

getFirstAttribute()

説明
最初の Attr を取り出します。

構文
public XMLNode getFirstAttribute()

戻り値
最初の Attr ノード、または属性がない場合は null

getLocalName()

説明
この要素のローカル名を取得します。

構文
```
public java.lang.String getLocalName()
```

指定方法
インタフェース oracle.xml.util.NSName 内の oracle.xml.util.NSName.getLocalName()

戻り値
ローカル名

getNamespaceURI()

説明
この要素の名前空間 URI を取得します。

構文
```
public java.lang.String getNamespaceURI()
```

戻り値
この要素の名前空間 URI

適用対象
DOM レベル 2

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

getPrefix()

説明
この要素の名前空間接頭辞を取得します。

構文
```
public java.lang.String getPrefix()
```

指定方法
インタフェース oracle.xml.util.NSName 内の oracle.xml.util.NSName.getPrefix()

戻り値
名前空間接頭辞

getQualifiedName()

説明
この要素の修飾名を取得します。

構文
```
public java.lang.String getQualifiedName()
```

指定方法
インタフェース oracle.xml.util.NSName 内の oracle.xml.util.NSName.getQualifiedName()

戻り値
修飾名

getTagName()

説明
要素の名前を取得します。

構文
```
public java.lang.String getTagName()
```

コメント
たとえば、<elementExample id="demo"> ... </elementExample> と指定すると、tagName には値 "elementExample" が設定されます。これは、DOM のすべての操作と同様に、XML においては大 / 小文字が保存されることに注意してください。HTML の DOM は、ソースの HTML ドキュメントの文字に関係なく、大文字の正規形で HTML 要素の tagName を戻します。

戻り値
要素名

hasAttribute(String)

説明
指定した名前の属性がこの要素に指定されている場合、または属性がデフォルト値である場合は true を戻し、それ以外の場合は false を戻します。

構文
```
public boolean hasAttribute(java.lang.String name)
```

パラメータ
String ― 存在をチェックされる属性の名前

戻り値
指定した名前の属性が存在する場合は true、存在しない場合は null

適用対象
DOM レベル 2

hasAttributeNS(String, String)

説明

指定したローカル名と名前空間 URI を持つ属性がこの要素に指定されている場合、または属性がデフォルト値である場合は true を戻し、それ以外の場合は false を戻します。

構文

```
public boolean hasAttributeNS(java.lang.String namespaceURI, java.lang.String
localName)
```

パラメータ

`namespaceURI` ― 存在をチェックされる属性の名前空間

`localName` ― 存在をチェックされる属性のローカル名

戻り値

指定したローカル名と名前空間 URI を持つ属性がこの要素で指定されている場合、または属性がデフォルト値である場合は true、それ以外の場合は false

適用対象

DOM レベル 2

hasAttributes()

説明

このノード（ノードが要素の場合）に属性があるかどうかを戻します。

構文

```
public boolean hasAttributes()
```

戻り値

このノードに属性がある場合は `true`、それ以外の場合は `false`

適用対象

DOM レベル 2

normalize()

説明

この Element の下のすべてのサブツリーにあるすべての Text ノードを、標準フォームにします。標準フォームでは、Text ノードは 1 つのマークアップ（例 : タグ、コメント、処理命令、CDATA セクションおよびエンティティ参照）のみで区切られます（つまり、隣接する Text ノードはありません）。このメソッドを使用すると、ドキュメントの DOM ビューが、保存および再ロードされた場合と同じになることが保証され、特定のドキュメント・ツリー構造に依存する操作（XPointer 参照など）を使用するときに役立ちます。

構文

```
public void normalize()
```

コメント

DOM レベル 2 では使用できません。

関連項目

XMLNode.normalize()

readExternal(ObjectInput)

説明

このメソッドは、入力ストリームを読み込み、その入力ストリームの情報ごとに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。

構文

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

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ

inArg ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外

IOException ― 圧縮ストリームの読込み時に例外があった場合に発生します。

ClassNotFoundException ― クラスが検出されない場合に発生します。

removeAttribute(String)

説明

名前を指定して属性を削除します。削除する属性にデフォルト値が設定されている場合は、すぐに置換されます。

構文

```
public void removeAttribute(java.lang.String name)
```

パラメータ

name ― 削除する属性の名前

例外

DOMException ― NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用の場合に発生します。

removeAttributeNode(Attr)

説明

指定された属性を削除します。

構文

```
public org.w3c.dom.Attr removeAttributeNode(org.w3c.dom.Attr oldAttr)
```

パラメータ

oldAttr ― 属性リストから削除する Attr ノード。削除する Attr にデフォルト値が設定されている場合は、すぐに置換されます。

戻り値

削除された Attr ノード

例外

DOMException ― NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用の場合に発生します。NOT_FOUND_ERR: oldAttr が要素の属性ではない場合に発生します。

removeAttributeNS(String, String)

説明
ローカル名と名前空間 URI を指定して属性を削除します。

構文
```
public void removeAttributeNS(java.lang.String namespaceURI, java.lang.String
localName)
```

パラメータ
`namespaceURI` − 削除する属性の名前空間

`localName` − 削除する属性のローカル名

例外
`DOMEXception` − NO_MODIFICATIONS_ALLOWED_ERR: この要素が読取り専用の場合

適用対象
DOM レベル 2

reportSAXEvents(ContentHandler)

説明
DOM ツリーから SAX イベントをレポートします。

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

パラメータ
`cntHandler` − ContentHandler

例外
`SAXException` − SAX コールバック関数によって発生します。

resolveNamespacePrefix(String)

説明

名前空間接頭辞を指定して、この要素の有効範囲内にある名前空間定義を検索します。

構文

```
public java.lang.String resolveNamespacePrefix(java.lang.String prefix)
```

指定方法

インタフェース NSResolver 内の NSResolver.resolveNamespacePrefix(String)

パラメータ

prefix ― 解決される名前空間接頭辞。接頭辞がデフォルト値の場合はデフォルトの名前空間を戻します。

戻り値

解決済み名前空間（接頭辞を解決できなかった場合は null）

setAttribute(String, String)

説明

新しい属性を追加します。指定した名前の属性が要素内にすでに存在している場合は、その値が value パラメータの値に変更されます。

構文

```
public void setAttribute(java.lang.String name, java.lang.String value)
```

コメント

この値は単純文字列であるため、設定時には解析されません。このため、マークアップ（例 : エンティティ参照として認識される構文）はリテラル・テキストとして処理され、書込み時には実装によって適切にエスケープされる必要があります。エンティティ参照を含んだ属性値を割り当てるには、ユーザーは Attr ノードの他に Text ノードと EntityReference ノードを作成し、適切なサブツリーを構築し、setAttributeNode を使用してそれを属性の値として割り当てる必要があります。このメソッドでは名前空間が認識されないため、このメソッドを使用して新しい属性を追加しても名前空間表は更新されません。

パラメータ

name − 作成または変更する属性の名前

value − 設定する値（文字列フォーム）

例外

DOMException − INVALID_CHARACTER_ERR: 指定した名前に無効な文字が含まれている場合に発生します。NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用の場合に発生します。

setAttributeNode(Attr)

説明

新しい属性を追加します。その名前の属性が要素内にすでに存在している場合は、新しい属性で置換されます。

構文

```
public org.w3c.dom.Attr setAttributeNode(org.w3c.dom.Attr newAttr)
```

パラメータ

newAttr − 属性リストに追加する Attr ノード

戻り値

newAttr 属性で同じ名前の既存属性が置換されると、既存の Attr ノードが戻されます。それ以外の場合は null が戻されます。

例外

DOMException − WRONG_DOCUMENT_ERR:newAttr が要素を作成したドキュメントと異なるドキュメントから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の場合に発生します。INUSE_ATTRIBUTE_ERR: newAttr がすでに別の Element オブジェクトの属性である場合に発生します。DOM ユーザーは、Attr ノードを明示的に複製し、それを別の要素で再使用する必要があります。

setAttributeNodeNS(Attr)

説明

新しい属性を追加します。そのローカル名および名前空間 URI を持つ属性が要素内にすでに存在している場合は、新しい属性で置換されます。

構文

```
public org.w3c.dom.Attr setAttributeNodeNS(org.w3c.dom.Attr newAttr)
```

パラメータ

newAttr ― 追加されるノード

戻り値

追加された属性ノード

例外

DOMException ― WRONG_DOCUMENT_ERR: ドキュメントを作成したドキュメントと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_MODIFICATIONS_ALLOWED_ERR: この要素が読取り専用の場合に発生します。INUSE_ATTRIBUTE_ERR: newAttr がすでに別の Element オブジェクトの属性である場合に発生します。

適用対象

DOM レベル 2

setAttributeNS(String, String, String)

説明

新しい属性を追加します。同じローカル名と名前空間 URI を持つ属性が要素にすでに存在している場合、その接頭辞は qualifiedName 内の接頭辞に変更され、その値は value パラメータに変更されます。この値は単純文字列であるため、設定時には解析されません。このため、マークアップ（例 : エンティティ参照として認識される構文）はリテラル・テキストとして処理され、書込み時には実装によって適切にエスケープされる必要があります。

構文

```
public void setAttributeNS(java.lang.String namespaceURI, java.lang.String
qualifiedName, java.lang.String value)
```

パラメータ

namespaceURI ― 追加される属性の名前空間

qualifiedName ― 追加される属性のローカル名

value ― 追加される属性の値

例外

DOMException − INVALID_CHARACTER_ERR: 指定した修飾名に無効な文字が含まれている場合に発生します。NAMESPACE_ERR: 修飾名の形式が誤っている場合、修飾名に接頭辞があり名前空間 URI が null または空の文字列の場合、qualifiedName が "xmlns" で名前空間 URI が "http://www.w3.org/2000/xmlns/" でない場合、または qualifiedName に接頭辞 "xml" があり名前空間 URI が "http://www.w3.org/XML/1998/namespaces" でない場合に発生します。NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用の場合に発生します。

適用対象

DOM レベル 2

validateContent(DTD)

説明

要素ノードの内容の妥当性をチェックします。

構文

```
public boolean validateContent(DTD dtd)
```

パラメータ

dtd − 要素の妥当性チェックに使用される DTD オブジェクト

戻り値

有効な場合は true、それ以外は false

validateContent(XMLSchema)

説明

指定した XML Schema パラメータのスキーマ（妥当性チェックに使用されるスキーマ）と照合して、要素の内容の妥当性をチェックします。

構文

```
public boolean validateContent(oracle.xml.parser.schema.XMLSchema schema)
```

戻り値

有効な場合は true、それ以外は false

validateContent(XMLSchema, String)

説明

モード指定された XML Schema と照合して、要素の内容の妥当性をチェックします。

構文

```
public boolean validateContent(oracle.xml.parser.schema.XMLSchema schema,
java.lang.String mode)
```

パラメータ

schema ― 内容の妥当性チェックに使用されるスキーマ

mode ― 妥当性チェック・モード

戻り値

有効な場合は true、それ以外は false

writeExternal(ObjectOutput)

説明

このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文

```
public void writeExternal(java.io.ObjectOutput outArg)
```

パラメータ

outArg ― シリアル化または圧縮されたストリームの書込みに使用する ObjectOutput ストリーム

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

XMLEntity

説明

このクラスは、DOM の Entity インタフェースを実装し、XML の Document Type Definition（DTD）で定義されているように XML の内部または外部エンティティを表します。

構文

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

oracle.xml.parser.v2.XMLEntity

実装されるインタフェース

java.io.Externalizable, java.io.Serializable

コンストラクタ

XMLEntity()

説明

デフォルトのコンストラクタです。

構文

```
public  XMLEntity()
```

コメント

このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。

メソッド

cloneNode(boolean)

説明

このノードの複製を戻します。つまり、ノードの汎用コピー・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)
```

コメント

複製ノードには親はありません（parentNode は null を戻します）。Element をクローニングすると、属性とその値がすべて（デフォルトの属性を示すために XML プロセッサによって生成された属性とその値も含む）コピーされますが、このメソッドでは、ディープ・クローンでないかぎり、それに含まれるテキストをコピーしません。これは、テキストが子の Text ノードに含まれているためです。他のタイプのノードをクローニングすると、単純にこのノードのコピーが戻されます。

パラメータ

deep − true の場合は、指定したノードの下位サブツリーが再帰的にクローニングされ、false の場合は、そのノード（Element の場合はその属性）のみクローニングされます。

戻り値

複製ノード

getNodeType()

説明

実際のオブジェクトのタイプを示すコードを取得します。

構文

```
public short getNodeType()
```

戻り値

ノードのタイプ

getNodeValue()

説明
このノードの値を、そのタイプに基づいて取得します。

構文
```
public java.lang.String getNodeValue()
```

戻り値
このノードの値

例外
DOMException − NO_MODIFICATION_ALLOWED_ERR: ノードが読取り専用の場合に発生します。DOMSTRING_SIZE_ERR: 実装プラットフォームにおける DOMString 変数に収まらない文字が戻された場合に発生します。

getNotationName()

説明
エンティティが未解析の場合は、そのエンティティの表記法の名前を取得します。エンティティが解析済みの場合は、null が戻されます。

構文
```
public java.lang.String getNotationName()
```

戻り値
表記法の名前

getPublicId()

説明
エンティティに関連付けられている公開識別子を取得します（指定されている場合）。公開識別子が指定されていない場合は、null が戻されます。

構文
```
public java.lang.String getPublicId()
```

戻り値
公開識別子

getSystemId()

説明

エンティティに関連付けられているシステム識別子を取得します（指定されている場合）。システム識別子が指定されていない場合は、null が戻されます。

構文

```
public java.lang.String getSystemId()
```

戻り値

システム識別子

readExternal(ObjectInput)

説明

このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

構文

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

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ

inArg － 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外

IOException － 入力ストリームの読込み時にエラーがあった場合に発生します。

ClassNotFoundException － クラスが検出されない場合に発生します。

setNodeValue(String)

説明

エンティティの値を設定します。

構文

```
public void setNodeValue(java.lang.String arg)
```

パラメータ

arg ― エンティティの新しい値

writeExternal(ObjectOutput)

説明

このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文

```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

パラメータ

outArg ― シリアル化または圧縮されたストリームの書込みに使用する ObjectOutput ストリーム

例外

IOException ― シリアル化または圧縮されたストリームの書込み時に例外があった場合に発生します。

XMLEntityReference

説明

このクラスは、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
```

oracle.xml.parser.v2.XMLEntityReference

実装されるインタフェース

java.lang.Cloneable, java.io.Externalizable, java.io.Serializable

コンストラクタ

XMLEntityReference()

説明

デフォルトのコンストラクタです。

構文

```
public  XMLEntityReference()
```

コメント

このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。

メソッド

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

readExternal(ObjectInput)

説明
このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

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

指定方法
インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ
`inArg` ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外
`IOException` ― 入力ストリームの読込み時にエラーがあった場合に発生します。

`ClassNotFoundException` ― クラスが検出されない場合に発生します。

writeExternal(ObjectOutput)

説明

このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文

```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

パラメータ

`outArg` ― 圧縮ストリームの書込みに使用する ObjectOutput ストリーム

例外

`IOException` ― 圧縮ストリームの書込み時に例外があった場合に発生します。

XMLError

説明
このクラスは、エラー・メッセージおよびエラーが発生した行番号を保持します。

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

oracle.xml.util.XMLError
  |
  +--oracle.xml.parser.v2.XMLError
```

コンストラクタ

XMLError()

説明
デフォルトのコンストラクタです。

構文
```
public  XMLError()
```

メソッド

error(int, int, String, String, String, int, int, boolean)

説明
新規エラーをベクトルに追加します。

構文
```
public void error(int line, int col, java.lang.String pubId, java.lang.String sysId,
java.lang.String mesg, int id, int type, boolean stop)
```

パラメータ
line ― エラーが発生した行番号

col ― エラーが発生した列番号

pubId ― 公開識別子

sysId － システム識別子

mesg － エラー・メッセージ

id － エラー ID

type － エラー・タイプ

stop － 処理の停止が必要かどうかを示すブール値

例外

ParseException － 致命的エラーの場合に発生します。

flushErrorListener(DOMLocator)

説明

すべてのエラーをエラー・リスナーにフラッシュします。

構文

```
public void flushErrorListener(oracle.xml.parser.v2.DOMLocator locator)
```

パラメータ

locator － DOM のロケータ・オブジェクト

flushErrorListenerStream(DOMLocator)

説明

すべてのエラーをエラー・リスナーにフラッシュします。

構文

```
public void flushErrorListenerStream(oracle.xml.parser.v2.DOMLocator locator)
```

パラメータ

locator － DOM のロケータ・オブジェクト

flushErrors()

説明
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・ハンドラにフラッシュします。

構文
```
public void flushErrors()
```

例外
ParseException － 致命的エラーの場合に発生します。

getErrorHandler()

説明
登録エラー・ハンドラを戻します。

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

戻り値
ErrorHandler

getErrorListener()

説明
登録エラー・リスナーを戻します。

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

戻り値
ErrorListener

setErrorHandler(ErrorHandler)

説明

エラー・ハンドラを登録します。

構文

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

パラメータ

`err` − ErrorHandler

setErrorListener(ErrorListener)

説明

エラー・リスナーを登録します。

構文

```
public void setErrorListener(javax.xml.transform.ErrorListener el)
```

パラメータ

`el` − ErrorListener

XMLNode

説明

DOM の Node インタフェースを実装し、ドキュメント・オブジェクト・モデル全体のプライマリ・データ型として機能します。ドキュメント・ツリー内の単一ノードを表します。

構文

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

oracle.xml.parser.v2.XMLNode

ダイレクト・サブクラス

XMLNSNode

実装されるインタフェース

java.lang.Cloneable, java.io.Externalizable, java.io.Serializable

コメント

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機能として組み込まれています。特定の nodeType に対してこれらの属性（Element の nodeValue または Comment の attributes など）の明白なマッピングがない場合は、null が戻されます。導出されたクラスには、関連情報を取得および設定する、追加の便利な機能が含まれている場合があります。この DOM のノードは、XMLNSNode ではなく XMLNode の拡張で、DOM 仕様によって定義された固定の Nodename を持ちます。このクラスを拡張できるのは、子ノードを持つことができないノードのみです。

XMLNode のフィールド

表 11-17 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 java.lang.String Auto_Events	自動イベントを設定するフラグ
capturing	public static final java.lang.String capturing	イベント・ターゲットによって処理される前に、イベント・ターゲットの祖先のいずれかによって処理できます。

表 11-17 XMLNode のフィールド（続き）

フィールド	構文	説明
DOMAttrModified	public static final java.lang.String DOMAttrModified	ノードの属性が変更されたことを示します。
DOMCharacterDataModified	public static final java.lang.String DOMCharacterDataModified	ノード内の文字データが変更されたことを示します。
DOMNodeInserted	public static final java.lang.String DOMNodeInserted	ノードが別のノードの子として追加されたことを示します。
DOMNodeInsertedIntoDocument	public static final java.lang.String DOMNodeInsertedIntoDocument	ノードの直接挿入またはノードが含まれているサブツリーの挿入によって、ノードがドキュメントに挿入されたことを示します。
DOMNodeRemoved	public static final java.lang.String DOMNodeRemoved	ノードがその親ノードから削除されたことを示します。
DOMNodeRemovedFromDocument	public static final java.lang.String DOMNodeRemovedFromDocument	ノードを直接削除したか、ノードが含まれているサブツリーの削除によって、ノードがドキュメントから削除されたことを示します。
DOMSubtreeModified	public static final java.lang.String DOMSubtreeModified	ドキュメントへのすべての変更を通知するための汎用イベントです
ELEMENTDECL	public static final short ELEMENTDECL	要素宣言ノード
noncapturing	public static final java.lang.String noncapturing	イベント・ターゲットの祖先のいずれかによって処理される前に、イベント・ターゲットによって処理されます。
RANGE_DELETE_EVENT	public static final java.lang.String RANGE_DELETE_EVENT	範囲イベントを削除するフラグ
RANGE_DELETETEXT_EVENT	public static final java.lang.String RANGE_DELETETEXT_EVENT	範囲削除テキスト・イベントを設定するフラグ
RANGE_INSERT_EVENT	public static final java.lang.String RANGE_INSERT_EVENT	範囲イベントを設定するフラグ
RANGE_INSERTTEXT_EVENT	public static final java.lang.String RANGE_INSERTTEXT_EVENT	範囲挿入テキスト・イベントを設定するフラグ
RANGE_REPLACE_EVENT	public static final java.lang.String RANGE_REPLACE_EVENT	範囲イベントを置換するフラグ
RANGE_SETTEXT_EVENT	public static final java.lang.String RANGE_SETTEXT_EVENT	範囲テキスト・イベントを設定するフラグ

表 11-17 XMLNode のフィールド（続き）

フィールド	構文	説明
TRAVERSAL_DELETE_EVENT	public static final java.lang.String TRAVERSAL_DELETE_EVENT	走査削除イベントを設定するフラグ
TRAVERSAL_REPLACE_EVENT	public static final java.lang.String TRAVERSAL_REPLACE_EVENT	走査置換イベントを設定するフラグ
XMLDECL_NODE	public static final short XMLDECL_NODE	属性宣言ノード

メソッド

XMLNode()

説明

新しい XMLNode を構成します。

構文

```
protected  XMLNode()
```

addEventListener(String, EventListener, boolean)

説明

このメソッドによって、イベント・ターゲット（ノード）にイベント・リスナーを登録できます。

構文

```
public void addEventListener(java.lang.String type, org.w3c.dom.events.EventListener
listener, boolean useCapture)
```

パラメータ

type ― リスナーを登録するイベントのタイプ

listener ― リスナー・オブジェクト

useCapture ― リスナーで取得を開始するかどうかを示すフラグ

appendChild(Node)

説明

ノード newChild を、このノードにおける子のリストの最後に追加します。newChild がすでにツリーに存在している場合は、最初に削除されます。

構文

```
public org.w3c.dom.Node appendChild(org.w3c.dom.Node newChild)
```

パラメータ

newChild − 追加するノード。DocumentFragment オブジェクトの場合は、ドキュメント・フラグメントの内容全体がこのノードの子リストに移されます。

戻り値

追加されたノード

例外

DOMException − 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の場合、または追加するノードがこのノードにおける上位ノードの 1 つである場合に発生します。WRONG_DOCUMENT_ERR:newChild がこのノードを作成したドキュメントと異なるドキュメントから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の場合に発生します。

cloneNode(boolean)

説明

このノードの複製を戻します。つまり、ノードの汎用コピー・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)
```

コメント

複製ノードには親はありません（parentNode は null を戻します）。Element をクローニングすると、属性とその値がすべて（デフォルトの属性を示すために XML プロセッサによって生成された属性とその値も含む）コピーされますが、このメソッドでは、ディープ・クローンでないかぎり、それに含まれるテキストをコピーしません。これは、テキストが子の Text ノードに含まれているためです。他のタイプのノードをクローニングすると、単純にこのノードのコピーが戻されます。

パラメータ
deep − true の場合は、指定したノードの下位サブツリーが再帰的にクローニングされ、false の場合は、そのノード（Element の場合はその属性）のみクローニングされます。

戻り値
複製ノード

dispatchEvent(Event)

説明
このメソッドによって、イベントを実装イベント・モデルにディスパッチできます。

構文
```
public boolean dispatchEvent(org.w3c.dom.events.Event evt)
```

戻り値
preventDefault または stopPropogation がコールされたかどうかを示すブール値

例外
UNSPECIFIED_EVENT_TYPE: イベントを初期化することにより、dispatchEvent のコール前にイベントのタイプが指定されていない場合に発生します。

getAttributes()

説明
このノードの属性を含んだ NamedNodeMap（Element の場合）を取得します。それ以外の場合は null を戻します。

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

戻り値
このノードの属性

getChildNodes()

説明
このノードの子すべてを含んだ NodeList を取得します。子がない場合は、ノードを含まない NodeList が戻されます。

構文

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

コメント

戻される NodeList の内容は「最新」といえます。たとえば、ノード・オブジェクトの子に加えた変更は、NodeList アクセッサが戻すノードに即時に反映されます。つまり、ノードの内容の静的なスナップショットではありません。これは、getElementsByTagName メソッドが戻すものも含め、すべての NodeList に対して同様です。

戻り値

このノードの子

getColumnNumber()

説明

列番号のデバッグ情報を取得します。

構文

```
public int getColumnNumber()
```

戻り値

列番号

getDebugMode()

説明

デバッグ情報モードを取得します。

構文

```
public boolean getDebugMode()
```

戻り値

デバッグ・モードを示すフラグ

getFirstChild()

説明
このノードにおける最初の子を取得します。該当するノードがない場合は null が戻されます。

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

戻り値
このノードにおける最初の子

getLastChild()

説明
このノードにおける最後の子を取得します。該当するノードがない場合は null が戻されます。

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

戻り値
このノードにおける最後の子

getLineNumber()

説明
行番号のデバッグ情報を取得します。

構文
```
public int getLineNumber()
```

戻り値
行番号

getLocalName()

説明
名前空間を有効にするためにノード・タイプによってオーバーライドされるノードのローカル名を取得します。

構文
```
public java.lang.String getLocalName()
```

適用対象
DOM レベル 2

getNamespaceURI()

説明
名前空間を有効にするためにノード・タイプによってオーバーライドされるノードの名前空間 URI を取得します。

構文
```
public java.lang.String getNamespaceURI()
```

戻り値
名前空間

適用対象
DOM レベル 2

getNextSibling()

説明
このノードの 1 つ後のノードを取得します。該当するノードがない場合は `null` が戻されます。

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

戻り値
次のノード

getNodeName()

説明
ノードの名前を取得します。

構文
```
public java.lang.String getNodeName()
```

戻り値
ノードの名前

getNodeType()

説明
ノードのタイプを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

getNodeValue()

説明
このノードの値を、そのタイプに基づいて取得します。

構文
```
public java.lang.String getNodeValue()
```

戻り値
このノードの値

例外
DOMException − NO_MODIFICATION_ALLOWED_ERR: ノードが読取り専用の場合に発生します。DOMSTRING_SIZE_ERR: 実装プラットフォームにおける DOMString 変数に収まらない文字が戻された場合に発生します。

getOwnerDocument()

説明

このノードに関連付けられている Document オブジェクトを取得します。これは、新規ノードの作成に使用される Document オブジェクトでもあります。このノードが Document のときは、null が戻されます。

構文

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

戻り値

このノードに関連付けられているドキュメント

getParentNode()

説明

このノードの親を取得します。Document、DocumentFragment および Attr を除くすべてのノードに親が存在する可能性があります。ただし、ノードが作成されたばかりでまだ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()
```

戻り値

このノードの親

getPrefix()

説明

名前空間を有効にするためにノード・タイプによってオーバーライドされるノードの接頭辞を取得します。

構文

```
public java.lang.String getPrefix()
```

戻り値

接頭辞

適用対象

DOM レベル 2

getPreviousSibling()

説明
このノードの 1 つ前のノードを取得します。該当するノードがない場合は null が戻されます。

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

戻り値
先行ノード

getProperty(String)

説明
ノードのプロパティを取得します。

構文
```
public java.lang.Object getProperty(java.lang.String propName)
```

パラメータ
propName − プロパティの名前

戻り値
オブジェクトの propValue − プロパティの値

getSystemId()

説明
このノードを含むエンティティのシステム識別子を取得します。

構文
```
public java.lang.String getSystemId()
```

戻り値
システム識別子

getText()

説明

この要素によって挿入された、マークアップされていないテキストを戻します。テキスト要素の場合は、ロー・データです。子ノードを持つ要素の場合、このメソッドはサブツリー全体を走査し、末端のテキスト要素ごとにテキストを追加して、サブツリーの XML マークアップを効率的に削除します。

構文

```
public java.lang.String getText()
```

コメント

たとえば、XML 文書に "William Shakespeare" が含まれている場合、`XMLDocument.getText` は "William Shakespeare" を戻します。

戻り値

この要素に含まれる、マークアップされていないテキスト

hasAttributes()

説明

このノード（ノードが要素の場合）に属性があるかどうかを戻します。

構文

```
public boolean hasAttributes()
```

戻り値

このノードに属性がある場合は `true`、それ以外の場合は `false`

適用対象

DOM レベル 2

hasChildNodes()

説明
ノードに子があるかどうかを簡単に判断できる便利なメソッドです。

構文
```
public boolean hasChildNodes()
```

戻り値
ノードに子がある場合は true、ノードに子がない場合は false

insertBefore(Node, Node)

説明
ノード newChild を、既存の子ノード refChild の前に挿入します。

構文
```
public org.w3c.dom.Node insertBefore(org.w3c.dom.Node newChild, org.w3c.dom.Node
refChild)
```

コメント
refChild が null の場合、newChild は子のリストの最後に挿入されます。newChild が DocumentFragment オブジェクトの場合、その子すべてが同じ順序で refChild の前に挿入されます。newChild がすでにツリーに存在している場合は、最初に削除されます。

パラメータ
newChild − 挿入するノード

refChild − 参照ノード（新規ノードを挿入する位置の後ろのノード）

戻り値
挿入されたノード

例外
DOMException − 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の場合、または挿入するノードがこのノードにおける上位ノードの 1 つである場合に発生します。WRONG_DOCUMENT_ERR:newChild がこのノードを作成したドキュメントと異なるドキュメントから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の場合に発生します。
NOT_FOUND_ERR:refChild がこのノードの子ではない場合に発生します。

isNodeFlag(int)

説明
ノード・フラグ情報を戻します。

構文
```
public boolean isNodeFlag(int flag)
```

戻り値
フラグが設定されている場合は true

isSupported(String, String)

説明
DOM 実装で特定の機能が実装され、その機能がこのノードでサポートされているかどうかをテストします。

構文
```
public boolean isSupported(java.lang.String feature, java.lang.String version)
```

パラメータ
String ― 機能、文字列バージョン

戻り値
機能がサポートされている場合は true、サポートされていない場合は false

print(OutputStream)

説明
このノードの内容を、指定された出力ストリームに書き込みます。

構文
```
public void print(java.io.OutputStream out)
```

パラメータ
out ― 書込み先の OutputStream

例外
IOException ― エラーが発生した場合

print(OutputStream, String)

説明

このノードの内容を、指定された出力ストリームに書き込みます。

構文

```
public void print(java.io.OutputStream out, java.lang.String enc)
```

パラメータ

out ― 書込み先の OutputStream

enc ― 出力に使用するエンコーディング

例外

IOException ― 無効なエンコーディングが指定された場合、またはその他のエラーが発生した場合

print(PrintWriter)

説明

このノードの内容を、指定された出力ライターを使用して書き込みます。

構文

```
public void print(java.io.PrintWriter out)
```

パラメータ

out ― 使用する PrintWriter

例外

IOException ― エラーが発生した場合

readExternal(ObjectInput)

説明

このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

構文

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

指定方法
インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ
`inArg` ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外
`IOException` ― 入力ストリームの読込み時にエラーがあった場合に発生します。

`ClassNotFoundException` ― クラスが検出されない場合に発生します。

removeChild(Node)

説明
`oldChild` で指定された子ノードを子のリストから削除して、そのノードを戻します。

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

パラメータ
`oldChild` ― 削除するノード

戻り値
削除されたノード

例外
`DOMException` ― NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用の場合に発生します。NOT_FOUND_ERR: `oldChild` がこのノードの子ではない場合に発生します。

removeEventListener(String, EventListener, boolean)

説明

このメソッドによって、イベント・ターゲット（ノード）からイベント・リスナーを削除できます。

構文

```
public void removeEventListener(java.lang.String type,
org.w3c.dom.events.EventListener listener, boolean useCapture)
```

パラメータ

type － リスナーを登録するイベントのタイプ

listener － リスナー・オブジェクト

useCapture － リスナーで取得を開始するかどうかを示すフラグ

replaceChild(Node, Node)

説明

子のリストにおいて、子ノード oldChild を newChild で置換し、oldChild ノードを戻します。newChild がすでにツリーに存在している場合は、最初に削除されます。

構文

```
public org.w3c.dom.Node replaceChild(org.w3c.dom.Node newChild, org.w3c.dom.Node
oldChild)
```

パラメータ

newChild － 子リストに入れる新しいノード

oldChild － リスト内の置換対象のノード

戻り値

置換されたノード

例外

DOMException － 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の場合、または挿入するノードがこのノードにおける上位ノードの 1 つである場合に発生します。WRONG_DOCUMENT_ERR:newChild がこのノードを作成したドキュメントと異なるドキュメントから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の場合に発生します。
NOT_FOUND_ERR: oldChild がこのノードの子ではない場合に発生します。

reportSAXEvents(ContentHandler)

説明

DOM ツリーから SAX イベントをレポートします。

構文

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

パラメータ

`ContentHandler` ― cntHandler

例外

`SAXException` ― SAX コールバック関数によって発生します。

resetNodeFlag(int)

説明

ノード・フラグ情報をリセットします。

構文

```
public void resetNodeFlag(int flag)
```

パラメータ

`flag` ― ノード・フラグ

selectNodes(String)

説明

指定されたパターンと一致するノードをツリーから選択します。このメソッドでは、パターンに名前空間接頭辞が含まれていないことが前提です。

構文

```
public org.w3c.dom.NodeList selectNodes(java.lang.String pattern)
```

パラメータ

`pattern` ― 照合する XSL パターン

戻り値
一致するノードのリスト

例外
XSLException － 照合時にエラーがあった場合に発生します。

selectNodes(String, NSResolver)

説明
指定されたパターンと一致するノードをツリーから選択します。

構文
```
public org.w3c.dom.NodeList selectNodes(java.lang.String pattern, NSResolver nsr)
```

パラメータ
pattern － 照合する XSL パターン

nsr － 指定したパターンにおいて発生する名前空間接頭辞を解決する NSResolver

戻り値
一致するノードのリスト

例外
XSLException － 照合時にエラーがあった場合に発生します。

selectSingleNode(String)

説明
指定されたパターンと一致する最初のノードをツリーから選択します。

構文
```
public org.w3c.dom.Node selectSingleNode(java.lang.String pattern)
```

パラメータ
pattern － 照合する XSL パターン

戻り値
一致するノード

例外
XSLException ― 照合時にエラーがあった場合に発生します。

selectSingleNode(String, NSResolver)

説明
指定されたパターンと一致する最初のノードをツリーから選択します。

構文
```
public org.w3c.dom.Node selectSingleNode(java.lang.String pattern, NSResolver nsr)
```

パラメータ
pattern ― 照合する XSL パターン

nsr ― 指定したパターンにおいて発生する接頭辞を解決する NSResolver

戻り値
一致するノード

例外
XSLException ― 照合時にエラーがあった場合に発生します。

setDebugInfo(int, int, String)

説明
ノードにデバッグ情報を設定します。

構文
```
public void setDebugInfo(int line, int col, java.lang.String sysid)
```

パラメータ
line ― 行番号

col ― 列番号

sysid ― システム識別子

setNodeFlag(int)

説明
ノード・フラグ情報を設定します。

構文
```
public void setNodeFlag(int flag)
```

パラメータ
flag − ノード・フラグ

setNodeValue(String)

説明
このノードの値を、そのタイプに基づいて設定します。

構文
```
public void setNodeValue(java.lang.String nodeValue)
```

例外
DOMException − NO_MODIFICATION_ALLOWED_ERR: ノードが読取り専用の場合に発生します。DOMSTRING_SIZE_ERR: 実装プラットフォームにおける DOMString 変数に収まらない文字が戻された場合に発生します。

setPrefix(String)

説明
名前空間を有効にするためにノード・タイプによってオーバーライドされるノードの接頭辞を設定します。

構文
```
public void setPrefix(java.lang.String prefix)
```

パラメータ
prefix − 接頭辞を設定

例外
DOMException ― DOM 例外が発生した場合

適用対象
DOM レベル 2

setProperty(String, Object)

説明
ノードのプロパティを設定します。

構文
```
public void setProperty(java.lang.String propName, java.lang.Object propValue)
```

パラメータ
propName ― プロパティの名前

propValue ― プロパティの値

supports(String, String)

説明
DOM 実装で特定の機能が実装され、その機能がこのノードでサポートされているかどうかをテストします。

構文
```
public boolean supports(java.lang.String feature, java.lang.String version)
```

コメント
使用できません。かわりに isSupported を使用してください。

パラメータ
feature ― 機能

version ―バージョン

戻り値
機能がサポートされている場合は true、サポートされていない場合は false

適用対象
DOM レベル 2

transformNode(XSLStylesheet)

説明
指定されたスタイルシートを使用して、ツリー内のノードを変換します。

構文
```
public org.w3c.dom.DocumentFragment transformNode(XSLStylesheet xsl)
```

パラメータ
`xsl` ― 変換に使用される `XSLStylesheet`

戻り値
ドキュメント・フラグメント

例外
`XSLException` ― XSL 変換時にエラーがあった場合に発生します。

valueOf(String)

説明
パターンと一致する最初のノードの値をツリーから選択します。

構文
```
public java.lang.String valueOf(java.lang.String pattern)
```

パラメータ
`pattern` ― 照合する XSL パターン

戻り値
一致するノードの値

例外
`XSLException` ― 照合時にエラーがあった場合に発生します。

valueOf(String, NSResolver)

説明
パターンと一致する最初のノードの値をツリーから選択します。

構文
```
public java.lang.String valueOf(java.lang.String pattern, NSResolver nsr)
```

パラメータ
pattern − 照合する XSL パターン

nsr − 指定したパターンにおいて発生する接頭辞を解決する NSResolver

戻り値
一致するノードの値

例外
XSLException − 照合時にエラーがあった場合に発生します。

writeExternal(ObjectOutput)

説明
このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

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

指定方法
インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

パラメータ
out − シリアル化または圧縮されたストリームの書込みに使用する ObjectOutput ストリーム

例外
IOException − シリアル化または圧縮されたストリームの書込み時に例外があった場合に発生します。

XMLNotation

説明

このクラスは、DOM の Notation インタフェースを実装し、Document Type Definition（DTD）で宣言された表記法を表します。

構文

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

oracle.xml.parser.v2.XMLNotation

実装されるインタフェース

java.io.Externalizable, java.io.Serializable

コンストラクタ

XMLNotation()

説明

デフォルトのコンストラクタです。

構文

```
public  XMLNotation()
```

コメント

このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。通常の XMLElement を作成する場合は、XMLNotation(String) を使用してください。

メソッド

cloneNode(boolean)

説明

このノードの複製を戻します。つまり、ノードの汎用コピー・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)
```

コメント

複製ノードには親はありません（parentNode は null を戻します）。Element をクローニングすると、属性とその値がすべて（デフォルトの属性を示すために XML プロセッサによって生成された属性とその値も含む）コピーされますが、このメソッドでは、ディープ・クローンでないかぎり、それに含まれるテキストをコピーしません。これは、テキストが子の Text ノードに含まれているためです。他のタイプのノードをクローニングすると、単純にこのノードのコピーが戻されます。

パラメータ

deep ― true の場合は、指定したノードの下位サブツリーが再帰的にクローニングされ、false の場合は、そのノード（Element の場合はその属性）のみクローニングされます。

戻り値

複製ノード

getNodeName()

説明

表記法の名前を取得します。

構文

```
public java.lang.String getNodeName()
```

戻り値

ノードの名前

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

getPublicId()

説明
公開識別子を取得します。指定されていない場合は null が戻されます。

構文
```
public java.lang.String getPublicId()
```

戻り値
公開識別子

getSystemId()

説明
システム識別子を取得します。指定されていない場合は null が戻されます。

構文
```
public java.lang.String getSystemId()
```

戻り値
システム識別子

readExternal(ObjectInput)

説明

このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

構文

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

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ

inArg ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外

IOException ― 入力ストリームの読込み時にエラーがあった場合に発生します。

ClassNotFoundException ― クラスが検出されない場合に発生します。

setPublicId(String)

説明

公開識別子を設定します。

構文

```
public void setPublicId(java.lang.String pubid)
```

パラメータ

pubid ― 設定する公開識別子

setSystemId(String)

説明
システム識別子を設定します。

構文
```
public void setSystemId(java.lang.String url)
```

パラメータ
url ― 設定するシステム識別子

writeExternal(ObjectOutput)

説明
このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文
```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法
インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

パラメータ
outArg ― シリアル化または圧縮されたストリームの書込みに使用する ObjectOutput ストリーム

例外
IOException ― シリアル化または圧縮されたストリームの書込み時に例外があった場合に発生します。

XMLNSNode

構文

```
public class XMLNSNode extends oracle.xml.parser.v2.XMLNode

oracle.xml.parser.v2.XMLNode
  |
  +--oracle.xml.parser.v2.XMLNSNode
```

実装されるインタフェース

```
java.lang.Cloneable, java.io.Externalizable, java.io.Serializable
```

説明

XMLNode を拡張して、名前空間名と子に対するサポートを強化します。

コンストラクタ

XMLNSNode(String)

説明

名前を指定して新しい XMLNSNode を構成します。

構文

```
protected  XMLNSNode(java.lang.String tag)
```

パラメータ

tag − ノードの名前

メソッド

addText(char[], int, int)

説明

このノードにテキストを追加します。最後の子がテキスト・ノードの場合は、最後の子に文字列を追加します。

構文

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

パラメータ

ch ― 追加する文字配列

start ― 文字配列内の開始インデックス

length ― 追加される文字数

例外

XMLDOMException ― テキストをこのノードに追加できない場合

addText(String)

説明

このノードにテキストを追加します。最後の子がテキスト・ノードの場合は、最後の子に文字列を追加します。

構文

```
public XMLNode addText(java.lang.String str)
```

パラメータ

str ― 追加するテキスト

例外

XMLDOMException ― テキストをこのノードに追加できない場合

appendChild(Node)

説明

ノード newChild を、このノードにおける子のリストの最後に追加します。newChild がすでにツリーに存在している場合は、最初に削除されます。

構文

```
public org.w3c.dom.Node appendChild(org.w3c.dom.Node newChildArg)
```

オーバーライド

クラス XMLNode の XMLNode.appendChild(Node)

パラメータ

newChildArg − 追加するノード。DocumentFragment オブジェクトの場合は、ドキュメント・フラグメントの内容全体がノードの子リストに移されます。

戻り値

追加されたノード

例外

DOMException − HIERARCHY_REQUEST_ERR: ノードが、newChild ノードのタイプの子を許可しないタイプの場合、または追加するノードがノードにおける上位ノードの 1 つである場合に発生します。WRONG_DOCUMENT_ERR:newChild がノードを作成したドキュメントと異なるドキュメントから作成された場合に発生します。NO_MODIFICATION_ALLOWED_ERR: ノードが読取り専用の場合に発生します。

getChildNodes()

説明

このノードの子すべてを含んだ NodeList を取得します。子がない場合は、ノードを含まない NodeList が戻されます。戻される NodeList の内容は「最新」といえます。たとえば、ノード・オブジェクトの子に加えた変更は、NodeList アクセッサが戻すノードに即時に反映されます。つまり、ノードの内容の静的なスナップショットではありません。これは、getElementsByTagName メソッドが戻すものも含め、すべての NodeList に対して同様です。

構文

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

オーバーライド
クラス XMLNode の XMLNode.getChildNodes()

戻り値
このノードの子

getFirstChild()

説明
このノードにおける最初の子を取得します。該当するノードがない場合は null が戻されます。

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

オーバーライド
クラス XMLNode の XMLNode.getFirstChild()

戻り値
このノードにおける最初の子

getLastChild()

説明
このノードにおける最後の子を取得します。該当するノードがない場合は null が戻されます。

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

オーバーライド
クラス XMLNode の XMLNode.getLastChild()

戻り値
このノードにおける最後の子

getLocalName()

説明
名前空間を有効にするためにノード・タイプによってオーバーライドされるノードのローカル名を取得します。

構文
```
public java.lang.String getLocalName()
```

オーバーライド
クラス XMLNode の XMLNode.getLocalName()

戻り値
ノードのローカル名

適用対象
DOM レベル 2

getNamespaceURI()

説明
名前空間を有効にするためにノード・タイプによってオーバーライドされるノードの名前空間 URI を取得します。

構文
```
public java.lang.String getNamespaceURI()
```

オーバーライド
クラス XMLNode の XMLNode.getNamespaceURI()

戻り値
ノードの名前空間

適用対象
DOM レベル 2

getNodeName()

説明
このノードの名前を、そのタイプに基づいて取得します。

構文
```
public java.lang.String getNodeName()
```

オーバーライド
クラス XMLNode の XMLNode.getNodeName()

戻り値
このノードの名前

getPrefix()

説明
名前空間を有効にするためにノード・タイプによってオーバーライドされるノードの接頭辞を取得します。

構文
```
public java.lang.String getPrefix()
```

オーバーライド
クラス XMLNode の XMLNode.getPrefix()

戻り値
ノードの接頭辞

適用対象
DOM レベル 2

getText()Description

説明

この要素によって挿入された、マークアップされていないテキストを戻します。

構文

```
public java.lang.String getText()
```

コメント

テキスト要素の場合は、ロー・データです。子ノードを持つ要素の場合、このメソッドはサブツリー全体を走査し、末端のテキスト要素ごとにテキストを追加して、サブツリーの XML マークアップを効率的に削除します。たとえば、XML 文書に "William Shakespeare" が含まれている場合、`XMLDocument.getText` は "William Shakespeare" を戻します。

オーバーライド

クラス `XMLNode` の `XMLNode.getText()`

戻り値

この要素によって含まれた、マークアップされていないテキスト

hasChildNodes()

説明

ノードに子があるかどうかを簡単に判断できる便利なメソッドです。

構文

```
public boolean hasChildNodes()
```

オーバーライド

クラス `XMLNode` の `XMLNode.hasChildNodes()`

戻り値

ノードに子がある場合は `true`、ノードに子がない場合は `false`

insertBefore(Node, Node)

説明

ノード newChild を、既存の子ノード refChild の前に挿入します。

構文

```
public org.w3c.dom.Node insertBefore(org.w3c.dom.Node newChild, org.w3c.dom.Node
refChild)
```

コメント

refChild が null の場合、newChild は子のリストの最後に挿入されます。newChild が DocumentFragment オブジェクトの場合、その子すべてが同じ順序で refChild の前に挿入されます。newChild がすでにツリーに存在している場合は、最初に削除されます。

オーバーライド

クラス XMLNode の XMLNode.insertBefore(Node, Node)

パラメータ

newChild − 挿入するノード

refChild − 参照ノード（新規ノードを挿入する位置の後ろのノード）

戻り値

挿入されたノード

例外

DOMException − 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の場合、または挿入するノードがこのノードにおける上位ノードの 1 つである場合に発生します。WRONG_DOCUMENT_ERR:newChild がこのノードを作成したドキュメントと異なるドキュメントから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の場合に発生します。
NOT_FOUND_ERR:refChild がこのノードの子ではない場合に発生します。

normalize()

説明

このノードの下のすべてのサブツリーにあるすべての Text ノード（属性ノードを含む）を、標準フォームにします。標準フォームでは、Text ノードは 1 つの構造（例 : 要素、コメント、処理命令、CDATA セクションおよびエンティティ参照）のみで区切られます（つまり、隣接する Text ノードや空の Text ノードはありません）。

構文

```
public void normalize()
```

コメント

このメソッドを使用すると、ドキュメントの DOM ビューが、保存および再ロードされた場合と同じになることが保証され、特定のドキュメント・ツリー構造に依存する操作（XPointer 参照など）を使用するときに役立ちます。

オーバーライド

クラス `XMLNode` の `XMLNode.normalize()`

適用対象

DOM レベル 2

removeChild(Node)

説明

`oldChild` で指定された子ノードを子のリストから削除して、そのノードを戻します。

構文

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

オーバーライド

クラス `XMLNode` の `XMLNode.removeChild(Node)`

パラメータ

`oldChild` ― 削除するノード

戻り値
削除されたノード

例外
DOMException − NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用の場合に発生します。NOT_FOUND_ERR: oldChild がこのノードの子ではない場合に発生します。

replaceChild(Node, Node)

説明
子のリストにおいて、子ノード oldChild を newChild で置換し、oldChild ノードを戻します。newChild がすでにツリーに存在している場合は、最初に削除されます。

構文
```
public org.w3c.dom.Node replaceChild(org.w3c.dom.Node newChild, org.w3c.dom.Node
oldChild)
```

オーバーライド
クラス XMLNode の XMLNode.replaceChild(Node, Node)

パラメータ
newChild − 子リストに入れる新しいノード

oldChild − リスト内の置換対象のノード

戻り値
置換されたノード

例外
DOMException − 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の場合、または挿入するノードがこのノードにおける上位ノードの 1 つである場合に発生します。WRONG_DOCUMENT_ERR:newChild がこのノードを作成したドキュメントと異なるドキュメントから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の場合に発生します。
NOT_FOUND_ERR: oldChild がこのノードの子ではない場合に発生します。

setPrefix(String)

説明

名前空間を有効にするためにノード・タイプによってオーバーライドされるノードの接頭辞を設定します。

構文

```
public void setPrefix(java.lang.String prefix)
```

オーバーライド

クラス XMLNode の XMLNode.setPrefix(String)

パラメータ

prefix − ノードの接頭辞を設定

適用対象

DOM レベル 2

XMLOutputStream

説明

XMLOutputSteam は、出力ストリームを書き込み、XML エンコーディングを処理できます。

構文

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

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

XMLOutputStream のフィールド

表 11-18 XMLOutputStream のフィールド

フィールド	構文	説明
COMPACT	public static int COMPACT	追加のインデントおよび新しい行なし
DEFAULT	public static int DEFAULT	追加のインデントおよび新しい行なし
PRETTY	public static int PRETTY	可読性を高めるためにインデントおよび新しい行を追加します

コンストラクタ

XMLOutputStream(OutputStream)

説明

ASCII 出力ストリームを作成します。

構文

```
public  XMLOutputStream(java.io.OutputStream out)
```

パラメータ

out ― 出力ストリーム

XMLOutputStream(PrintWriter)

説明
PrintWriter から出力ストリームを作成します。

構文
```
public  XMLOutputStream(java.io.PrintWriter out)
```

パラメータ
out − PrintWriter のストリーム

メソッド

addIndent(int)

説明
出力のインデント・レベルを設定します。

構文
```
public void addIndent(int offset)
```

パラメータ
offset − インデント・レベル

close()

説明
出力ストリームをクローズします。

構文
```
public void close()
```

例外
IOException − 出力ストリームのクローズ中にエラーが発生した場合

flush()

説明
出力ストリームをフラッシュします。

構文
```
public void flush()
```

例外
IOException ― 出力ストリームのフラッシュ中にエラーが発生した場合

getOutputStyle()

説明
現行の出力スタイルを取得します。

構文
```
public int getOutputStyle()
```

setEncoding(String, boolean, boolean)

説明
出力文字エンコーディングを設定します。

構文
```
public void setEncoding(java.lang.String encoding, boolean lendian, boolean
byteOrderMark)
```

パラメータ
encoding ― ストリームのエンコーディング

lendian ― エンコーディングのタイプが little endian であるかどうかを示すフラグ

byteOrderMark ― バイト・オーダー・マークが設定されているかどうかを示すフラグ

例外
IOException ― エンコーディング・タイプの設定中にエラーが発生した場合

setOutputStyle(int)

説明
出力スタイルを設定します。

構文
```
public void setOutputStyle(int style)
```

パラメータ
style ― 出力スタイル

write(int)

説明
出力ストリームのタイプに応じて文字を出力します。

構文
```
public void write(int c)
```

パラメータ
c ― 書き込む文字

例外
IOException ― 文字の書込み中にエラーが発生した場合

writeChars(String)

説明
文字列を出力に書き込みます。

構文
```
public void writeChars(java.lang.String str)
```

パラメータ
str ― 出力ストリームに書き込む文字列

例外
IOException ― 文字列の書込み中にエラーが発生した場合

writeIndent()

説明
インデントを出力します。

構文
```
public void writeIndent()
```

例外
IOException ― 文字列の書込み中にエラーが発生した場合

writeNewLine()

説明
改行ライターです。

構文
```
public void writeNewLine()
```

例外
IOException ― 文字列の書込み中にエラーが発生した場合

writeQuotedString(String)

説明
文字列を引用符で囲んで書き込みます。

構文
```
public void writeQuotedString(java.lang.String str)
```

パラメータ
str ― 出力ストリームに書き込む文字列

例外
IOException ― 文字列の書込み中にエラーが発生した場合

XMLParseException

説明

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

XMLParseException のフィールド

表 11-19 XMLParseException のフィールド

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

コンストラクタ

XMLParseException(String, String, String, int, int, int)

説明

構文

```
public  XMLParseException(java.lang.String mesg, java.lang.String pubId,
java.lang.String sysId, int line, int col, int type)
```

メソッド

formatErrorMessage(int)

説明
指定されたインデックスにおけるエラー・メッセージをフォーマットします。

構文
```
public java.lang.String formatErrorMessage(int i)
```

戻り値
エラー・メッセージ

getColumnNumber(int)

説明
指定されたインデックスにおけるエラーの列番号を取得します。

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

戻り値
列番号

getException(int)

説明
指定されたインデックスにおけるエラーで発生した例外（存在する場合）を取得します。

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

戻り値
例外

getLineNumber(int)

説明
指定されたインデックスにおけるエラーの行番号を取得します。

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

戻り値
行番号

getMessage(int)

説明
指定されたインデックスにおけるエラー・メッセージを取得します。

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

戻り値
エラー・メッセージ

getMessageType(int)

説明
指定されたインデックスにおけるエラー・メッセージのタイプを取得します。

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

戻り値
エラー・メッセージのタイプ

getNumMessages()

説明
解析中に検出されたエラーまたは警告の合計数を戻します。

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

戻り値
エラーまたは警告の数

getPublicId(int)

説明
指定されたインデックスにおけるエラーが発生したときの入力の公開識別子を取得します。

構文
```
public java.lang.String getPublicId(int i)
```

戻り値
公開識別子

getSystemId(int)

説明
指定されたインデックスにおけるエラーが発生したときの入力のシステム識別子を取得します。

構文
```
public java.lang.String getSystemId(int i)
```

戻り値
システム識別子

XMLParser

説明

このクラスは、DOMParser クラスと SAXParser クラスのベース・クラスとして機能します。World Wide Web Consortium（W3C）の勧告に従って、XML 1.0 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（アプリケーションはそれぞれの要件に基づいて DOM パーサーまたは SAX パーサーを使用できます）。

構文

```
public abstract class XMLParser
```

oracle.xml.parser.v2.XMLParser

XMLParser のフィールド

表 11-20 XMLParser のフィールド

フィールド	構文	説明
BASE_URL	public static final java.lang.String BASE_URL	エンティティの解析で使用するベース URL。setBaseURL() のかわりに使用します。URL オブジェクトである必要があります。
DTD_OBJECT	public static final java.lang.String DTD_OBJECT	妥当性チェックで使用する DTD オブジェクト。XMLParser.setDoctype() のかわりに使用します。
SCHEMA_OBJECT	public static final java.lang.String SCHEMA_OBJECT	妥当性チェックで使用するスキーマ・オブジェクト。XMLParser.setXMLSchema() のかわりに使用します。
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 は取得されません。
USE_DTD_ONLY_FOR_VALIDATION	public static final java.lang.String USE_DTD_ONLY_FOR_VALIDATION	true の場合、DTD オブジェクトは妥当性チェックのみに使用され、パーサー・ドキュメントには追加されません（Boolean.TRUE または Boolean.FALSE）。このプロパティ / 属性では、setDoctype がコールされた場合のみ、固定した DTD を使用します。

メソッド

getAttribute(String)

説明
実際の実装の特定の属性を取り出すことができます。

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

パラメータ
name ― 属性の名前

戻り値
属性の値

例外
IllegalArgumentException ― 実際の実装で属性を認識できない場合に発生します。

getBaseURL()

説明
ベース URL を取得します。

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

戻り値
ベース URL

getEntityResolver()

説明
現行のエンティティ・リゾルバを戻します。

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

戻り値
現行のエンティティ・リゾルバ、またはエンティティ・リゾルバが登録されていない場合は null

適用対象
SAX 2.0

関連項目
`setEntityResolver(EntityResolver)`

getErrorHandler()

説明
現行のエラー・ハンドラを戻します。

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

戻り値
現行のエラー・ハンドラ、またはエラー・ハンドラが登録されていない場合は null

適用対象
SAX 2.0

関連項目
`setErrorHandler(ErrorHandler)`

getReleaseVersion()

説明
Oracle XML Parser のバージョン番号を戻します。

構文
```
public static java.lang.String getReleaseVersion()
```

戻り値
バージョン番号の文字列

getValidationModeValue()

説明

妥当性チェック・モードを戻します。

構文

```
public int getValidationModeValue()
```

戻り値

0 － XML Parser が NONVALIDATING の場合。

1 － XML Parser が PARTIAL_VALIDATION の場合。

2 － XML Parser が DTD_VALIDATION の場合。

3 － XML Parser が SCHEMA_VALIDATION の場合。

getXMLProperty(String)

説明

プロパティを取得します。プロパティが存在し、サポートされている場合はそのプロパティが戻されます。それ以外の場合は null が戻されます。

構文

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

パラメータ

name － プロパティの名前

戻り値

オブジェクト－ プロパティの値

isXMLPropertyReadOnly(String)

説明

プロパティが読取り専用かどうかをチェックします。プロパティがサポートされていない場合は true が戻されます。

構文

```
public boolean isXMLPropertyReadOnly(java.lang.String name)
```

パラメータ
name − プロパティの名前

戻り値
ブール値− 読取り専用の場合は true

isXMLPropertySupported(String)

説明
プロパティがサポートされているかどうかをチェックします。

構文
```
public boolean isXMLPropertySupported(java.lang.String name)
```

パラメータ
name − プロパティの名前

戻り値
ブール値− サポートされている場合は true

parse(InputSource)

説明
指定された入力ソースから XML を解析します。

構文
```
public void parse(org.xml.sax.InputSource in)
```

パラメータ
in − 解析する org.xml.sax.InputSource

例外
XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

parse(InputStream)

説明

指定された入力ストリームから XML を解析します。外部エンティティと DTD を解決するために、ベース URL を設定する必要があります。

構文

```
public final void parse(java.io.InputStream in)
```

パラメータ

in ― 解析する XML データを含んだ InputStream

例外

XMLParseException ― 構文エラーまたはその他のエラーが発生した場合。

SAXException ― SAX 例外。他の例外をラップしている場合もあります。

IOException ― I/O エラー。

関連項目

setBaseURL(URL)

parse(Reader)

説明

指定された入力ストリームから XML を解析します。外部エンティティと DTD を解決するために、ベース URL を設定する必要があります。

構文

```
public final void parse(java.io.Reader r)
```

パラメータ

r ― 解析する XML データを含んだ Reader

例外

XMLParseException ― 構文エラーまたはその他のエラーが発生した場合。

SAXException ― SAX 例外。他の例外をラップしている場合もあります。

IOException ― I/O エラー。

関連項目

setBaseURL(URL)

parse(String)

説明
指定された URL から XML を解析します。

構文
```
public void parse(java.lang.String in)
```

パラメータ
in − 解析する URL を含んだ String

例外
XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

parse(URL)

説明
指定された URL が指す XML 文書を解析し、対応する XML 文書階層を作成します。

構文
```
public final void parse(java.net.URL url)
```

パラメータ
url − 解析する XML 文書を指し示す URL

例外
XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

reset()

説明
パーサーの状態をリセットします。

構文
```
public void reset()
```

setAttribute(String, Object)

説明
実際の実装に特定の属性を設定できます。

構文
```
public void setAttribute(java.lang.String name, java.lang.Object value)
```

パラメータ
`name` ― 属性の名前

`value` ― 属性の値

例外
`IllegalArgumentException` ― 実際の実装で属性を認識できない場合に発生します。

setBaseURL(URL)

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

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

パラメータ
`url` ― ベース URL

setDoctype(DTD)

説明

DTD を設定します。

構文

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

パラメータ

`dtd` ― 解析時に設定および使用される `DTD`

setEntityResolver(EntityResolver)

説明

アプリケーションでエンティティ・リゾルバを登録します。

構文

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

コメント

アプリケーションによるエンティティ・リゾルバの登録がない場合、XMLReader は独自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中に新規または別のリゾルバを登録でき、SAX パーサーはその新規リゾルバの使用を即時に開始する必要があります。

パラメータ

`resolver` ― エンティティ・リゾルバ

例外

`java.lang.NullPointerException` ― リゾルバの引数が null の場合

関連項目

`getEntityResolver()`

setErrorHandler(ErrorHandler)

説明
アプリケーションでエラー・イベント・ハンドラを登録します。

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

コメント
アプリケーションによるエラー・ハンドラの登録がない場合、SAX パーサーでレポートされるすべてのエラー・イベントは無視されます。ただし、通常の処理は継続できません。予期しないバグを回避するために、すべての SAX アプリケーションでエラー・ハンドラを実装することをお薦めします。

アプリケーションは解析中に新規または別のハンドラを登録でき、SAX パーサーはその新規ハンドラの使用を即時に開始する必要があります。

パラメータ
`handler` ― エラー・ハンドラ

例外
`java.lang.NullPointerException` ― ハンドラの引数が null の場合

関連項目
`getErrorHandler()`

setLocale(Locale)

説明
アプリケーションは、このメソッドでエラー・レポート用のロケールを設定します。

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

パラメータ
`locale` ― 設定するロケール

例外

SAXException ― SAXException が発生する場合があります。

関連項目

org.xml.sax.Parser

setPreserveWhitespace(boolean)

説明

空白保持モードを設定します。

構文

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

パラメータ

flag ― 保持モード

setValidationMode(int)

説明

妥当性チェックのモードを設定します。

構文

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

コメント

このメソッドは、パーサーの妥当性チェック・モードを 4 つのタイプ（NONVALIDATING、PARTIAL_VALIDATION、DTD_VALIDATION および SCHEMA_VALIDATION）のいずれかに設定します。

パラメータ

valMode ― パーサーに設定する妥当性チェック・モードのタイプを決定します。

setXMLProperty(String, Object)

説明

プロパティを設定します。

構文

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

コメント

プロパティが正常に設定されている場合は、そのプロパティの値が戻されます。プロパティが読取り専用で、設定できない場合、またはプロパティがサポートされていない場合は、null が戻されます。

パラメータ

name ー プロパティの名前

value ー プロパティの値

戻り値

オブジェクトー 設定されたプロパティ

setXMLSchema(Object)

説明

インスタンス文書の妥当性チェック用に XMLSchema を設定します。

構文

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

パラメータ

schema ー XMLSchema オブジェクト

XMLPI

説明
このクラスは、DOM の Processing Instruction インタフェースを実装します。

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

oracle.xml.parser.v2.XMLPI

ダイレクト・サブクラス
XMLDeclPI

実装される全インタフェース
java.io.Externalizable, java.io.Serializable

関連項目
ProcessingInstruction、NodeFactory、DOMParser.setNodeFactory(NodeFactory)

コンストラクタ

XMLPI()

説明
デフォルトのコンストラクタです。

構文
```
public  XMLPI()
```

コメント
このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。

メソッド

addText(String)

説明
テキスト文字列をノードに追加します。

構文
```
public XMLNode addText(java.lang.String str)
```

パラメータ
str ー 追加されるテキスト文字列

戻り値
ノード

getNodeName()

説明
PI の名前を取得します。

構文
```
public java.lang.String getNodeName()
```

戻り値
ノードの名前

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

getTarget()

説明

この PI のターゲットを戻します。XML は、処理命令を開始するマークアップに続く最初のトークンとしてこれを定義します。

構文

```
public java.lang.String getTarget()
```

戻り値

PI のターゲット

readExternal(ObjectInput)

説明

このメソッドは、writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

構文

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

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ

`inArg` ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外

`IOException` ― 入力ストリームの読込み時にエラーがあった場合に発生します。

`ClassNotFoundException` ― クラスが検出されない場合に発生します。

reportSAXEvents(ContentHandler)

説明
DOM ツリーから SAX イベントをレポートします。

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

パラメータ
cntHandler − ContentHandler

例外
SAXException − SAX コールバック関数によって発生します。

writeExternal(ObjectOutput)

説明
このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、そのオブジェクトの状態を保存します。

構文
```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法
インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

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

例外
IOException − 圧縮ストリームの書込み時に例外があった場合に発生します。

XMLPrintDriver

説明

XMLPrintDriver は、PrintDriver インタフェースを実装します。

構文

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

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

実装されるインタフェース

PrintDriver

XMLPrintDriver のフィールド

表 11-21 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

コンストラクタ

XMLPrintDriver(OutputStream)

説明

出力ストリームから XMLPrintDriver のインスタンスを作成します。

構文

```
public  XMLPrintDriver(java.io.OutputStream os)
```

XMLPrintDriver(PrintWriter)

説明
出力ライターから XMLPrintDriver のインスタンスを作成します。

構文
```
public  XMLPrintDriver(java.io.PrintWriter pw)
```

メソッド

close()

説明
出力ストリームまたは出力ライターをクローズします。

構文
```
public void close()
```

指定方法
インタフェース PrintDriver 内の PrintDriver.close()

flush()

説明
出力ストリームまたは出力ライターをフラッシュします。

構文
```
public void flush()
```

指定方法
インタフェース PrintDriver 内の PrintDriver.flush()

printAttribute(XMLAttr)

説明
XMLAttr ノードを出力します。

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

指定方法
インタフェース PrintDriver 内の PrintDriver.printAttribute(XMLAttr)

パラメータ
attr − XMLAttr ノード

printAttributeNodes(XMLElement)

説明
XMLElement の属性ごとに出力メソッドをコールします。

構文
```
public final void printAttributeNodes(XMLElement elem)
```

指定方法
インタフェース PrintDriver 内の
PrintDriver.printAttributeNodes(XMLElement)

パラメータ
elem − 属性を出力する要素

printCDATASection(XMLCDATA)

説明
XMLCDATA ノードを出力します。

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

指定方法
インタフェース PrintDriver 内の PrintDriver.printCDATASection(XMLCDATA)

パラメータ
cdata − XMLCDATA ノード

printChildNodes(XMLNode)

説明
XMLNode の子ごとに出力メソッドをコールします。

構文
```
public final void printChildNodes(XMLNode node)
```

指定方法
インタフェース PrintDriver 内の PrintDriver.printChildNodes(XMLNode)

パラメータ
node ― 子を出力するノード

printComment(XMLComment)

説明
XMLComment ノードを出力します。

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

指定方法
インタフェース PrintDriver 内の PrintDriver.printComment(XMLComment)

パラメータ
comment ― コメント・ノード

printDoctype(DTD)

説明
DTD を出力します。

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

指定方法
インタフェース PrintDriver 内の PrintDriver.printDoctype(DTD)

パラメータ
dtd − 出力する DTD

printDocument(XMLDocument)

説明
XMLDocument を出力します。

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

指定方法
インタフェース PrintDriver 内の PrintDriver.printDocument(XMLDocument)

パラメータ
doc − 出力するドキュメント

printDocumentFragment(XMLDocumentFragment)

説明
空の XMLDocumentFragment オブジェクトを出力します。

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

指定方法
インタフェース PrintDriver 内の
PrintDriver.printDocumentFragment(XMLDocumentFragment)

パラメータ
dfrag − 出力するドキュメント・フラグメント

printElement(XMLElement)

説明
XMLElement を出力します。

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

指定方法
インタフェース PrintDriver 内の PrintDriver.printElement(XMLElement)

パラメータ
elem − 出力する要素

printEntityReference(XMLEntityReference)

説明
XMLEntityReference ノードを出力します。

構文
```
public void printEntityReference(XMLEntityReference en)
```

指定方法
インタフェース PrintDriver 内の
PrintDriver.printEntityReference(XMLEntityReference)

パラメータ
en − XMLEntityReference ノード

printProcessingInstruction(XMLPI)

説明
XMLPI ノードを出力します。

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

指定方法
インタフェース PrintDriver 内の
PrintDriver.printProcessingInstruction(XMLPI)

パラメータ
pi − XMLPI ノード

printTextNode(XMLText)

説明
XMLText ノードを出力します。

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

指定方法
インタフェース PrintDriver 内の PrintDriver.printTextNode(XMLText)

パラメータ
text − テキスト・ノード

setEncoding(String)

説明
出力ドライバのエンコーディングを設定します。

構文
```
public void setEncoding(java.lang.String enc)
```

指定方法
インタフェース PrintDriver 内の PrintDriver.setEncoding(String)

パラメータ
enc − 出力するドキュメントのエンコーディング

XMLRangeException

説明

このクラスは、RangeException をカスタマイズします。

構文

```
public class XMLRangeException
```

oracle.xml.parser.v2.XMLRangeException

コンストラクタ

XMLRangeException(short)

説明

XMLRangeException インスタンスを生成します。

構文

```
public  XMLRangeException(short code)
```

XMLText

説明
このクラスは、DOM の Text インタフェースを実装します。

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

oracle.xml.parser.v2.XMLText

実装されるインタフェース
java.io.Externalizable, java.io.Serializable

関連項目
Text、NodeFactory、DOMParser.setNodeFactory(NodeFactory)

コンストラクタ

XMLText()

説明
デフォルトのコンストラクタです。

構文
```
public  XMLText()
```

コメント
このコンストラクタは、DOM ノードの非シリアル化または解凍時にのみ使用することに注意してください。このノードを非シリアル化して、シリアル化または圧縮されたストリームから DOM ノードを構成するには、オブジェクトのハンドルを作成する必要があります。

メソッド

addText(char[], int, int)

説明

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

ch − 追加する文字配列

start − 開始インデックス

length − 文字配列の長さ

getData()

説明

このインタフェースを実装するノードの文字データです。

構文

```
public java.lang.String getData()
```

コメント

DOM 実装では、Text ノードに格納できるデータの量を任意に制限できません。ただし、実装の制限によって、ノードのデータ全体が単一の DOMString に収まらない場合があります。この場合、ユーザーは、substringData をコールして、データを適切なサイズのピースで取り出すことができます。

例外

DOMException − NO_MODIFICATION_ALLOWED_ERR: ノードが読取り専用の場合に発生します。

DOMException − DOMSTRING_SIZE_ERR: 実装プラットフォームにおける DOMString 変数に収まらない文字が戻された場合に発生します。

getNodeName()

説明
XMLText の名前を取得します。

構文
```
public java.lang.String getNodeName()
```

戻り値
ノードの名前

getNodeType()

説明
実際のオブジェクトのタイプを示すコードを取得します。

構文
```
public short getNodeType()
```

戻り値
ノードのタイプ

getNodeValue()

説明
このテキスト・ノードの値を取得します。

構文
```
public java.lang.String getNodeValue()
```

戻り値
ノードの文字列値

例外
DOMException ― 値の取得時にエラーが発生した場合

isWhiteSpaceNode()

説明
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()
```

戻り値
ブール値

readExternal(ObjectInput)

説明
writeExternal メソッドによって圧縮ストリームに書き込まれた情報を読み込み、それに対応するオブジェクトをリストアします。

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

コメント
このメソッドがコールされるのは、XMLText オブジェクトが独立ノードとして非シリアル化されるか読み込まれて、他の DOM ノードからコールされていない場合です。

指定方法
インタフェース java.io.Externalizable 内の
java.io.Externalizable.readExternal(java.io.ObjectInput)

パラメータ
`inArg` ― 圧縮ストリームの読込みに使用する ObjectInput ストリーム

例外
`IOException` ― 入力ストリームの読込み時にエラーがあった場合に発生します。

`ClassNotFoundException` ― クラスが検出されない場合に発生します。

reportSAXEvents(ContentHandler)

説明

DOM ツリーから SAX イベントをレポートします。

構文

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

パラメータ

`cntHandler` − ContentHandler

例外

`SAXException` − SAX コールバック関数によって発生します。

splitText(int)

説明

`Text` ノードを、指定されたオフセットで 2 つの Text ノードに分割します。2 つのノードは兄弟ノードとなり、元のノードにはオフセットまでの内容のみ含まれます。次の兄弟ノードとして挿入された新規ノードには、オフセット・ポイント以後の内容がすべて含まれます。

構文

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

パラメータ

`offset` − 分割する位置のオフセット（開始位置は 0）

戻り値

新しい `Text` ノード

例外

`DOMException` − INDEX_SIZE_ERR: 指定したオフセットが負数であるか、またはデータ内の文字数よりも大きい場合に発生します。NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用の場合に発生します。

writeExternal(ObjectOutput)

説明

このメソッドは、オブジェクトに関する情報を使用してバイナリの圧縮ストリームを作成することで、このオブジェクトの状態を保存します。

構文

```
public void writeExternal(java.io.ObjectOutput outArg)
```

指定方法

インタフェース java.io.Externalizable 内の
java.io.Externalizable.writeExternal(java.io.ObjectOutput)

パラメータ

`outArg` ― 圧縮ストリームを書込みに使用する ObjectOutput ストリーム

例外

`IOException` ― 圧縮ストリームの書込み時に例外があった場合に発生します。

XMLToken インタフェース

説明

XMLToken の基本インタフェースです。

構文

```
public interface XMLToken
```

コメント

トークン化機能を備えたすべての XMLParser アプリケーションがこのインタフェースを実装する必要があります。インタフェースは、`XMLParser` のメソッド `setTokenHandler(XMLToken handler)` を使用して登録する必要があります。

XMLToken handler != null の場合、登録済で検出された各トークンごとに、パーサーは XMLToken コールバック・メソッド `token(int token, String value)` をコールします。トークン化時に、パーサーはドキュメントの妥当性チェックを行わず、内部または外部エンティティの組込みまたは読込みは行いません。XMLToken handler == null の場合、パーサーは通常どおり解析します。

XML トークンに関する要求は、XMLParser のメソッド `setToken (int token, boolean set)` を使用して登録または登録解除されます。要求は解析時にも（コールバック・メソッド内部から）登録できます。

XML トークンは、`XMLToken` インタフェースにおけるパブリック定数として定義されます。これらは、W3C の XML 構文仕様の XML 構文変数に対応します。

XMLToken のフィールド

表 11-22 XMLToken のフィールド

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

表 11-22 XMLToken のフィールド（続き）

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

表 11-22 XMLToken のフィールド（続き）

フィールド	構文	説明
Reference	public static final int Reference	Reference ::= EntityRef | CharRef | PEReference EntityRef ::= '&' Name ';' PEReference ::= '%' Name ';' CharRef ::= '&#' [0-9]+ ';' | '&#x' [0-9a-fA-F]+ ';
STag	public static final int STag	STag ::= '<' STagName (S Attribute)* S? '>'
STagName	public static final int STagName	STagName ::= Name
TextDecl	public static final int TextDecl	TextDecl ::= '<' '?' 'xml' VersionInfo? EncodingDecl S? '?>'
XMLDecl	public static final int XMLDecl	XMLDecl ::= '<' '?' 'xml' VersionInfo EncodingDecl? SDDecl? S? '?' '>'

メソッド

token(int, String)

説明

インタフェース・コールバック・メソッド。XML トークンとそれに対応する値を受け取ります。

構文

```
public void token(int token, java.lang.String value)
```

パラメータ

`token` ― インタフェースに指定された XML トークン定数

`value` ― 解析済テキストの対応する部分文字列

XMLTokenizer

説明

このクラスは、World Wide Web Consortium（W3C）の勧告に従って、XML 1.0 パーサーを実装します。

構文

```
public class XMLTokenizer
```

oracle.xml.parser.v2.XMLTokenizer

コンストラクタ

XMLTokenizer()

説明

新しい Tokenizer オブジェクトを作成します。

構文

```
public  XMLTokenizer()
```

XMLTokenizer(XMLToken)

説明

新しい Tokenizer オブジェクトを作成します。

構文

```
public  XMLTokenizer(XMLToken handler)
```

メソッド

parseDocument()

説明
ドキュメントを解析します。

構文
```
public void parseDocument()
```

setErrorHandler(ErrorHandler)

説明
アプリケーションは、このメソッドで新しいエラー・イベント・ハンドラを登録できます。このメソッドは、エラー処理に関する以前の設定を置換します。

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

パラメータ
`handler` ― 登録する `ErrorHandler`

setErrorStream(OutputStream)

説明
エラー用の出力ストリームを登録します。

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

setToken(int, boolean)

説明

アプリケーションは、このメソッドで XML Tokenizer 用の新規トークンを登録できます。

構文

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

パラメータ

token － 設定する XMLToken

setTokenHandler(XMLToken)

説明

アプリケーションは、このメソッドで新しい XML Tokenizer イベント・ハンドラを登録できます。

構文

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

パラメータ

handler － 登録する XMLToken

tokenize(InputSource)

説明

指定された入力ソースから XML をトークン化します。

構文

```
public final void tokenize(org.xml.sax.InputSource in)
```

パラメータ

in － 解析する org.xml.sax.InputSource

例外

XMLParseException － 構文エラーまたはその他のエラーが発生した場合。

SAXException － SAX 例外。他の例外をラップしている場合もあります。

IOException － I/O エラー。

tokenize(InputStream)

説明
指定された入力ストリームから XML をトークン化します。

構文
```
public final void tokenize(java.io.InputStream in)
```

パラメータ
in − 解析する XML データを含んだ InputStream

例外
XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

関連項目
XMLParser.setBaseURL(URL)

tokenize(Reader)

説明
指定された入力ストリームから XML をトークン化します。

構文
```
public final void tokenize(java.io.Reader r)
```

パラメータ
r − 解析する XML データを含んだ Reader

例外
XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

関連項目
XMLParser.setBaseURL(URL)

tokenize(String)

説明

指定された URL から XML をトークン化します。

構文

```
public final void tokenize(java.lang.String in)
```

パラメータ

in − 解析する URL を含んだ String

例外

XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

tokenize(URL)

説明

指定された URL が指す XML 文書をトークン化し、対応する XML 文書階層を作成します。

構文

```
public final void tokenize(java.net.URL url)
```

パラメータ

url − 解析する XML 文書を指し示す URL

例外

XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

JXDocumentBuilder

説明

XML 文書から DOM 文書のインスタンスを取得する 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、XML から org.w3c.dom.Document を取得できます。

構文

```
public class JXDocumentBuilder
```

oracle.xml.jaxp.JXDocumentBuilder

コメント

このクラスのインスタンスは、DocumentBuilderFactory.newDocumentBuilder メソッドを使用して取得できます。このクラスのインスタンスを取得した後、様々な入力ソースから XML を解析できます。入力ソースには、InputStream、ファイル、URL および SAX InputSource があります。

このクラスでは、SAX API の複数のクラスが再利用されることに注意してください。このため、実際の DOM 実装の実装者は、XML 文書の Document への解析に SAX パーサーを使用する必要がありません。実装で必要なのは、これらの既存の API を使用してアプリケーションと通信を行うことのみです。

適用対象

JAXP 1.0

メソッド

getDOMImplementation()

説明

このドキュメントを処理する DOMImplementation オブジェクト。DOM アプリケーションは、複数の実装からオブジェクトを使用する場合があります。

構文

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

戻り値

関連付けられている DOM 実装

isNamespaceAware()

説明

このパーサーが名前空間を認識するように構成されているかどうかを示します。

構文

```
public boolean isNamespaceAware()
```

isValidating()

説明

このパーサーが XML 文書の妥当性をチェック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()

説明

DOM ツリーを作成するため、DOM 文書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(InputSource)

説明

指定した入力ソースの内容を XML 文書として解析し、新しい DOM 文書オブジェクトを戻します。

構文

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

パラメータ

`is` ― 解析される内容を含んだ InputSource

例外

IOException － I/O エラーが発生した場合

SAXException － 解析エラーが発生した場合

IllegalArgumentException － InputSource が null の場合

関連項目

org.xml.sax.DocumentHandler

setEntityResolver(EntityResolver)

説明

解析する XML 文書に存在するエンティティの解決に使用する EntityResolver を指定します。null に設定すると、実際の実装では、独自の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)
```

setErrorHandler(ErrorHandler)

説明

解析する XML 文書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。null に設定すると、実際の実装では、独自の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)
```

JXDocumentBuilderFactory

説明

アプリケーションが、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取得できる、ファクトリ API を定義します。

構文

```
public class JXDocumentBuilderFactory
```

oracle.xml.jaxp.JXDocumentBuilderFactory

適用対象

JAXP 1.0

JXDocumentBuilderFactory のフィールド

表 11-23 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・モードを Boolean.TRUE または Boolean.FALSE に設定します。
DTD_OBJECT	public static final java.lang.String DTD_OBJECT	妥当性チェックで使用する 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（ERROR_STREAM が設定されている場合のみ）を使用したエラー・レポートのエンコーディング。
ERROR_STREAM	public static final java.lang.String ERROR_STREAM	エラー・レポート用のエラー・ストリーム。オブジェクトは OutputStream または PrintWriter です。ErrorHandler が設定されている場合、この属性は無視されます。
NODE_FACTORY	public static final java.lang.String NODE_FACTORY	NodeFactory を設定してカスタム・ノードを作成します。
SCHEMA_ OBJECT	public static final java.lang.String SCHEMA_OBJECT	妥当性チェックで使用するスキーマ・オブジェクト。

表 11-23 JXDocumentBuilderFactory のフィールド（続き）

フィールド	構文	説明
SHOW_ WARNINGS	public static final java.lang.String SHOW_WARNINGS	警告を無視するかどうかを示すブール値。Boolean.TRUE または Boolean.FALSE に設定します。
USE_DTD_ONLY_ FOR_VALIDATION	public static final java.lang.String USE_DTD_ONLY_FOR_ VALIDATION	true の場合、DTD オブジェクトは妥当性チェックでのみ使用され、パーサー・ドキュメントには追加されません。

コンストラクタ

JXDocumentBuilderFactory()

説明

デフォルトのコンストラクタです。

構文

```
public  JXDocumentBuilderFactory()
```

メソッド

getAttribute(String)

説明

実際の実装の特定の属性を取り出すことができます。

構文

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

パラメータ

name ― 属性の名前

戻り値

属性の値

例外

IllegalArgumentException ― 実際の実装で属性を認識できない場合に発生します。

isExpandEntityReferences()

説明

エンティティ参照ノードを拡張するパーサーを作成するようにファクトリが構成されているかどうかを示します。常に true を戻します。現在、エンティティ参照の拡張を回避する方法はありません。

構文

```
public boolean isExpandEntityReferences()
```

戻り値

ブール値

isIgnoringComments()

説明

コメントを無視するパーサーを作成するようにファクトリが構成されているかどうかを示します。常に false を戻します。現在は、コメントを無視するように構成できません。

構文

```
public boolean isIgnoringComments()
```

戻り値

ブール値

isNamespaceAware()

説明

名前空間を認識するパーサーを作成するようにファクトリが構成されているかどうかを示します。常に true を戻します。現在、名前空間を無視する方法はありません。

構文

```
public boolean isNamespaceAware()
```

戻り値

名前空間を認識するかどうかのブール値

newDocumentBuilder()

説明

現在構成されているパラメータを使用して、DocumentBuilder の新しいインスタンスを作成します。

構文

```
public javax.xml.parsers.DocumentBuilder newDocumentBuilder()
```

例外

`ParserConfigurationException` ― 要求された構成を満たす DocumentBuilder を作成できない場合

setAttribute(String, Object)

説明

実際の実装に特定の属性を設定できます。

構文

```
public void setAttribute(java.lang.String name, java.lang.Object value)
```

パラメータ

`name` ― 属性の名前

`value` ― 属性の値

例外

`IllegalArgumentException` ― 実際の実装で属性を認識できない場合に発生します。

JXSAXParser

説明

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、このクラスで 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。

構文

```
public class JXSAXParser
```

oracle.xml.jaxp.JXSAXParser

コメント

移行を容易にするため、このクラスでは、新しいメソッドとともに、同じ名前とインタフェースを引き続きサポートします。このクラスのインスタンスは、SAXParserFactory.newSAXParser メソッドを使用して取得できます。このクラスのインスタンスを取得した後、様々な入力ソースから XML を解析できます。入力ソースには、InputStream、ファイル、URL および SAX InputSource があります。

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ます。プロパティが未定義の場合は、プラットフォームのデフォルトを使用します。

作成するファクトリ実装を制御するシステム・プロパティは、"javax.xml.style.TransformFactory" という名前です。このプロパティは、この抽象クラスの具体的なサブクラスであるクラスを指定します。プロパティが未定義の場合は、プラットフォームのデフォルトが使用されます。

実際のパーサーで内容を解析すると、指定した HandlerBase のメソッドがコールされます。

適用対象

JAXP 1.0

メソッド

getParser()

説明
このクラスの実装によってカプセル化される SAX パーサーを戻します。

構文
```
public org.xml.sax.Parser getParser()
```

コメント
使用できません。getXMLReader() を使用してください。

getProperty(String)

説明
org.xml.sax.XMLReader の実際の実装で要求された特定のプロパティを戻します。

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

パラメータ
name ― 取り出すプロパティの名前

戻り値
要求されたプロパティの値

例外
SAXNotRecognizedException ― 実際の XMLReader でプロパティ名を認識しない場合

SAXNotSupportedException ― 実際の XMLReader でプロパティ名は認識するが、そのプロパティはサポートしていない場合

関連項目
org.xml.sax.XMLReader.getProperty

getXMLReader()

説明

このクラスの実装によってカプセル化される XMLReader を戻します。

構文

```
public org.xml.sax.XMLReader getXMLReader()
```

isNamespaceAware()

説明

このパーサーが名前空間を認識するように構成されているかどうかを示します。

構文

```
public boolean isNamespaceAware()
```

isValidating()

説明

このパーサーが XML 文書の妥当性をチェック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(String, Object)

説明

org.xml.sax.XMLReader の実際の実装に特定のプロパティを設定します。

構文

```
public void setProperty(java.lang.String name, java.lang.Object value)
```

パラメータ

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

例外

`SAXNotRecognizedException` ― 実際の XMLReader でプロパティ名を認識しない場合

`SAXNotSupportedException` ― 実際の XMLReader でプロパティ名は認識するが、そのプロパティはサポートしていない場合

関連項目

`org.xml.sax.XMLReader.setProperty`

JXSAXParserFactory

説明
XML 文書を解析するために、アプリケーションが SAX ベースのパーサーを構成して取得できる、ファクトリ API を定義します。

構文
```
public class JXSAXParserFactory

oracle.xml.jaxp.JXSAXParserFactory
```

適用対象
JAXP 1.0

コンストラクタ

JXSAXParserFactory()

説明
デフォルトのコンストラクタです。

構文
```
public  JXSAXParserFactory()
```

メソッド

getFeature(String)

説明
org.xml.sax.XMLReader の実際の実装で要求された特定のプロパティを戻します。

構文
```
public boolean getFeature(java.lang.String name)
```

パラメータ
name ― 取り出すプロパティの名前

戻り値
要求されたプロパティの値

例外
SAXNotRecognizedException ― 実際の XMLReader でプロパティ名を認識しない場合

SAXNotSupportedException ― 実際の XMLReader でプロパティ名を認識するが、そのプロパティをサポートしていない場合

関連項目
org.xml.sax.XMLReader.getProperty

isNamespaceAware()

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

構文
```
public boolean isNamespaceAware()
```

newSAXParser()

説明
現在構成されているファクトリ・パラメータを使用して、SAXParser の新しいインスタンスを作成します。

構文
```
public javax.xml.parsers.SAXParser newSAXParser()
```

例外
ParserConfigurationException ― 要求された構成を満たすパーサーを作成できない場合

setFeature(String, boolean)

説明

org.xml.sax.XMLReader の実際の実装に特定の機能を設定します。

構文

```
public void setFeature(java.lang.String name, boolean value)
```

パラメータ

name ― 設定する機能の名前

value ― 設定する機能の値

例外

SAXNotRecognizedException ― 実際の XMLReader でプロパティ名を認識しない場合

SAXNotSupportedException ― 実際の XMLReader でプロパティ名を認識するが、そのプロパティをサポートしていない場合

関連項目

org.xml.sax.XMLReader.setFeature

JXSAXTransformerFactory

説明

JXTransformerFactory インスタンスを使用すると、Transformer オブジェクトと Template オブジェクトを作成できます。

構文

```
public class JXSAXTransformerFactory
```

oracle.xml.jaxp.JXSAXTransformerFactory

コメント

作成するファクトリ実装を決定するシステム・プロパティは、"javax.xml.transform.TransformerFactory" という名前です。このプロパティは、TransformerFactory 抽象クラス（ここでは、JXSAXTransformerFactory）の具体的なサブクラスを指定します。プロパティが未定義の場合は、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。また、ContentHandler のタイプとして、Transformer オブジェクトを作成するタイプおよび Template オブジェクトを作成するタイプを提供します。

変換時に使用する XMLReader の ErrorHandler または EntityResolver をアプリケーションで設定する場合は、URIResolver を使用して、XMLReader への参照を提供する（getXMLReader を使用）SAXSource を戻す必要があります。

コンストラクタ

JXSAXTransformerFactory()

説明

デフォルトのコンストラクタです。

構文

```
public  JXSAXTransformerFactory()
```

メソッド

getAssociatedStylesheet(Source, String, String, String)

説明

XML スタイルシート処理命令（http://www.w3.org/TR/xml-stylesheet/ を参照）を使用して、ソース・パラメータで指定したドキュメントに関連し、指定した条件と一致するスタイルシート仕様を取得します。

構文

```
public javax.xml.transform.Source getAssociatedStylesheet(javax.xml.transform.Source
source, java.lang.String media, java.lang.String title, java.lang.String charset)
```

コメント

複数のスタイルシートを戻すことができることに注意してください。この場合、スタイルシートは、単一のスタイルシートにインポートまたはカスケードされたリストのように適用されます。

パラメータ

`source` ― XML ソース・ドキュメント。

`media` ― 照合するメディア属性。null を設定できます。その場合は最適なテンプレートが使用されます（つまり、alternate = no）。

`title` ― 照合するタイトル属性の値。null を設定できます。

`charset` ― 照合するキャラクタ・セット属性の値。null を設定できます。

戻り値

TransformerFactory に渡すことができるソース・オブジェクト

getAttribute(String)

説明

実際の実装の特定の属性を取り出すことができます。

構文

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

パラメータ

`name` ― 属性の名前

戻り値
属性の値

例外
`IllegalArgumentException` ― 実際の実装で属性を認識できない場合に発生します。

getErrorListener()

説明
TransformerFactory のエラー・イベント・ハンドラを取得します。

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

戻り値
現行のエラー・ハンドラ。null が戻されることはありません。

getFeature(String)

説明
機能の値を検索します。 機能名は絶対 URI です。

構文
```
public boolean getFeature(java.lang.String name)
```

パラメータ
`name` ― 機能名（絶対 URI）

戻り値
機能の現在の状態（true または false）

getURIResolver()

説明

document()、xsl:import または xsl:include で使用する URI を解決するため、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。

構文

```
public javax.xml.transform.URIResolver getURIResolver()
```

戻り値

setURIResolver を使用して設定された URIResolver

newTemplates(Source)

説明

ソースを処理して Template オブジェクトを作成します。このオブジェクトは、コンパイル済みのソースを表します。この Template オブジェクトは、複数のスレッド間で同時に使用できます。

構文

```
public javax.xml.transform.Templates newTemplates(javax.xml.transform.Source source)
```

コメント

Template オブジェクトを作成すると、TransformerFactory では、ランタイム変換に影響を与えずに変換命令のパフォーマンスを詳細に最適化できます。

パラメータ

`source` − URL、入力ストリームなどを保持するオブジェクト

戻り値

変換で使用できる Template オブジェクト。null が戻されることはありません。

例外

`TransformerConfigurationException` − Template オブジェクトの作成に失敗すると、解析中にこの例外が発生する場合があります。

newTemplatesHandler()

説明

SAX の ContentHandler イベントを処理して Template オブジェクトを作成できる TemplatesHandler オブジェクトを取得します。

構文

```
public javax.xml.transform.sax.TemplatesHandler newTemplatesHandler()
```

戻り値

TransformerHandler への参照（null 以外）。SAX の解析イベント用の ContentHandler として使用できます。

例外

`TransformerConfigurationException` － なんらかの理由で TemplatesHandler を作成できない場合

newTransformer()

説明

結果に対してソースのコピーを実行する新しい Transformer オブジェクトを作成します。

構文

```
public javax.xml.transform.Transformer newTransformer()
```

戻り値

単一スレッドでの変換の実行に使用できる Transformer オブジェクト。null が戻されることはありません。

例外

`TransformerConfigurationException` － Template オブジェクトの作成に失敗すると、解析中にこの例外が発生する場合があります。

newTransformer(Source)

説明

ソースを処理して Transformer オブジェクトを作成します。このオブジェクトは、同時に実行されている複数のスレッドで使用しないでください。スレッドごとに異なる TransformerFactory を同時に使用することは可能です。

構文

```
public javax.xml.transform.Transformer newTransformer(javax.xml.transform.Source
source)
```

パラメータ

source − URI、入力ストリームなどを保持するオブジェクト

戻り値

単一スレッドでの変換の実行に使用できる Transformer オブジェクト。null が戻されることはありません。

例外

TransformerConfigurationException − Template オブジェクトの作成に失敗すると、解析中にこの例外が発生する場合があります。

newTransformerHandler()

説明

SAX の ContentHandler イベントを処理して結果を作成できる TransformerHandler オブジェクトを取得します。変換は、識別情報（またはコピー）の変換として定義されます。たとえば、一連の SAX 解析イベントを DOM ツリーにコピーします。

構文

```
public javax.xml.transform.sax.TransformerHandler newTransformerHandler()
```

戻り値

TransformerHandler への参照（null 以外）。SAX の解析イベント用の ContentHandler として使用できます。

例外

TransformerConfigurationException − なんらかの理由で TransformerHandler を作成できない場合

newTransformerHandler(Source)

説明

SAX の ContentHandler イベントを、引数で指定された変換命令に基づいて処理して結果を作成できる、TransformerHandler オブジェクトを取得します。

構文

```
public javax.xml.transform.sax.TransformerHandler
newTransformerHandler(javax.xml.transform.Source src)
```

パラメータ

`src` ― 変換命令のソース

戻り値

SAX イベントを変換できる TransformerHandler

例外

`TransformerConfigurationException` ― なんらかの理由で TransformerHandler を作成できない場合

newTransformerHandler(Templates)

説明

SAX の ContentHandler イベントを、Templates 引数に基づいて処理して結果を作成できる、TransformerHandler オブジェクトを取得します。

構文

```
public javax.xml.transform.sax.TransformerHandler
newTransformerHandler(javax.xml.transform.Templates templates)
```

パラメータ

`templates` ― コンパイル済み変換命令

戻り値

SAX イベントを変換できる TransformerHandler

例外

`TransformerConfigurationException` ― なんらかの理由で TransformerHandler を作成できない場合

newXMLFilter(Source)

説明

指定したソースを変換命令として使用する XMLFilter を作成します。

構文

```
public org.xml.sax.XMLFilter newXMLFilter(javax.xml.transform.Source src)
```

パラメータ

`src` ― 変換命令のソース

戻り値

XMLFilter オブジェクト、この機能がサポートされていない場合は null

例外

`TransformerConfigurationException` ― なんらかの理由で TemplatesHandler を作成できない場合

newXMLFilter(Templates)

説明

Templates 引数に基づいて XMLFilter を作成します。

構文

```
public org.xml.sax.XMLFilter newXMLFilter(javax.xml.transform.Templates templates)
```

パラメータ

`templates` ― コンパイル済み変換命令

戻り値

XMLFilter オブジェクト、この機能がサポートされていない場合は null

例外

`TransformerConfigurationException` ― なんらかの理由で TemplatesHandler を作成できない場合

setAttribute(String, Object)

説明

実際の実装に特定の属性を設定できます。このコンテキストにおける属性は、実装で提供されるオプションとして定義されます。

構文

```
public void setAttribute(java.lang.String name, java.lang.Object value)
```

パラメータ

`name` ― 属性の名前

`value` ― 属性の値

例外

`IllegalArgumentException` ― 実際の実装で属性を認識できない場合に発生します。

setErrorListener(ErrorListener)

説明

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され、変換自体には使用されません。

構文

```
public void setErrorListener(javax.xml.transform.ErrorListener listener)
```

パラメータ

`listener` ― 新しいエラー・リスナー

例外

`IllegalArgumentException` ― リスナーが null の場合

setURIResolver(URIResolver)

説明

xsl:import または xsl:include で使用する URI を解決するため、変換時にデフォルトで使用されるオブジェクトを設定します。

構文

```
public void setURIResolver(javax.xml.transform.URIResolver resolver)
```

パラメータ

`resolver` − URIResolver インタフェースを実装するオブジェクト、または null

JXTransformer

説明

このクラスのインスタンスを使用すると、ソース・ツリーを結果ツリーに変換できます。

構文

```
public class JXTransformer
```

oracle.xml.jaxp.JXTransformer

コメント

このクラスのインスタンスは、TransformerFactory.newTransformer メソッドを使用して取得できます。次に、このインスタンスを使用して、様々なソースから XML を処理し、変換出力を様々な出力先に書き込みます。

このクラスのオブジェクトは、同時に実行されている複数のスレッドでは使用できません。スレッドごとに異なる Transformer を同時に使用することは可能です。

1 つの Transformer を複数回使用できます。パラメータと出力プロパティは、複数の変換にわたって保持されます。

コンストラクタ

JXTransformer()

説明

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。

構文

```
public  JXTransformer()
```

JXTransformer(XSLStylesheet)

説明

JXTransformer オブジェクトを作成します。

構文

```
public  JXTransformer(oracle.xml.parser.v2.XSLStylesheet templates)
```

パラメータ

templates − XSLStylesheet またはテンプレート

メソッド

clearParameters()

説明

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

構文

```
public void clearParameters()
```

getErrorListener()

説明

変換に影響を与えるエラー・イベント・リスナーを取得します。

構文

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

戻り値

現行のエラー・リスナー。null が戻されることはありません。

getOutputProperties()

説明

変換で使用する出力プロパティのコピーを取得します。

構文

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

コメント

戻されるプロパティには、ユーザーが設定したプロパティおよびスタイルシートで設定したプロパティが含まれています。これらのプロパティは、W3C 勧告の第 16 項の XSL 変換（XSLT）で指定されているデフォルト・プロパティにデフォルト設定されます。ユーザーまたはスタイルシートによって特別に設定されたプロパティは、ベースのプロパティ・リストに含める必要があります。一方、特別に設定されていない XSLT のデフォルト・プロパティは、デフォルトのプロパティ・リストに含める必要があります。したがって、getOutputProperties().getProperty(String key) は、`setOutputProperty(String, String)`、`setOutputProperties(Properties)`、スタイルシート、またはデフォルト・プロパティで設定されたプロパティを取得します。一方、getOutputProperties().get(String key) が取得するのは、`setOutputProperty(String, String)`、`setOutputProperties(Properties)` またはスタイルシートで明示的に設定されたプロパティのみです。

戻されるプロパティ・オブジェクトの推移が、変換で使用するプロパティに影響を与えることはありません。

認識されない引数キーがあり、その引数キーが修飾された名前空間でない場合、そのプロパティは無視されます。つまり、動作は setOutputProperties の影響を受けません。

関連項目

`javax.xml.transform.OutputKeys`、`java.util.Properties`

getOutputProperty(String)

説明

変換に影響を与える出力プロパティを取得します。

構文

```
public java.lang.String getOutputProperty(java.lang.String name)
```

コメント

setOutputProperty で設定したプロパティ、またはスタイルシートで指定したプロパティも指定できます。

パラメータ

name − 出力プロパティ名を指定する文字列（null 以外）。修飾された名前空間も設定できます。

戻り値

出力プロパティの文字列値、プロパティを検出できない場合は null

例外

IllegalArgumentException ー プロパティがサポートされていない場合

関連項目

javax.xml.transform.OutputKeys

getParameter(String)

説明

setParameter（1 つまたは複数）を使用して明示的に設定されたパラメータを取得します。

構文

```
public java.lang.Object getParameter(java.lang.String name)
```

パラメータ

name ー パラメータ名

このメソッドは、デフォルトのパラメータ値を戻しません。これは、変換処理中にノード・コンテキストが評価されるまでデフォルトのパラメータ値を判別できないためです。

戻り値

setParameter で設定されたパラメータ、または指定した名前のパラメータを検出できない場合は null

getURIResolver()

説明

document() などで使用する URI を解決するためのオブジェクトを取得します。

構文

```
public javax.xml.transform.URIResolver getURIResolver()
```

戻り値

URIResolver インタフェースを実装するオブジェクト、または null

setErrorListener(ErrorListener)

説明

変換に影響を与えるエラー・イベント・リスナーを取得します。

構文

```
public void setErrorListener(javax.xml.transform.ErrorListener listener)
```

パラメータ

`listener` ― 新しいエラー・リスナー

例外

`IllegalArgumentException` ― リスナーが null の場合

setOutputProperties(Properties)

説明

変換で使用する出力プロパティを設定します。ここで設定するプロパティは、xsl:output を使用してテンプレートで設定されたプロパティをオーバーライドします。

構文

```
public void setOutputProperties(java.util.Properties oformat)
```

コメント

この関数の引数が null の場合、前に設定したプロパティは削除され、テンプレート・オブジェクトで定義された値に設定されます。

修飾されたプロパティ・キー名は、中カッコ（{}）で囲まれた名前空間 URI、およびそれに続くローカル名の 2 つの部分で構成される文字列として渡されます。この名前に null の URL がある場合、文字列に含まれるのはローカル名のみです。アプリケーションでは、名前の最初の文字が '{' かどうかを確認することによって、null 以外の URI を安全にチェックできます。

たとえば、URI とローカル名を <xyz:foo xmlns:xyz="http://xyz.foo.com/yada/baz.html"/> で定義された要素から取得した場合、その修飾名は "{http://xyz.foo.com/yada/baz.html}foo" になります。接頭辞は使用されないことに注意してください。

パラメータ

`oformat` ― 変換に影響を与えるプロパティ内の同じプロパティをオーバーライドするために使用する一連の出力プロパティ

例外

IllegalArgumentException ― 認識されない引数キーがあり、その引数キーが修飾された名前空間でない場合

関連項目

javax.xml.transform.OutputKeys、java.util.Properties

setOutputProperty(String, String)

説明

変換に影響を与える出力プロパティを設定します。

構文

```
public void setOutputProperty(java.lang.String name, java.lang.String value)
```

コメント

修飾されたプロパティ名は、中カッコ（{}）で囲まれた名前空間 URI、およびそれに続くローカル名の 2 つの部分で構成される文字列として渡されます。この名前に null の URL がある場合、文字列に含まれるのはローカル名のみです。アプリケーションでは、名前の最初の文字が '{' かどうかを確認することによって、null 以外の URI を安全にチェックできます。

たとえば、URI とローカル名を <xyz:foo xmlns:xyz="http://xyz.foo.com/yada/baz.html"/> で定義された要素から取得した場合、その修飾名は "{http://xyz.foo.com/yada/baz.html}foo" になります。接頭辞は使用されないことに注意してください。

setOutputProperties(Properties) に渡されるプロパティ・オブジェクトは、このメソッドをコールしても影響を受けません。

パラメータ

name ― 出力プロパティ名を指定する文字列（null 以外）。修飾された名前空間も設定できます。

value ― 出力プロパティの文字列値（null 以外）

例外

IllegalArgumentException ― プロパティがサポート対象でなく、名前空間で修飾されていない場合

関連項目

javax.xml.transform.OutputKeys

setParameter(String, Object)

説明

変換で使用するパラメータを追加します。

構文

```
public void setParameter(java.lang.String name, java.lang.Object value)
```

コメント

修飾名は、中カッコ（{}）で囲まれた名前空間 URI、およびそれに続くローカル名の 2 つの部分で構成される文字列として渡されます。この名前に null の URL がある場合、文字列に含まれるのはローカル名のみです。アプリケーションでは、名前の最初の文字が '{' かどうかを確認することによって、null 以外の URI を安全にチェックできます。

たとえば、URI とローカル名を <xyz:foo xmlns:xyz="http://xyz.foo.com/yada/baz.html"/> で定義された要素から取得した場合、その修飾名は "{http://xyz.foo.com/yada/baz.html}foo" になります。接頭辞は使用されないことに注意してください。

パラメータ

`name` ― パラメータの名前。中カッコ（{}）で囲まれた名前空間 URI で始まる名前を使用できます。

`value` ― 値オブジェクト。有効な Java オブジェクトを指定できます。拡張機能で使用するために、適切なオブジェクトを強制するか、またはオブジェクトを渡すのみかはプロセッサによって異なります。

setURIResolver(URIResolver)

説明

document() で使用する URI を解決するためのオブジェクトを設定します。現在、document() 関数で URIResolver はサポートされていません。

構文

```
public void setURIResolver(javax.xml.transform.URIResolver resolver)
```

コメント

リゾルバの引数が null の場合は、URIResolver の値が消去され、デフォルト動作が使用されます。

パラメータ

`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(Source, Result)

説明

ソース・ツリーを処理して出力結果を作成します。

構文

```
public void transform(javax.xml.transform.Source xmlSource,
javax.xml.transform.Result outputTarget)
```

パラメータ

`xmlSource` ― ソース・ツリーの入力

`outputTarget` ― 出力ターゲット

例外

`TransformerException` ― 変換中にリカバリ不能のエラーが発生した場合

XSLT Processor クラス

表 11-24「XSLT Processor クラスの概要」では、oracle.xml.parser.v2 パッケージに含まれる XSLT Processor クラスの概要を説明します。

表 11-24 XSLT Processor クラスの概要

クラス	説明
oraxsl クラス	複数の XML 文書にスタイルシートを適用するコマンドライン・インタフェースを提供します。
XPathException クラス	XSL 変換時に例外が発生したことを示します。
XSLProcessor クラス	事前に構成された `XSLStylesheet` を使用して、入力 XML 文書を変換するメソッドを提供します。
XSLStylesheet クラス	XSL スタイルシートの情報（テンプレート、キー、変数および属性セットなど）を保持します。

oraxsl クラス

oraxsl の説明

oraxsl クラスは、複数の XML 文書にスタイルシートを適用するコマンドライン・インタフェースを提供します。その動作方法を決定するいくつかのコマンドライン・オプションを受け入れます。

oraxsl の構文

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

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

oraxsl の使用方法

```
java oraxsl options* source? stylesheet? result?
```

表 11-25 oraxsl のコマンドライン・オプション

コマンド	説明
-w	警告の表示
-e <error log>	エラーを書き込むファイル
-l <xml file list>	変換するファイルのリスト
-d <directory>	変換するファイルがあるディレクトリ
-x <source extension>	除外する拡張機能
-i <source extension>	挿入する拡張機能
-s <stylesheet>	使用するスタイルシート
-r <result extension>	結果に使用する拡張機能
-o <result extension>	結果を格納するディレクトリ
-p <param list>	パラメータのリスト
-t <# of threads>	使用するスレッドの数
-v	冗長モード

oraxsl のメソッド

表 11-26 oraxsl のメソッドの概要

メソッド	説明
oraxsl()、11-313 ページ	クラスのコンストラクタです。
main()、11-313 ページ	oraxsl ドライバを起動します。

oraxsl()

説明

クラスのコンストラクタです。

構文

```
public  oraxsl()
```

main()

説明

oraxsl ドライバを起動します。

構文

```
public static void main(java.lang.String[] args)
```

パラメータ

表 11-27 main() のパラメータ

パラメータ	説明
args	コマンドラインの引数

XPathException クラス

説明

XPath 処理時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

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--oracle.xml.util.XMLException
                    |
                    +--oracle.xml.parser.v2.XSLException
                          |
                          +--oracle.xml.parser.v2.XPathException
```

実装されるインタフェース

```
java.io.Serializable
```

メソッド

getErrorID()

説明

エラー ID を取得します。

構文

```
public int getErrorID()
```

getMessage()

説明

エラー ID およびエラー・パラメータからエラー・メッセージを作成するために、getMessage をオーバーライドします。

構文

```
public java.lang.String getMessage()
```

オーバーライド

クラス java.lang.Throwable の java.lang.Throwable.getMessage()

getMessage(XMLError)

説明

パラメータとして送られた XMLError に基づいて検出されたメッセージを取得します。

構文

```
public java.lang.String getMessage(XMLError err)
```

パラメータ

`err` ― エラー・メッセージの取得に使用される XMLError

XSLException クラス

XSLException の説明

XSL 変換時に例外が発生したことを示します。

XSLException の構文

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

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--oracle.xml.util.XMLException
                    |
                    +--oracle.xml.parser.v2.XSLException
```

XSLException のダイレクト・サブクラス

```
XPathException
```

XSLException により実装されるインタフェース

```
java.io.Serializable
```

XSLException のコンストラクタ

XSLException()

説明

新しい XSL の例外を生成します。

構文

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

XSLExtensionElement クラス

XSLExtensionElement の説明

拡張要素のベース要素です。

XSLExtensionElement の構文

```
public class XSLExtensionElement
```

oracle.xml.parser.v2.XSLExtensionElement

XSLExtensionElement のメソッド

表 11-28 XSLExtensionElement のメソッドの概要

メソッド	説明
XSLExtensionElement()	デフォルトのコンストラクタです。
getAttributeTemplateValue()	属性値をテンプレートとして取得します。
getAttributeValue()	属性値を取得します。
getChildNodes()	拡張要素の childNodes を取得します。
processAction()	拡張要素の本体を実行するためにコールする関数です。
processContent()	拡張要素の内容を処理します。

XSLExtensionElement()

説明

デフォルトのコンストラクタです。

構文

```
public  XSLExtensionElement()
```

getAttributeTemplateValue()

説明

属性値をテンプレートとして取得します。

構文

```
protected final String getAttributeTemplateValue(
          XSLTContext context, String namespace, String name)
```

パラメータ

表 11-29 getAttributeTempateValue のパラメータ

パラメータ	説明
context	XSLTContext
namespace	属性の名前空間
name	属性の名前

戻り値

属性の値

getAttributeValue()

説明

属性値を取得します。

構文

```
protected final String getAttributeValue(String namespace, String name);
```

パラメータ

表 11-30 getAttributeValue のパラメータ

パラメータ	説明
namespace	属性の名前空間
name	属性の名前

戻り値

属性の値

getChildNodes()

説明

拡張要素の childNodes を取得します。

構文

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

戻り値

Nodelist

processAction()

説明

拡張要素の本体を実行するためにコールする関数です。

構文

```
public void processAction(XSLTContext context);
```

パラメータ

表 11-31 processAction のパラメータ

パラメータ	説明
context	XSLTContext

processContent()

説明

拡張要素の内容を処理します。

構文

```
protected final void processContent(XSLTContext context);
```

パラメータ

表 11-32 processContent のパラメータ

パラメータ	説明
context	XSLTContext

XSLProcessor クラス

XSLProcessor の説明

このクラスは、事前に構成された XSLStylesheet を使用して、入力 XML 文書を変換するメソッドを提供します。行われる変換は、XSLT 1.0 仕様に準拠しています。

XSLProcessor の構文

```
public class XSLProcessor
```

oracle.xml.parser.v2.XSLProcessor

XSLProcessor のメソッド

表 11-33 XSLProcessor のメソッドの概要

メソッド	説明
XSLProcessor()	デフォルトのコンストラクタです。
getParam()	
newXSLStylesheet()	XSLStylesheet を構成します。
processXSL()	入力 XML 文書を変換します。
removeParam()	最上位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()	すべてのパラメータ・セットをリセットします。
setBaseURL()	include/import href を解決するためのベース URL を設定します。
setEntityResolver()	include/import href を解決するためのエンティティ・リゾルバを設定します。
setErrorStream()	警告の出力に使用する出力ストリームを設定します。
setLocale()	アプリケーションは、このメソッドでエラー・レポート用のロケールを設定できます。
setParam()	最上位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()	オーバーライドする XSLOutput オブジェクトを設定します。
showWarnings()	警告の内容を出力するかどうかを切り替えます。

XSLProcessor()

説明
デフォルトのコンストラクタです。

構文
```
public  XSLProcessor();
```

getParam()

説明
最上位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 java.lang.Object getParam(String uri, String name)
```

パラメータ
uri ― パラメータの名前空間 URI

name ― パラメータのローカル名

戻り値
パラメータの値

newXSLStylesheet()

説明
XSLStylesheet を構成します。XMLDocument として入力スタイルシートに再アクセスする方法がないため、XSL の document('') 関数はサポートされていません。

表 11-34 newXSLStylesheet() のバージョン

構文	説明
public XSLStylesheet newXSLStylesheet(InputStream xsl);	指定された Inputstream XSL を使用して XSLStylesheet を構成します。
public XSLStylesheet newXSLStylesheet(Reader xsl);	指定された Reader を使用して XSLStylesheet を構成します。
public XSLStylesheet newXSLStylesheet(java.net.URL xsl);	指定された URL を使用して XSLStylesheet を構成します。
public XSLStylesheet newXSLStylesheet(XMLDocument xsl);	指定された XMLDocument を使用して XSLStylesheet を構成します。

パラメータ

表 11-35 newXSLStylesheet() のパラメータ

パラメータ	説明
xsl	XSL 入力

戻り値

新しい XSL スタイルシート

例外

`XSLException` ― エラーの場合

processXSL()

説明

入力 XML 文書を変換します。表 11-36 では、オプションについて説明します。

表 11-36 processXSL() のバージョン

構文	説明
public XMLDocumentFragment processXSL(XSLStylesheet xsl, InputStream xml, URL ref);	指定された InputStream とスタイルシートを使用して、入力 XML 文書を変換します。
public XMLDocumentFragment processXSL(XSLStylesheet xsl, Reader xml, URL ref);	指定された Reader とスタイルシートを使用して、入力 XML 文書を変換します。
public XMLDocumentFragment processXSL(XSLStylesheet xsl, URL xml, URL ref)	指定された URL とスタイルシートを使用して、入力 XML 文書を変換します。
public XMLDocumentFragment processXSL(XSLStylesheet xsl, XMLDocument xml)	指定された XMLDocument とスタイルシートを使用して、入力 XML 文書を変換します。
public void processXSL(XSLStylesheet xsl, XMLDocument xml, org.xml.sax.ContentHandler handler)	指定された XMLDocument とスタイルシートを使用して、入力 XML 文書を変換します。
public XMLDocumentFragment processXSL(XSLStylesheet xsl, XMLDocumentFragment xml)	指定された XMLDocument とスタイルシートを使用して、入力 XML 文書を変換します。

表 11-36 processXSL() のバージョン（続き）

構文	説明
public void processXSL(XSLStylesheet xsl, XMLDocumentFragment xml, OutputStream os)	指定された XMLDocumentFragment とスタイルシートを使用して、入力 XML を変換します。
public void processXSL(XSLStylesheet xsl, XMLDocumentFragment xml, printWriter pw)	指定された XMLDocumentFragment とスタイルシートを使用して、入力 XML を変換します。
public void processXSL(XSLStylesheet xsl, XMLDocumentFragment xml, XMLDocumentHandler handlerXML)	指定された XMLDocument とスタイルシートを使用して、入力 XML 文書を変換します。XSLT の結果はドキュメント・フラグメントであるため、XMLDocumentHandler の次の関数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)	指定された XMLDocument とスタイルシートを使用して、入力 XML 文書を変換します。
public void processXSL(XSLStylesheet xsl, XMLDocument xml, java.io.PrintWriter pw)	指定された XMLDocument とスタイルシートを使用して、入力 XML 文書を変換します。
public void processXSL(XSLStylesheet xsl, XMLDocument xml, XMLDocumentHandler handlerXML)	指定された XMLDocument とスタイルシートを使用して、入力 XML 文書を変換します。変換の出力は XMLDocumentHandler を使用してレポートされます。XSLT の結果はドキュメント・フラグメントであるため、XMLDocumentHandler の次の関数はコールされません。setDocumentLocator、startDocument、endDocument、setDoctype、endDoctype、setXMLDecl、setTextDecl。
public XMLDocumentFragment processXSL(XSLStylesheet xsl, XMLElement inp)	指定された XMLDocument とスタイルシートを使用して、入力 XML 文書を変換します。
public void processXSL(XSLStylesheet xsl, XMLElement inp, org.xml.sax.ContentHandler handler)	指定された XMLElement とスタイルシートを使用して、入力 XML 文書を変換します。変換の出力は ContentHandler を使用してレポートされます。XSLT の結果はドキュメント・フラグメントであるため、ContentHandler の次の関数はコールされません。setDocumentLocator、startDocument、endDocument。
public void processXSL(XSLStylesheet xsl, XMLElement xml, OutputStream out)	指定された XMLElement とスタイルシートを使用して、入力 XML を変換します。

表 11-36 processXSL() のバージョン（続き）

構文	説明
public void processXSL(XSLStylesheet xsl, XMLElement xml, PrintWriter pw)	指定された XMLElement とスタイルシートを使用して、入力 XML を変換します。
public void processXSL(XSLStylesheet xsl, XMLElement xml, XMLDocumentHandler handlerXML)	指定された XMLElement とスタイルシートを使用して、入力 XML 文書を変換します。XSLT の結果はドキュメント・フラグメントであるため、XMLDocumentHandler の次の関数はコールされません。setDocumentLocator、startDocument、endDocument、setDoctype、endDoctype、setXMLDecl、setTextDecl。

パラメータ

表 11-37 processXSL のパラメータ

パラメータ	説明
xsl	変換に使用される `XSLStylesheet`
xml	変換される XML 入力
ref	入力 xml ファイルにおける外部エンティティを解決するための参照 URL
handler	コンテンツ・ハンドラ
out	結果が出力される出力ストリーム
pw	結果が印刷される PrintWriter
handlerXML	XMLDocument ハンドラ

戻り値

`XMLDocumentFragment` または `void`（関数のフォームによって決まります）

例外

`XSLException` ― エラーの場合

removeParam()

説明

最上位レベルのスタイルシート・パラメータの値を削除します。

構文

```
public void removeParam(String uri, String name)
```

パラメータ

表 11-38 removeParam のパラメータ

パラメータ	説明
uri	パラメータの URI
name	パラメータ名

例外

XSLException ― エラーの場合

resetParams()

説明

すべてのパラメータ・セットをリセットします。

構文

```
public void resetParams()
```

例外

XSLException ― エラーの場合

setBaseURL()

説明

include/import href を解決するためのベース URL を設定します。 EntityResolver（設定されている場合）はベース URL を使用する前に使用されます。setEntityResolver() を参照してください。

構文

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

パラメータ

表 11-39 setBaseURL のパラメータ

パラメータ	説明
url	設定するベース URL

setEntityResolver()

説明

include/import href を解決するためのエンティティ・リゾルバを設定します。エンティティ・リゾルバが設定されていない場合は、ベース URL（設定されている場合）が使用されます。

構文

```
public void setEntityResolver(org.xml.sax.EntityResolver eResolver)
```

パラメータ

表 11-40 setEntityResolver のパラメータ

パラメータ	説明
eResolver	エンティティ・リゾルバ

setErrorStream()

説明

警告の出力に使用する出力ストリームを設定します。警告用の出力ストリームを指定しない場合、警告は出力されません。

構文

```
public final void setErrorStream(java.io.OutputStream out)
```

パラメータ

表 11-41 setErrorStream のパラメータ

パラメータ	説明
out	エラーと警告に使用する出力ストリーム

setLocale()

説明

アプリケーションは、このメソッドでエラー・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

表 11-42 setLocale のパラメータ

パラメータ	説明
locale	設定するロケール

setParam()

説明

最上位レベルのスタイルシート・パラメータの値を設定します。

パラメータ値は、有効な XPath 式であると予測されます（したがって、文字列リテラル値は明示的に引用する必要があります）。パラメータ関数は、XSLStylesheet のパラメータ関数とともに使用することはできません。XSLProcessor のパラメータ関数を使用する場合、XSLStylesheet 関数を使用するために設定されたパラメータは無視されます。

構文

```
public void setParam(String uri, String name, Object value)
```

パラメータ

表 11-43 setParam のパラメータ

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（下位互換性を維持するため、文字列は XPath 式として処理されます）

例外

XSLException ー エラーの場合

showWarnings()

説明

警告の内容を出力するかどうかを切り替えます。

構文

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

パラメータ

表 11-44 showWarnings のパラメータ

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

XSLStylesheet の説明

このクラスは、XSL スタイルシートの情報（テンプレート、キー、変数および属性セットなど）を保持します。一度構成された同じスタイルシートを、複数の XML 文書の変換に使用できます。

XSLStylesheet の構文

```
public class XSLStylesheet

oracle.xml.parser.v2.XSLStylesheet
```

XSLStylesheet のフィールド

表 11-45 XSLStylesheet のフィールド

フィールド	構文	説明
output	public oracle.xml.parser.v2.XSLOutput output	出力します。

XSLStylesheet のメソッド

表 11-46 XSLStylesheet のメソッドの概要

メソッド	説明
getDecimalFormat()	スタイルシートで指定した 10 進数フォーマット記号を取得します。
getOutputEncoding()	`xsl:output` で指定したエンコーディングの値を取得します。
getOutputMediaType()	`xsl:output` で指定したメディア・タイプの値を取得します。
getOutputProperties()	
newTransformer()	

getDecimalFormat()

説明

スタイルシートで指定した 10 進数フォーマット記号を取得します。

構文

```
public java.text.DecimalFormatSymbols getDecimalFormat(NSName nsname)
```

パラメータ

表 11-47 getDecimalFormat のパラメータ

パラメータ	説明
nsname	XSL の 10 進数フォーマットの修飾名

戻り値

DecimalFormatSymbols

getOutputEncoding()

説明

xsl:output で指定したエンコーディングの値を取得します。

構文

```
public java.lang.String getOutputEncoding()
```

戻り値

エンコーディング

getOutputMediaType()

説明

xsl:output で指定したメディア・タイプの値を取得します。

構文

```
public java.lang.String getOutputMediaType()
```

戻り値

メディア・タイプ

getOutputProperties()

説明

xsl:output で指定された出力プロパティを java.util.Properties として戻します。

構文

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

newTransformer()

説明

変換用のスタイルシートを使用する JAXP Transformer オブジェクトを戻します。

構文

```
public javax.xml.transform.Transformer newTransformer()
```

XSLTContext クラス

XSLTContext の構文

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

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

XSLTContext の説明

Xpath 処理コンテキスト用のクラスです。

XSLTContext のメソッド

表 11-48 XSLTContext のメソッドの概要

メソッド	説明
getContextNode()	現行のコンテキスト・ノードを取得します。
getContextPosition()	現行のコンテキスト・ノードの位置を取得します。
getContextSize()	現行のコンテキストのサイズを取得します。
getError()	エラー・レポート用の XMLError インスタンスを取得します。
getVariable()	指定したスタック・オフセットにおける変数を取り出します。
reportCharacters()	現行の出力ハンドラに文字をレポートします。
reportNode()	現行の出力ハンドラに XMLNode をレポートします。
setError()	XMLError を設定します。

getContextNode()

説明
現行のコンテキスト・ノードを取得します。

構文
```
public XMLNode getContextNode()
```

戻り値
XMLNode ― 現行のコンテキスト・ノード

getContextPosition()

説明
現行のコンテキスト・ノードの位置を取得します。

構文
```
public int getContextPosition()
```

戻り値
int ― 現行のコンテキスト・ノードの位置

getContextSize()

説明
現行のコンテキストのサイズを取得します。

構文
```
public int getContextSize()
```

戻り値
int ― 現行のコンテキストのサイズ

getError()

説明
エラー・レポート用の XMLError インスタンスを取得します。

構文
```
public XMLError getError()
```

戻り値
XMLError

getVariable()

説明
指定したスタック・オフセットにおける変数を取り出します。

構文
```
public getVariable(NSName name, int offset)
```

パラメータ

表 11-49 getVariable のパラメータ

パラメータ	説明
name	変数の名前
offset	変数のオフセット

reportCharacters()

説明
現行の出力ハンドラに文字をレポートします。

構文
```
public void reportCharacters(String data, boolean disableoutesc)
```

パラメータ

表 11-50 reportCharacters のパラメータ

パラメータ	説明
chars	出力される文字列
disableoutesc	W3C の XML 1.0 仕様の定義に従って文字のエスケープを有効にするか無効にするかを示すブール値。`true` の場合は文字のエスケープが有効になり、`false` の場合は無効になります。

reportNode()

説明

現行の出力ハンドラに XMLNode をレポートします。

構文

```
public void reportNode(XMLNode node)
```

パラメータ

表 11-51 reportNode のパラメータ

パラメータ	説明
node	出力されるノード

setError()

説明

XMLError を設定します。

構文

```
public void setError(XMLError err)
```

パラメータ

表 11-52 setError のパラメータ

パラメータ	説明
err	XMLError のインスタンス

第 III 部

Oracle9*i* XDK for JavaBeans 用の Java パッケージ

第 III 部では、Oracle9*i* XDK for JavaBeans を構成する Java パッケージについて説明します。Oracle XML Transviewer Beans は、Oracle9*i* Enterprise Edition および Standard Edition に同梱されている XDK for JavaBeans の一部として提供されます。これは、Java アプリケーションまたは Applet で XML 文書を表示および変換するために使用します。標準の Java Beans として、グラフィックな Java 開発環境で使用できます。

第 III 部は、次の章で構成されています。

- 第 12 章「パッケージ oracle.xml.async」
- 第 13 章「パッケージ oracle.xml.dbviewer」
- 第 14 章「パッケージ oracle.xml.differ」
- 第 15 章「パッケージ oracle.xml.srcviewer」
- 第 16 章「パッケージ oracle.xml.transviewer」
- 第 17 章「パッケージ oracle.xml.treeviewer」

このパッケージは、製品版の Oracle XDK を完全にサポートし、市販の再配布ライセンスを備えています。製品ライブラリは、OTN および OTN-J Web サイトで定期的に更新されます。詳細は、次の OTN および OTN-J の Web サイトで XDK for JavaBeans を参照してください。

- Oracle XDK ホーム : `http://otn.oracle.co.jp/tech/xml/xdk/index.html`
- Oracle XML Developer's Kit for Java: `http://otn.oracle.co.jp/software/tech/xml/xdk/index.html`
- Oracle XML Developer's Kit for JavaBeans: `http://otn.oracle.com/tech/xml/xdk_jbeans/content.html`

12

パッケージ oracle.xml.async

この章では、パッケージ oracle.xml.async について説明します。このパッケージは、Bean インタフェースを使用して XML Parser for Java の `DOMParser` クラスをカプセル化してその機能を拡張し、非同期の解析を可能にします。このパッケージは、XDK for JavaBeans の一部として提供される Oracle XML Transviewer Beans の一部です。XML Transviewer Beans を使用すると、グラフィックで視覚的なインタフェースを XML アプリケーションに追加できます。

この章は、次の項で構成されています。

パッケージ oracle.xml.async の説明

パッケージ oracle.xml.async は、XML 文書から DOM ツリーを作成する DOMBuilder Bean（非可視の Bean）を実装します。

DOMBuilder Bean は、Bean インタフェースを使用して XML Parser for Java の `DOMParser` クラスをカプセル化してその機能を拡張し、非同期の解析を可能にします。リスナーを登録すると、Java アプリケーションは、大規模なドキュメントやドキュメントの連続するインスタンスを解析でき、制御をコール元に即時に戻すことができます。

また、バックグラウンドで、異なるスレッドにある DOM を非同期で解析できます。EventHandler インタフェースを利用して、ジョブの完了時にコール側クラスに通知します。

Oracle XML Transviewer Beans を使用したアプリケーション開発については、『Oracle9*i* XML Developer's Kit ガイド - XDK』を参照してください。

パッケージ oracle.xml.async の概要

表 12-1　oracle.xml.async のクラスの概要

クラス	説明
DOMBuilder	XML1.0 パーサーをカプセル化して、XML 文書を解析し、DOM ツリーを作成します。
DOMBuilderBeanInfo	DOMBuilder Bean に関する情報を提供します。
DOMBuilderErrorEvent	解析例外の発生時に送られるエラー・イベントを定義します。
DOMBuilderErrorListener	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実装されます。
DOMBuilderEvent	登録されているすべてのリスナーに、解析イベントについて通知するために DOMBuilder が使用するイベント・オブジェクト。
DOMBuilderListener	非同期の解析中にイベントに関する通知を受け取るために実装されます。
ResourceManager	固定数の論理リソースへのアクセスを維持する単純セマフォを実装します。
XSLTransformer	バックグラウンド・スレッドに XSL 変換を適用します。
XSLTransformerBeanInfo	XSLTransformer Bean に関する情報を提供します。
XSLTransformerErrorEvent	登録されているすべてのリスナーに、変換イベントについて通知するために XSLTransformer が使用するエラー・イベント・オブジェクト。
XSLTransformerErrorListener	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。
XSLTransformerEvent	XSLTransformer Bean に関する情報を提供します。
XSLTransformerListener	非同期の変換中にイベントに関する通知を受け取るために実装されます。

DOMBuilder

構文

```
public class DOMBuilder extends java.lang.Object implements java.io.Serializable,
oracle.xml.async.DOMBuilderConstants, java.lang.Runnable

java.lang.Object
  |
  +--oracle.xml.async.DOMBuilder
```

実装される全インタフェース

```
oracle.xml.async.DOMBuilderConstants, java.lang.Runnable, java.io.Serializable
```

説明

このクラスは、XML1.0 パーサーをカプセル化して、XML 文書を解析し、DOM ツリーを作成します。解析は別のスレッドで実行されます。ツリーの作成が完了したときには DOMBuilderListener インタフェースを使用して通知する必要があります。

フィールド

inSource

```
protected org.xml.sax.InputSource inSource
```

解析する XML データを含んだ InputSource

inStream

```
protected java.io.InputStream inStream
```

解析する XML データを含んだ InputStream

inString

```
protected java.lang.String inString
```

解析する XML データの URL を含んだ文字列

methodToCall

```
protected int methodToCall
```

入力タイプに基づいてコールする XML Parser メソッド

reader

```
protected java.io.Reader reader
```
解析する XML データを含んだ java.io.Reader

result

```
protected oracle.xml.async.XMLDocument result
```
解析対象の XML 文書

rootName

```
protected java.lang.String rootName
```
ルートとして使用される XML 要素の名前

url

```
protected java.net.URL url
```
解析する XML データの URL

コンストラクタ

DOMBuilder()

```
public  DOMBuilder()
```
新しいパーサー・オブジェクトを作成します。

DOMBuilder(int)

```
public  DOMBuilder(int id)
```
指定された 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

`id` − DOMBuilder の ID

メソッド

addDOMBuilderErrorListener(DOMBuilderErrorListener)

```
public void addDOMBuilderErrorListener(DOMBuilderErrorListener p0)
```
DOMBuilderErrorListener を追加します。

パラメータ

`p0` − 追加する DOMBuilderErrorListener

addDOMBuilderListener(DOMBuilderListener)

```
public void addDOMBuilderListener(DOMBuilderListener p0)
```
DOMBuilderListener を追加します。

パラメータ
p0 ー 追加する DOMBuilderListener

getDoctype()

```
public synchronized oracle.xml.async.DTD getDoctype()
```
DTD を取得します。

戻り値
DTD

getDocument()

```
public synchronized oracle.xml.async.XMLDocument getDocument()
```
ドキュメントを取得します。

戻り値
解析対象のドキュメント

getId()

```
public int getId()
```
パーサー・オブジェクト ID を戻します。

戻り値
DOMBuilder の ID

getReleaseVersion()

```
public synchronized java.lang.String getReleaseVersion()
```
Oracle XML Parser のバージョン番号を戻します。

戻り値
バージョン番号の文字列

getResult()

```
public synchronized org.w3c.dom.Document getResult()
```
ドキュメントを取得します。

戻り値
解析対象のドキュメント

getValidationMode()

```
public synchronized boolean getValidationMode()
```
妥当性チェック・モードを戻します。

戻り値
XML Parser が妥当性チェックをしている場合は true、していない場合は false

parse(InputSource)

```
public final synchronized void parse(org.xml.sax.InputSource in)
```
指定された入力ソースから XML を解析します。

パラメータ
in − 解析する org.xml.sax.InputSouce

例外
`XMLParseException` − 構文エラーまたはその他のエラーが発生した場合。

`SAXException` − SAX 例外。他の例外をラップしている場合もあります。

`IOException` − I/O エラー。

parse(InputStream)

```
public final synchronized void parse(java.io.InputStream in)
```
指定された入力ストリームから XML を解析します。外部エンティティと DTD を解決するために、ベース URL を設定する必要があります。

パラメータ
`in` − 解析する XML データを含んだ `InputStream`

例外

XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

関連項目

oracle.xml.parser.v2.XMLParser

parse(Reader)

```
public final synchronized void parse(java.io.Reader r)
```

指定された入力ストリームから XML を解析します。外部エンティティと DTD を解決するために、ベース URL を設定する必要があります。

パラメータ

r − 解析する XML データを含んだ Reader

例外

XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

関連項目

oracle.xml.parser.v2.XMLParser

parse(String)

```
public final synchronized void parse(java.lang.String in)
```

指定された URL から XML を解析します。

パラメータ

in − 解析する URL を含んだ String

例外

XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

parse(URL)

```
public final synchronized void parse(java.net.URL url)
```
指定された URL が指す XML 文書を解析し、対応する XML 文書階層を作成します。

パラメータ

`url` − 解析する XML 文書を指し示す URL

例外

`XMLParseException` − 構文エラーまたはその他のエラーが発生した場合。

`SAXException` − SAX 例外。他の例外をラップしている場合もあります。

`IOException` − I/O エラー。

parseDTD(InputSource, String)

```
public final synchronized void parseDTD(org.xml.sax.InputSource in, java.lang.String
rootName)
```
指定された入力ソースから XML 外部 DTD を解析します。

パラメータ

`in` − 解析する `org.xml.sax.InputSource`

`rootName` − ルート要素として使用される要素

例外

`XMLParseException` − 構文エラーまたはその他のエラーが発生した場合。

`SAXException` − SAX 例外。他の例外をラップしている場合もあります。

`IOException` − I/O エラー。

parseDTD(InputStream, String)

```
public final synchronized void parseDTD(java.io.InputStream in, java.lang.String
rootName)
```
指定された入力ストリームから XML 外部 DTD を解析します。外部エンティティと DTD を解決するために、ベース URL を設定する必要があります。

パラメータ

`in` − 解析する XML データを含んだ `InputStream`

`rootName` − ルート要素として使用される要素

例外

XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

関連項目

oracle.xml.parser.v2.XMLParser

parseDTD(Reader, String)

```
public final synchronized void parseDTD(java.io.Reader r, java.lang.String rootName)
```

指定された入力ストリームから XML 外部 DTD を解析します。外部エンティティと DTD を解決するために、ベース URL を設定する必要があります。

パラメータ

r − 解析する XML データを含んだ Reader

rootName − ルート要素として使用される要素

例外

XMLParseException − 構文エラーまたはその他のエラーが発生した場合。

SAXException − SAX 例外。他の例外をラップしている場合もあります。

IOException − I/O エラー。

関連項目

oracle.xml.parser.v2.XMLParser

parseDTD(String, String)

```
public final synchronized void parseDTD(java.lang.String in, java.lang.String
rootName)
```

指定された URL から XML 外部 DTD を解析します。

パラメータ

in − 解析する URL を含んだ String

rootName − ルート要素として使用される要素

例外

XMLParseException － 構文エラーまたはその他のエラーが発生した場合。

SAXException － SAX 例外。他の例外をラップしている場合もあります。

IOException － I/O エラー。

parseDTD(URL, String)

public final synchronized void parseDTD(java.net.URL url, java.lang.String rootName)
指定された URL が指す XML 外部 DTD ドキュメントを解析し、対応する XML 文書階層を作成します。

パラメータ

url － 解析する XML 文書を指し示す URL

rootName － ルート要素として使用される要素

例外

XMLParseException － 構文エラーまたはその他のエラーが発生した場合。

SAXException － SAX 例外。他の例外をラップしている場合もあります。

IOException － I/O エラー。

removeDOMBuilderErrorListener(DOMBuilderErrorListener)

public synchronized void removeDOMBuilderErrorListener(DOMBuilderErrorListener p0)
DOMBuilderErrorListener を削除します。

パラメータ

p0 － 削除する DOMBuilderErrorListener

removeDOMBuilderListener(DOMBuilderListener)

public synchronized void removeDOMBuilderListener(DOMBuilderListener p0)
DOMBuilderListener を削除します。

パラメータ

p0 － 削除する DOMBuilderListener

run()

```
public void run()
```
このメソッドは、スレッド内で実行します。

指定方法
インタフェース java.lang.Runnable 内の java.lang.Runnable.run()

setBaseURL(URL)

```
public synchronized void setBaseURL(java.net.URL url)
```
外部エンティティおよび DTD をロードするためのベース URL を設定します。このメソッドは、XML 文書の解析に parse(InputStream) が使用される場合にコールする必要があります。

パラメータ
`url` ― ベース URL

setDebugMode(boolean)

```
public void setDebugMode(boolean flag)
```
フラグを設定してドキュメントのデバッグ情報をオンにします。

パラメータ
`flag` ― デバッグ情報を格納するかどうかを決定します。

setDoctype(DTD)

```
public synchronized void setDoctype(oracle.xml.async.DTD dtd)
```
DTD を設定します。

パラメータ
`dtd` ― 解析時に設定および使用される `DTD`

setErrorStream(OutputStream)

```
public final synchronized void setErrorStream(java.io.OutputStream out)
```
エラーと警告の出力に使用する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ない場合、パーサーは、エラーと警告の出力に標準のエラー出力ストリーム `System.err` を使用します。

パラメータ
`out` ― エラーと警告に使用する出力ストリーム

setErrorStream(OutputStream, String)

```
public final synchronized void setErrorStream(java.io.OutputStream out,
java.lang.String enc)
```

エラーと警告の出力に使用する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ない場合、パーサーは、エラーと警告の出力に標準のエラー出力ストリーム System.err を使用します。また、指定し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。

パラメータ

out ― エラーと警告に使用する出力ストリーム

enc ― 使用するエンコーディング

例外

IOException ― サポートされていないエンコーディングが指定された場合

setErrorStream(PrintWriter)

```
public final synchronized void setErrorStream(java.io.PrintWriter out)
```

エラーと警告の出力に使用する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ない場合、パーサーは、エラーと警告の出力に標準のエラー出力ストリーム System.err を使用します。

パラメータ

out ― エラーと警告に使用する PrintWriter

setNodeFactory(NodeFactory)

```
public synchronized void setNodeFactory(oracle.xml.async.NodeFactory factory)
```

ノード・ファクトリを設定します。アプリケーションで NodeFactory を拡張し、このメソッドを使用して登録できます。パーサーは、ユーザーが提供した NodeFactory を使用して、DOM ツリーのノードを作成します。

パラメータ

factory ― 設定する NodeFactory

例外

XMLParseException ― 無効なファクトリが設定された場合

関連項目

NodeFactory

setPreserveWhitespace(boolean)

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

空白保持モードを設定します。

パラメータ

flag − 保持モード

setValidationMode(boolean)

```
public synchronized void setValidationMode(boolean yes)
```

妥当性チェック・モードを設定します。

パラメータ

yes − XML Parser が妥当性チェックをする必要があるかどうかの判断

showWarnings(boolean)

```
public synchronized void showWarnings(boolean yes)
```

警告の内容を出力するかどうかを切り替えます。

パラメータ

yes − 警告を表示するかどうかの決定

DOMBuilderBeanInfo

構文

```
public class DOMBuilderBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.async.DOMBuilderBeanInfo
```

実装される全インタフェース

```
java.beans.BeanInfo
```

説明

このクラスは、DOMBuilder Bean に関する情報を提供します。

コンストラクタ

DOMBuilderBeanInfo()

```
public  DOMBuilderBeanInfo()
```

デフォルトのコンストラクタです。

メソッド

getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

ツールバーやツールボックスなどに DOMBuilder Bean を表示するのに使用できるイメージ・オブジェクトを取得します。

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getIcon(int)

パラメータ

iconKind ー 要求されるアイコンの種類

戻り値

DOMBuilder Bean に要求されたアイコンのタイプを表すイメージ・オブジェクト。

getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

DOMBuilder Bean の PropertyDescriptors を取得します。

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getPropertyDescriptors()

戻り値

DOMBuilder Bean がサポートする編集可能なプロパティを記述する PropertyDescriptors の配列。

DOMBuilderErrorEvent

構文

```
public class DOMBuilderErrorEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--oracle.xml.async.DOMBuilderErrorEvent
```

実装される全インタフェース

```
java.io.Serializable
```

説明

このクラスは、解析例外の発生時に送られるエラー・イベントを定義します。

フィールド

```
protected java.lang.Exception e
```

発生した例外

コンストラクタ

DOMBuilderErrorEvent(Object, Exception)

```
public  DOMBuilderErrorEvent(java.lang.Object p0, java.lang.Exception e)
```

DOMBuilderErrorEvent のコンストラクタ

パラメータ

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

メソッド

getException()

```
public java.lang.Exception getException()
```

例外を取得します。

戻り値

発生した例外

getMessage()

```
public java.lang.String getMessage()
```

パーサーによって生成されたエラー・メッセージを戻します。

戻り値

エラー・メッセージ文字列

DOMBuilderErrorListener

構文

```
public interface DOMBuilderErrorListener extends java.util.EventListener
```

全スーパーインタフェース

```
java.util.EventListener
```

説明

解析中にエラーが検出された場合に通知を受け取るためには、このインタフェースを実装する必要があります。addDOMBuilderErrorListener メソッドを使用して、このインタフェースを実装するクラスを DOMBuilder に追加する必要があります。

メソッド

domBuilderErrorCalled(DOMBuilderErrorEvent)

```
public void domBuilderErrorCalled(DOMBuilderErrorEvent p0)
```

このメソッドは、パーサー・エラーが発生するとコールされます。

パラメータ

`p0` ― 解析エラーが発生すると、DOMBuilder によって作成される DOMBuilderErrorEvent オブジェクト。

DOMBuilderEvent

構文

```
public class DOMBuilderEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--oracle.xml.async.DOMBuilderEvent
```

実装される全インタフェース

```
java.io.Serializable
```

説明

登録されているすべてのリスナーに、解析イベントについて通知するために DOMBuilder が使用するイベント・オブジェクト。

フィールド

id

```
protected int id
```

ソース DOMBuilder オブジェクトの ID

コンストラクタ

DOMBuilderEvent(Object, int)

```
public  DOMBuilderEvent(java.lang.Object p0, int p1)
```

新しい DOMBuilder イベントを作成します。

パラメータ

`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

メソッド

getID()

```
public int getID()
```

DOMBuilder オブジェクトの一意な ID を戻します。DOMBuilder の複数のインスタンスがバックグラウンドで実行しているときに、この ID をもとにイベントを生成した DOMBuilder のインスタンスを識別できます。

戻り値

このイベントのソース `DOMBuilder` の一意な ID。

DOMBuilderListener

構文

```
public interface DOMBuilderListener extends java.util.EventListener
```

全スーパーインタフェース

```
java.util.EventListener
```

説明

非同期解析中に発生したイベントについての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。addDOMBuilderListener メソッドを使用して、このインタフェースを実装するクラスを DOMBuilder に追加する必要があります。

メソッド

domBuilderError(DOMBuilderEvent)

```
public void domBuilderError(DOMBuilderEvent p0)
```

このメソッドは、解析エラーが発生するとコールされます。

パラメータ

p0 − DOMBuilder によって生成される DOMBuilderEvent オブジェクト

domBuilderOver(DOMBuilderEvent)

```
public void domBuilderOver(DOMBuilderEvent p0)
```

このメソッドは、解析が完了するとコールされます。

パラメータ

p0 − DOMBuilder によって生成される DOMBuilderEvent オブジェクト

domBuilderStarted(DOMBuilderEvent)

```
public void domBuilderStarted(DOMBuilderEvent p0)
```

このメソッドは、解析が開始するとコールされます。

パラメータ

p0 − DOMBuilder によって生成される DOMBuilderEvent オブジェクト

ResourceManager

構文

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

java.lang.Object
  |
  +--oracle.xml.async.ResourceManager
```

コンストラクタ

ResourceManager(int)

```
public  ResourceManager(int i)
```

ResourceManager コンストラクタ

パラメータ

i ― 管理対象のリソースの数

メソッド

activeFound()

```
public boolean activeFound()
```

管理対象の論理リソースが使用中かどうかをチェックします。

戻り値

1 つ以上のリソースが使用中の場合は `true`、使用中のリソースがない場合は `false`

getResource()

```
public synchronized void getResource()
```

使用可能なリソースの数がゼロでない場合、このメソッドはリソースの数を 1 ずつ減らします。それ以外の場合は、リソースが解放されて使用可能になるまで待機します。

releaseResource()

```
public void releaseResource()
```

リソースを解放します。このメソッドをコールすると、使用可能なリソースの数が 1 ずつ増えます。

sleep(int)

```
public void sleep(int i)
```

try/catch をせずに Thread sleep() を使用することが可能になります。

XSLTransformer

構文

```
public class XSLTransformer extends java.lang.Object implements
java.io.Serializable, oracle.xml.async.XSLTransformerConstants, java.lang.Runnable

java.lang.Object
  |
  +--oracle.xml.async.XSLTransformer
```

実装される全インタフェース

```
java.lang.Runnable, java.io.Serializable, oracle.xml.async.XSLTransformerConstants
```

説明

バックグラウンド・スレッドに XSL 変換を適用します。

フィールド

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

コンストラクタ

XSLTransformer()

```
public  XSLTransformer()
```

XSLTransformer コンストラクタ

XSLTransformer(int)

```
public  XSLTransformer(int id)
```
識別子を受け付ける XSLTransformer コンストラクタ

パラメータ
`id` ― イベント処理中に XSLTransformer インスタンスを識別するのに使用される一意な整数。

メソッド

addXSLTransformerErrorListener(XSLTransformerErrorListener)

```
public void addXSLTransformerErrorListener(XSLTransformerErrorListener p0)
```
XSLTransformer エラー・イベント・リスナーを追加します。

パラメータ
`p0` ― 追加する XSLTransformerErrorListener

addXSLTransformerListener(XSLTransformerListener)

```
public void addXSLTransformerListener(XSLTransformerListener p0)
```
XSLTransformer リスナーを追加します。

パラメータ
`p0` ― 追加する XSLTransformerListener

getId()

```
public int getId()
```
XSLTransformer の一意な ID を戻します。

戻り値
この XSLTransformer の一意な ID

getResult()

```
public synchronized oracle.xml.async.DocumentFragment getResult()
```
結果ドキュメントのドキュメント・フラグメントを戻します。このメソッドは、変換完了の通知を受け取ってからコールします。変換はバックグラウンドで非同期的に実行されるため、processXSL の直後にこのメソッドをコールすると、結果が出るまで制御権が保持されます。

戻り値

XSL 変換の結果ドキュメント・フラグメント

processXSL(XSLStylesheet, InputStream, URL)

```
public void processXSL(oracle.xml.async.XSLStylesheet xsl, java.io.InputStream xml,
java.net.URL ref)
```
バックグラウンドで XSL 変換を起動します。制御はただちに戻されます。

パラメータ

`xsl` − XSL 変換に使用するスタイルシート

`xml` − 使用する XML 文書（java.io.InputStream として指定）

`ref` − 入力 XML ファイルにおける外部エンティティを解決するための参照 URL

例外

`XSLException` − XSL 変換時にエラーが発生した場合

processXSL(XSLStylesheet, Reader, URL)

```
public void processXSL(oracle.xml.async.XSLStylesheet xsl, java.io.Reader xml,
java.net.URL ref)
```
バックグラウンドで XSL 変換を起動します。制御はただちに戻されます。

パラメータ

`xsl` − XSL 変換に使用するスタイルシート

`xml` − 使用する XML 文書（java.io.Reader として指定）

`ref` − 入力 XML ファイルにおける外部エンティティを解決するための参照 URL

例外

`XSLException` − XSL 変換時にエラーが発生した場合

processXSL(XSLStylesheet, URL, URL)

```
public void processXSL(oracle.xml.async.XSLStylesheet xsl, java.net.URL xml,
java.net.URL ref)
```

バックグラウンドで XSL 変換を起動します。制御はただちに戻されます。

パラメータ

xsl − XSL 変換に使用するスタイルシート

xml − 使用する XML 文書（java.net.URL として指定）

ref − 入力 XML ファイルにおける外部エンティティを解決するための参照 URL

例外

XSLException − XSL 変換時にエラーが発生した場合

processXSL(XSLStylesheet, XMLDocument)

```
public void processXSL(oracle.xml.async.XSLStylesheet xsl,
oracle.xml.async.XMLDocument xml)
```

バックグラウンドで XSL 変換を起動します。制御はただちに戻されます。

パラメータ

xsl − XSL 変換に使用するスタイルシート

xml − 使用する XML 文書（DOM ツリーとして指定）

例外

XSLException − XSL 変換時にエラーが発生した場合

processXSL(XSLStylesheet, XMLDocument, OutputStream)

```
public void processXSL(oracle.xml.async.XSLStylesheet xsl,
oracle.xml.async.XMLDocument xml, java.io.OutputStream os)
```

バックグラウンドで XSL 変換を起動します。制御はただちに戻されます。

パラメータ

xsl − XSL 変換に使用するスタイルシート

xml − 使用する XML 文書（DOM ツリーとして指定）

os − XSL 変換結果が書き込まれる出力ストリーム

例外

XSLException − XSL 変換時にエラーが発生した場合

removeDOMTransformerErrorListener(XSLTransformerErrorListener)

```
public synchronized void
removeDOMTransformerErrorListener(XSLTransformerErrorListener p0)
```

XSLTransformer エラー・イベント・リスナーを削除します。

パラメータ

p0 − 削除する XSLTransformerErrorListener

removeXSLTransformerListener(XSLTransformerListener)

```
public synchronized void removeXSLTransformerListener(XSLTransformerListener p0)
```

XSLTransformer リスナーを削除します。

パラメータ

p0 − 削除する XSLTransformerListener

run()

```
public void run()
```

XSL 変換を実行する別のスレッドを起動します。

指定方法

インタフェース java.lang.Runnable 内の java.lang.Runnable.run()

setErrorStream(OutputStream)

```
public final void setErrorStream(java.io.OutputStream out)
```

XSL プロセッサによって使用されるエラー・ストリームを設定します。

パラメータ

out − XSL プロセッサのエラー出力ストリーム

showWarnings(boolean)

```
public final void showWarnings(boolean yes)
```

XSL プロセッサによって使用される showWarning フラグを設定します。

パラメータ

yes − XSL プロセッサ警告を表示するか否かを示す boolean 型の値

XSLTransformerBeanInfo

構文

```
public class XSLTransformerBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.async.XSLTransformerBeanInfo
```

実装される全インタフェース

```
java.beans.BeanInfo
```

説明

このクラスは、XSLTransformer Bean に関する情報を提供します。

コンストラクタ

XSLTransformerBeanInfo()

```
public  XSLTransformerBeanInfo()
```

デフォルトのコンストラクタです。

メソッド

getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

ツールバーやツールボックスなどに XSLTransformer Bean を表示するためのイメージ・オブジェクトを取得します。

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getIcon(int)

パラメータ

iconKind ― 要求されるアイコンの種類

戻り値

XSLTransformer Bean に要求されたアイコンのタイプを表すイメージ・オブジェクト。

getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

XSLTransformer Bean の PropertyDescriptors を取得します。

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getPropertyDescriptors()

戻り値

XSLTransformer Bean がサポートする編集可能なプロパティを記述する PropertyDescriptors の配列。

XSLTransformerErrorEvent

構文

```
public class XSLTransformerErrorEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--oracle.xml.async.XSLTransformerErrorEvent
```

実装される全インタフェース

```
java.io.Serializable
```

説明

登録されているすべてのリスナーに、変換イベントについて通知するために XSLTransformer が使用するエラー・イベント・オブジェクト。

フィールド

```
protected java.lang.Exception e
```

発生した例外

コンストラクタ

XSLTransformerErrorEvent(Object, Exception)

```
public  XSLTransformerErrorEvent(java.lang.Object p0, java.lang.Exception e)
```

XSLTransformerErrorEvent のコンストラクタ

パラメータ

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

メソッド

getException()

```
public java.lang.Exception getException()
```

XSLTransformer によって検出された一意の ID の例外を戻します。

戻り値

変換例外

getMessage()

```
public java.lang.String getMessage()
```

XSLTransformer によって検出されたエラーを説明するエラー・メッセージを戻します。

戻り値

エラー・メッセージ

XSLTransformerErrorListener

構文

```
public interface XSLTransformerErrorListener extends java.util.EventListener
```

全スーパーインタフェース

```
java.util.EventListener
```

説明

非同期変換中のエラー・イベントについて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。addXSLTransformerListener メソッドを使用して、このインタフェースを実装するクラスを XSLTransformer に追加する必要があります。

メソッド

xslTransformerErrorCalled(XSLTransformerErrorEvent)

```
public void xslTransformerErrorCalled(XSLTransformerErrorEvent p0)
```

このメソッドは、解析エラーまたは変換エラーが発生するとコールされます。

パラメータ

p0 ー XSLTransformer によって生成される XSLTransformerErrorEvent オブジェクト

XSLTransformerEvent

構文

```
public class XSLTransformerEvent extends java.util.EventObject

java.lang.Object
  |
  +--java.util.EventObject
        |
        +--oracle.xml.async.XSLTransformerEvent
```

実装される全インタフェース

```
java.io.Serializable
```

フィールド

id

```
protected int id
```
ソース XSLTransformer オブジェクトの ID

コンストラクタ

XSLTransformerEvent(Object, int)

```
public  XSLTransformerEvent(java.lang.Object p0, int p1)
```
XSLTransformer ソース・オブジェクトとその一意な ID を使用して、XSLTransformerEvent オブジェクトを作成します。

パラメータ

`p0` ― この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な ID

メソッド

getID()

```
public int getID()
```

XSLTransformer オブジェクトの一意な ID を戻します。XSLTransformer の複数のインスタンスがバックグラウンドで実行しているときに、この ID をもとにイベントを生成した XSLTransformer のインスタンスを識別できます。

戻り値

このイベント・オブジェクトのソース XSLTransformer オブジェクトの一意な `ID`

XSLTransformerListener

構文

```
public interface XSLTransformerListener extends java.util.EventListener
```

全スーパーインタフェース

```
java.util.EventListener
```

説明

非同期変換中のイベントについて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。addXSLTransformerListener メソッドを使用して、このインタフェースを実装するクラスを XSLTransformer に追加する必要があります。

メソッド

xslTransformerError(XSLTransformerEvent)

```
public void xslTransformerError(XSLTransformerEvent p0)
```

このメソッドは、解析エラーまたは変換エラーが発生するとコールされます。

パラメータ

p0 − XSLTransformer によって生成される XSLTransformerEvent オブジェクト

xslTransformerOver(XSLTransformerEvent)

```
public void xslTransformerOver(XSLTransformerEvent p0)
```

このメソッドは、変換が完了するとコールされます。

パラメータ

p0 − XSLTransformer によって生成される XSLTransformerEvent オブジェクト

xslTransformerStarted(XSLTransformerEvent)

```
public void xslTransformerStarted(XSLTransformerEvent p0)
```

このメソッドは、変換が開始するとコールされます。

パラメータ

p0 − XSLTransformer によって生成される XSLTransformerEvent オブジェクト

13

パッケージ oracle.xml.dbviewer

この章では、パッケージ oracle.xml.dbviewer について説明します。このパッケージには、Oracle XML Transviewer Beans 用の DBViewer Bean が含まれています。

Oracle XML Transviewer Beans は、Oracle9*i* XDK for JavaBeans の一部として提供されます。XML Transviewer Beans を使用すると、グラフィックで視覚的なインタフェースを XML アプリケーションに追加できます。

DBViewer Bean は、XSL スタイルシートを適用し、HTML をスクロール可能なパネルに視覚的に提示することで、データベース問合せや任意の XML を表示します。

この章は、次の項目で構成されています。

パッケージ oracle.xml.dbviewer の説明

パッケージ oracle.xml.dbviewer は、XSL スタイルシートを適用して HTML をスクロール可能なパネルに表示することで、データベース問合せや任意の XML を表示できる JavaBean を実装します。

この Bean には、次の 3 つのバッファがあります。

- XML
- XSL
- 結果

Bean API を通じて、コール側プログラムで様々なソースからバッファをロードおよび保存したり、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 バッファと XSL バッファの内容はソースまたはツリー構造で表示できます。結果バッファの内容は、HTML で表示できる他、ソースまたはツリー構造としても表示できます。XML バッファはデータベース問合せからロードできます。すべてのバッファに、Oracle データベースの CLOB 表やファイル・システムからファイルをロードして保存することができます。したがって、Bean を使用してファイル・システムとデータベースのユーザー・スキーマの間でファイルを移動することも可能です。

パッケージ oracle.xml.dbviewer の概要

表 13-1 oracle.xml.dbviewer のクラスの概要

クラス	説明
DBViewer	XSL スタイルシートを適用し、HTML を表示することで、データベース問合せや任意の XML を表示するための Java Bean です。
DBViewerBeanInfo	Bean に関する情報を提供します。

DBViewer

構文

```
public class DBViewer extends javax.swing.JPanel implements java.io.Serializable

java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--oracle.xml.dbviewer.DBViewer
```

実装される全インタフェース

```
javax.accessibility.Accessible, java.awt.image.ImageObserver,
java.awt.MenuContainer, java.io.Serializable
```

説明

Java Bean を使用して、XSL スタイルシートを適用した HTML をスクロール可能なパネルに視覚化することで、データベース問合せや任意の XML を表示できます。この Bean には、XML、XSL 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 を通じて、コール側プログラムで様々なソースからバッファをロードおよび保存したり、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 バッファと XSL バッファの内容はソースまたはツリー構造で表示できます。結果バッファの内容は、HTML で表示できる他、ソースまたはツリー構造としても表示できます。XML バッファはデータベース問合せからロードできます。すべてのバッファに、Oracle データベースの CLOB 表やファイル・システムからファイルをロードして保存することができます。したがって、コントロールを使用してファイル・システムとデータベースのユーザー・スキーマの間でファイルを移動することも可能です。

コンストラクタ

DBViewer()

```
public  DBViewer()
```

新しいインスタンスを作成します。

メソッド

getHostname()

```
public java.lang.String getHostname()
```
データベース・ホスト名を取得します。

戻り値
ホスト名

getInstancename()

```
public java.lang.String getInstancename()
```
データベース・インスタンス名を取得します。

戻り値
データベース・インスタンス名

getPassword()

```
public java.lang.String getPassword()
```
ユーザー・パスワードを取得します。

戻り値
ユーザー・パスワード

getPort()

```
public java.lang.String getPort()
```
データベース・ポート番号を取得します。

戻り値
データベース・ポート番号を含んだ文字列

getResBuffer()

```
public java.lang.String getResBuffer()
```
結果バッファの内容を取得します。

戻り値
バッファの内容

getResCLOBFileName()

```
public java.lang.String getResCLOBFileName()
```

結果 CLOB ファイルの名前を取得します。

戻り値

結果 CLOB ファイルの名前

getResCLOBTableName()

```
public java.lang.String getResCLOBTableName()
```

結果 CLOB 表の名前を取得します。

戻り値

結果 CLOB 表の名前

getResFileName()

```
public java.lang.String getResFileName()
```

結果ファイルの名前を取得します。

戻り値

XSL ファイルの名前

getUsername()

```
public java.lang.String getUsername()
```

ユーザー名を取得します。

戻り値

ユーザー名

getXmlBuffer()

```
public java.lang.String getXmlBuffer()
```

XML バッファの内容を取得します。

戻り値

バッファの内容

getXmlCLOBFileName()

```
public java.lang.String getXmlCLOBFileName()
```
XML CLOB ファイルの名前を取得します。

戻り値

XML CLOB ファイルの名前

getXmlCLOBTableName()

```
public java.lang.String getXmlCLOBTableName()
```
XML CLOB 表の名前を取得します。

戻り値

XML CLOB 表の名前

getXmlFileName()

```
public java.lang.String getXmlFileName()
```
XML ファイルの名前を取得します。

戻り値

XML ファイルの名前

getXMLStringFromSQL(String)

```
public java.lang.String getXMLStringFromSQL(java.lang.String sqlText)
```
SQL 問合せからの結果セットの XML 表示を取得します。

戻り値

XML 文字列としての問合せ結果セット

getXslBuffer()

```
public java.lang.String getXslBuffer()
```
XSL バッファの内容を取得します。

戻り値

バッファの内容

getXslCLOBFileName()

```
public java.lang.String getXslCLOBFileName()
```

XSL CLOB ファイルの名前を取得します。

戻り値

XSL CLOB ファイルの名前

getXslCLOBTableName()

```
public java.lang.String getXslCLOBTableName()
```

XSL CLOB 表の名前を取得します。

戻り値

XSL CLOB 表の名前

getXslFileName()

```
public java.lang.String getXslFileName()
```

XSL ファイルの名前を取得します。

戻り値

XSL ファイルの名前

loadResBuffer(String)

```
public void loadResBuffer(java.lang.String filename)
```

ファイルから結果バッファをロードします。

パラメータ

`filename` − ファイル名

loadResBuffer(String, String)

```
public void loadResBuffer(java.lang.String tablename, java.lang.String filename)
```

CLOB ファイルから結果バッファをロードします。

パラメータ

`tablename` − CLOB 表の名前

`filename` − CLOB ファイルの名前

loadResBuffer(XMLDocument)

```
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

`resdoc` − XMLDocument

loadResBufferFromClob()

```
public void loadResBufferFromClob()
```

CLOB ファイルから結果バッファをロードします。

loadResBufferFromFile()

```
public void loadResBufferFromFile()
```

ファイルから結果バッファをロードします。

loadXmlBuffer(String)

```
public void loadXmlBuffer(java.lang.String filename)
```

ファイルから XML バッファをロードします。

パラメータ

`filename` − ファイル名

loadXmlBuffer(String, String)

```
public void loadXmlBuffer(java.lang.String tablename, java.lang.String filename)
```

CLOB ファイルから XML バッファをロードします。

パラメータ

`tablename` − CLOB 表の名前

`filename` − CLOB ファイルの名前

loadXmlBuffer(XMLDocument)

```
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

`filename` − ファイル名

loadXmlBufferFromClob()

```
public void loadXmlBufferFromClob()
```

CLOB ファイルから XML バッファをロードします。

loadXmlBufferFromFile()

```
public void loadXmlBufferFromFile()
```

ファイルから XML バッファをロードします。

loadXMLBufferFromSQL(String)

```
public void loadXMLBufferFromSQL(java.lang.String sqltext)
```

SQL 結果セットから XML バッファをロードします。

パラメータ

`sqltext` − SQL テキスト

loadXslBuffer(String)

```
public void loadXslBuffer(java.lang.String filename)
```

ファイルから XSL バッファをロードします。

パラメータ

`filename` − ファイル名

loadXslBuffer(String, String)

```
public void loadXslBuffer(java.lang.String tablename, java.lang.String filename)
```

CLOB ファイルから XSL バッファをロードします。

パラメータ

`tablename` − CLOB 表の名前

`filename` − CLOB ファイルの名前

loadXslBuffer(XMLDocument)

```
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

`xsldoc` − XML 文書

loadXslBufferFromClob()

```
public void loadXslBufferFromClob()
```

CLOB ファイルから XSL バッファをロードします。

loadXslBufferFromFile()

```
public void loadXslBufferFromFile()
```

ファイルから XSL バッファをロードします。

parseResBuffer()

```
public oracle.xml.parser.v2.XMLDocument parseResBuffer()
```

結果バッファを解析して、ツリー・ビューとソース・ビューを更新します。

戻り値

XMLDocument

parseXmlBuffer()

```
public oracle.xml.parser.v2.XMLDocument parseXmlBuffer()
```

XML バッファを解析して、ツリー・ビューとソース・ビューを更新します。

戻り値

XMLDocument

parseXslBuffer()

```
public oracle.xml.parser.v2.XMLDocument parseXslBuffer()
```

XSL バッファを解析して、ツリー・ビューとソース・ビューを更新します。

戻り値

XMLDocument

saveResBuffer(String)

```
public void saveResBuffer(java.lang.String filename)
```

ファイルに結果バッファを保存します。

パラメータ

`filename` — CLOB ファイルの名前

saveResBuffer(String, String)

```
public void saveResBuffer(java.lang.String tablename, java.lang.String filename)
```

CLOB ファイルに結果バッファを保存します。

パラメータ

`tablename` − CLOB 表の名前

`filename` − CLOB ファイルの名前

saveResBufferToClob()

```
public void saveResBufferToClob()
```

CLOB ファイルに結果バッファを保存します。

saveResBufferToFile()

```
public void saveResBufferToFile()
```

ファイルに結果バッファを保存します。

saveXmlBuffer(String)

```
public void saveXmlBuffer(java.lang.String filename)
```

ファイルに XML バッファを保存します。

パラメータ

`filename` − ファイル名

saveXmlBuffer(String, String)

```
public void saveXmlBuffer(java.lang.String tablename, java.lang.String filename)
```

CLOB ファイルに XML バッファを保存します。

パラメータ

`tablename` − CLOB 表の名前

`filename` − CLOB ファイルの名前

saveXmlBufferToClob()

```
public void saveXmlBufferToClob()
```

CLOB ファイルに XML バッファを保存します。

saveXmlBufferToFile()

```
public void saveXmlBufferToFile()
```
ファイルに XML バッファを保存します。

saveXslBuffer(String)

```
public void saveXslBuffer(java.lang.String filename)
```
ファイルに XSL バッファを保存します。

パラメータ

`filename` ― ファイル名

saveXslBuffer(String, String)

```
public void saveXslBuffer(java.lang.String tablename, java.lang.String filename)
```
CLOB ファイルに XSL バッファを保存します。

パラメータ

`tablename` ― CLOB 表の名前

`filename` ― CLOB ファイルの名前

saveXslBufferToClob()

```
public void saveXslBufferToClob()
```
CLOB ファイルに XSL バッファを保存します。

saveXslBufferToFile()

```
public void saveXslBufferToFile()
```
ファイルに XSL バッファを保存します。

setHostname(String)

```
public void setHostname(java.lang.String hostname)
```
データベース・ホスト名を設定します。

パラメータ

`hostname` ― ホスト名

setInstancename(String)

```
public void setInstancename(java.lang.String instancename)
```
データベース・インスタンス名を設定します。

パラメータ

instancename ― データベース・インスタンスの名前

setPassword(String)

```
public void setPassword(java.lang.String password)
```
ユーザー・パスワードを設定します。

パラメータ

password ― ユーザー・パスワード

setPort(String)

```
public void setPort(java.lang.String port)
```
データベース・ポート番号を設定します。

パラメータ

port ― ポート番号を含んだ文字列

setResBuffer(String)

```
public void setResBuffer(java.lang.String text)
```
結果バッファに新しいテキストを設定します。

パラメータ

text ― 新しいテキスト

setResCLOBFileName(String)

```
public void setResCLOBFileName(java.lang.String name)
```
結果 CLOB ファイルの名前を設定します。

パラメータ

name ― 結果 CLOB ファイルの名前

setResCLOBTableName(String)

```
public void setResCLOBTableName(java.lang.String name)
```
結果 CLOB 表の名前を設定します。

パラメータ

name ― 結果 CLOB 表の名前

setResFileName(String)

```
public void setResFileName(java.lang.String name)
```
結果ファイルの名前を設定します。

パラメータ

name ― 結果ファイルの名前

setResHtmlView(boolean)

```
public void setResHtmlView(boolean on)
```
HTML として結果バッファを表示します。

setResSourceEditView(boolean)

```
public void setResSourceEditView(boolean on)
```
XML ソースとして結果バッファを表示して、編集モードに入ります。

setResSourceView(boolean)

```
public void setResSourceView(boolean on)
```
XML ソースとして結果バッファを表示します。

setResTreeView(boolean)

```
public void setResTreeView(boolean on)
```
XML ツリー・ビューとして結果バッファを表示します。

setUsername(String)

```
public void setUsername(java.lang.String username)
```
ユーザー名を設定します。

パラメータ

username ― ユーザー名

setXmlBuffer(String)

```
public void setXmlBuffer(java.lang.String text)
```

XML バッファに新しいテキストを設定します。

パラメータ

`text` − XML テキスト

setXmlCLOBFileName(String)

```
public void setXmlCLOBFileName(java.lang.String name)
```

XML CLOB 表の名前を設定します。

パラメータ

`name` − XML CLOB 表の名前

setXmlCLOBTableName(String)

```
public void setXmlCLOBTableName(java.lang.String name)
```

XML CLOB 表の名前を設定します。

パラメータ

`name` − XML CLOB 表の名前

setXmlFileName(String)

```
public void setXmlFileName(java.lang.String name)
```

XML ファイルの名前を設定します。

パラメータ

`name` − XML ファイルの名前

setXmlSourceEditView(boolean)

```
public void setXmlSourceEditView(boolean on)
```

XML ソースとして XML バッファを表示して、編集モードに入ります。

setXmlSourceView(boolean)

```
public void setXmlSourceView(boolean on)
```

XML ソースとして XML バッファを表示します。

setXmlTreeView(boolean)

```
public void setXmlTreeView(boolean on)
```
ツリーとして XML バッファを表示します。

setXslBuffer(String)

```
public void setXslBuffer(java.lang.String text)
```
XSL バッファに新しいテキストを設定します。

パラメータ

`text` − XSL テキスト

setXslCLOBFileName(String)

```
public void setXslCLOBFileName(java.lang.String name)
```
XSL CLOB ファイルの名前を設定します。

パラメータ

`name` − XSL CLOB ファイルの名前

setXslCLOBTableName(String)

```
public void setXslCLOBTableName(java.lang.String name)
```
XSL CLOB 表の名前を設定します。

パラメータ

`name` − XSL CLOB 表の名前

setXslFileName(String)

```
public void setXslFileName(java.lang.String name)
```
XSL ファイルの名前を設定します。

パラメータ

`name` − XSL ファイルの名前

setXslSourceEditView(boolean)

```
public void setXslSourceEditView(boolean on)
```
XML ソースとして XSL バッファを表示して、編集モードに入ります。

setXslSourceView(boolean)

```
public void setXslSourceView(boolean on)
```

XML ソースとして XSL バッファを表示します。

setXslTreeView(boolean)

```
public void setXslTreeView(boolean on)
```

ツリーとして XSL バッファを表示します。

transformToDoc()

```
public oracle.xml.parser.v2.XMLDocument transformToDoc()
```

XSL バッファから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()

```
public void transformToRes()
```

XML バッファ内の XML に対して、XSL バッファからスタイルシート変換を適用して、その結果を結果バッファに格納します。

transformToString()

```
public java.lang.String transformToString()
```

XSL バッファから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。

DBViewerBeanInfo

構文

```
public class DBViewerBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.dbviewer.DBViewerBeanInfo
```

実装される全インタフェース

```
java.beans.BeanInfo
```

コンストラクタ

DBViewerBeanInfo()

```
public  DBViewerBeanInfo()
```

コンストラクタ

メソッド

getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getIcon(int)

getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getPropertyDescriptors()

14

パッケージ oracle.xml.differ

この章では、XML Diff Bean のパッケージ oracle.xml.differ について説明します。このパッケージは、Oracle XDK for JavaBeans の一部です。

この章は、次の項で構成されています。

パッケージ oracle.xml.differ の説明

oracle.xml.differ に含まれるクラスは、Graphical User Interface（GUI）を使用して 2 つの XML DOM ツリーを比較する XML Diff Bean を実装します。ノードの挿入、削除、移動または変更を行うことができます。

2 つの XML ツリー間の差異を XSL コード形式で生成できます。生成された XSL コードを使用して、1 番目の XML ファイルを 2 番目の XML ファイルに変換できます。

パッケージ oracle.xml.differ のクラスの概要

表 14-1　oracle.xml.differ のクラス

クラス	説明
XMLDiff クラス	2 つの XML 文書の比較に使用するインタフェースを定義します。
XMLDiffBeanInfo クラス	`java.beans.SimpleBeanInfo` を拡張します。

XMLDiff クラス

説明

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。2 つの XML ファイルを比較して、等価かどうかをチェックできます。差異がある場合は、その差異をグラフィック形式で表示するオブジェクトが提供されます。差異は XSL として表すこともできます。差異に対応する 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、1 番目の XML ファイルを 2 番目の XML ファイルに変換できます。

構文

```
public class XMLDiff
extends java.lang.Object
implements DOMBuilderListener, DOMBuilderErrorListener, java.io.Serializable

java.lang.Object
  |
  +--oracle.xml.differ.XMLDiff
```

コンストラクタ

XMLDiff

構文

```
public XMLDiff()
```

メソッド

setFiles

説明

比較が必要な 2 つの 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。この方法は、setInput1() と setInput2() をコールするよりも高速です。

構文

```
public void setFiles(java.io.File file1,
                     java.io.File file2)
```

パラメータ

file1 － 1 番目の XML ファイル（増分の File はカンマで区切ります）

file2 － 2 番目の XML ファイル

例外

java.io.IOException － I/O エラーが発生した場合。

XMLParseException － XML 文書の解析時に発生します。

SAXException － XML 文書の解析時に発生します。

java.lang.InterruptedException － スリープ中のスレッドの割込みがあった場合。

setDocuments

説明

比較が必要な XML 文書を設定します。

構文

```
public void setDocuments(XMLDocument doc1,
                         XMLDocument doc2)
```

パラメータ

doc1 － 1 番目の XML 文書

doc2 － 2 番目の XML 文書

setInput1

説明

比較が必要な 1 番目の XML ファイルを設定します。入力ファイルは、比較のために DOM ツリーに解析されます。

構文

```
public void setInput1(java.io.File file1)
```

パラメータ

file1 － 1 番目の XML ファイル

例外

java.io.IOException − I/O エラーが発生した場合。

XMLParseException − XML 文書の解析時に発生します。

SAXException − XML 文書の解析時に発生します。

java.lang.InterruptedException − スリープ中のスレッドの割込みがあった場合。

setInput2

説明

比較が必要な 2 番目の XML ファイルを設定します。入力ファイルは、比較のために DOM ツリーに解析されます。

構文

```
public void setInput2(java.io.File file2)
```

パラメータ

file2 − 2 番目の XML ファイル

例外

java.io.IOException − I/O エラーが発生した場合。

XMLParseException − XML 文書の解析時に発生します。

SAXException − XML 文書の解析時に発生します。

java.lang.InterruptedException − スリープ中のスレッドの割込みがあった場合。

setInput1

説明

比較が必要な 1 番目の XML 文書を設定します。

構文

```
public void setInput1(XMLDocument doc1)
```

パラメータ

doc1 − 1 番目の XML 文書

setInput2

説明
比較が必要な 2 番目の XML 文書を設定します。

構文
```
public void setInput2(XMLDocument doc2)
```

パラメータ
doc2 － 2 番目の XML 文書

getDocument1

説明
ドキュメント・ルートを 1 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 getDocument1()
```

戻り値
1 番目の XML ツリーのドキュメント・ルート

getDocument2

説明
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()
```

戻り値
2 番目の XML ツリーのドキュメント・ルート

diff

説明

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。

構文

```
public boolean diff()
```

戻り値

XML ファイルまたはドキュメントが同じ場合は false、異なる場合は true

例外

`java.lang.NullPointerException` − XML ファイルが正常に解析されていない場合にこの関数がコールされると発生します。または、XML 文書が設定されていない場合に発生します。

getDiffPane1

説明

1 番目の XML ファイルの差異を視覚的に表示するテキスト・パネルを、JTextPane オブジェクトとして取得します。

構文

```
public javax.swing.JTextPane getDiffPane1()
```

戻り値

1 番目の XML ファイルの差異を視覚的に表示するテキスト・パネル

getDiffPane2

説明

2 番目の XML ファイルまたはドキュメントの差異を視覚的に表示するテキスト・パネルを、JTextPane オブジェクトとして取得します。

構文

```
public javax.swing.JTextPane getDiffPane2()
```

戻り値

2 番目の XML ファイルの差異を視覚的に表示するテキスト・パネル

setIndentIncr

説明

XSL 生成時のインデントを設定します。この関数は、generateXSLFile() または generateXSLDoc() の前にコールする必要があります。このインデントは、すべての属性にのみ適用されます。属性以外に、新たに挿入されたノードをインデントする場合は、setNewNodeIndentIncr() を参照してください。

構文

```
public void setIndentIncr(int spaces)
```

パラメータ

spaces ― 属性のインデントの増分値（空白の数）

setNewNodeIndentIncr

説明

XSL 生成時のインデントを設定します。この関数は、generateXSLFile() または generateXSLDoc() の前にコールする必要があります。このインデントは、新たに挿入されたノードにのみ適用されます（属性を除きます）。属性に対してサポートされているインデントについては、setIndentIncr() を参照してください。

構文

```
public void setNewNodeIndentIncr(int spaces)
```

パラメータ

spaces ― 新しいノードのインデントの増分値（空白の数）

generateXSLFile

説明

最初に設定された 2 つの XML ファイル間の差異を表す入力ファイル名の XSL ファイルを生成します。入力ファイル名が null の場合は、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、1 番目の XML ファイルを 2 番目の XML ファイルに変換できます。2 つの XML ファイルが同じ場合、生成された XSL によって 1 番目の XML ファイルが 2 番目の XML ファイルに変換され、1 番目と 2 番目のファイルは等価になります。

構文

```
public void generateXSLFile(java.lang.String filename)
```

パラメータ

String ― 出力する XSL ファイルの名前

例外

java.io.IOException ― XSL ファイルが正常に作成されなかった場合に発生します。

generateXSLDoc

説明

最初に設定された 2 つの XML 文書間の差異を表す XSL スタイルシートを XMLDocument として生成します。生成された XSL スタイルシートを使用して、1 番目の XML ファイルを 2 番目の XML ファイルに変換できます。2 つの XML ファイルが同じ場合、生成された XSL によって 1 番目の XML ファイルが 2 番目の XML ファイルに変換され、1 番目と 2 番目のファイルは等価になります。

構文

```
public XMLDocument generateXSLDoc()
```

戻り値

XML 文書としての XSL スタイルシート

例外

java.io.IOException ― XSL ファイルが正常に作成されなかった場合に発生します。

java.io.FileNotFoundException ― 生成された XSL ファイルが検出できなかった場合。

SAXException ― XML 文書の解析時に発生します。

XMLParseException ― XML 文書の解析時に発生します。

equals

説明

2 つのノードを比較するメソッドです。差異検出のアルゴリズムによってコールされます。必要に応じて、この関数は、カスタマイズされた比較方法にオーバーライドされます。

構文

```
protected boolean equals(Node node1,
                         Node node2)
```

パラメータ

node1 ― 比較する 1 番目のノード

node2 ― 比較する 2 番目のノード

domBuilderErrorCalled

説明

解析中にエラーが発生したとき、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。

構文

```
public void domBuilderErrorCalled(DOMBuilderErrorEvent p0)
```

指定方法

インタフェース DOMBuilderErrorListener 内の domBuilderErrorCalled

パラメータ

p0 ― パーサーによって発生するエラー・オブジェクト

domBuilderError

説明

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。

構文

```
public void domBuilderError(DOMBuilderEvent p0)
```

指定方法
インタフェース DOMBuilderListener 内の domBuilderError

パラメータ
p0 − domBuilderErrorCalled によって処理されるパーサー・イベントのパーサー・エラー

domBuilderOver

説明
解析中に 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。

構文
```
public void domBuilderOver(DOMBuilderEvent p0)
```

指定方法
インタフェース DOMBuilderListener 内の domBuilderOver

パラメータ
p0 − パーサー・イベント

domBuilderStarted

説明
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。

構文
```
public void domBuilderStarted(DOMBuilderEvent p0)
```

指定方法
インタフェース DOMBuilderListener 内の domBuilderStarted

パラメータ
p0 − パーサー・イベント

printDiffTree

説明

アルゴリズムによって差異と識別されたノードの名前と値を示す差異ツリーを出力します。デバッグ用に役立ちます。

構文

```
public void printDiffTree(int tree,
                          java.io.BufferedWriter out)
```

パラメータ

`tree` ― 出力するツリー、つまり 1 または 2

`out` ― 出力された差異ツリーを格納

例外

`java.io.IOException` ― XSL ファイルが正常に作成されなかった場合に発生します。

setNoMoves

説明

差異検出のアルゴリズムによって検出される移動がないことを確認します。この関数は、diff() 関数の前にコールする必要があります。この関数によってパフォーマンスが向上します。

構文

```
public void setNoMoves()
```

XMLDiffBeanInfo クラス

構文

```
Public class XMLDiffBeanInfo
Extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.differ.XMLDiffBeanInfo
```

コンストラクタ

XMLDiffBeanInfo

構文

```
public XMLDiffBeanInfo()
```

メソッド

getPropertyDescriptors

構文

```
public java.beans.PropertyDescriptor[ ] getPropertyDescriptors()
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の getPropertyDescriptors

getIcon

構文

```
public java.awt.Image getIcon(int iconKind)
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の getIcon

15

パッケージ oracle.xml.srcviewer

この章では、XMLSourceView Bean を含むパッケージ oracle.xml.srcviewer について説明します。

Oracle XML Transviewer Beans は、Oracle9*i* XDK for JavaBeans の一部として提供されます。XML Transviewer Beans を使用すると、グラフィックで視覚的なインタフェースを XML アプリケーションに追加できます。

この章は、次の項目で構成されています。

パッケージ oracle.xml.srcviewer の説明

oracle.xml.srcviewer のクラスは、XML 文書の属性とソースを表示するために XMLSourceView Bean を実装します。XMLSourceView Bean は、XML 文書を簡単に編集するための視覚的なインタフェースを提供します。また、テキスト・エディタを使用した XML 文書の変更時に、カラー構文をハイライトして XML と XSL のフォーマット済みファイルを表示できます。これによって、ファイルの表示と編集が容易になります。DOMBuilder Bean と統合でき、解析前後の表示、および指定した DTD に対する妥当性チェックが可能となります。Oracle XML Transviewer Beans を使用したアプリケーション開発については、第 16 章にリストされている Oracle リソースを参照してください。

パッケージ oracle.xml.srcviewer の概要

表 15-1 oracle.xml.srcviewer のクラス

クラス	説明
XMLSourceView クラス	XML 文書を表示します。
XMLSourceViewBeanInfo クラス	`java.beans.SimpleBeanInfo` を拡張します。

XMLSourceView クラス

構文

```
public class XMLSourceView extends javax.swing.JPanel implements
java.io.Serializable

java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--oracle.xml.srcviewer.XMLSourceView
```

実装される全インタフェース

```
javax.accessibility.Accessible, java.awt.image.ImageObserver,
java.awt.MenuContainer, java.io.Serializable
```

説明

XML 文書を表示します。次の XML トークン・タイプを認識します。タグ、属性名、属性値、コメント、CDATA、PCDATA、PI データ、PI 名および NOTATION 記号。各トークン・タイプには、フォアグラウンド・カラーとフォント設定があります。デフォルトのカラーおよびフォント設定は変更可能です。入力として org.w3c.dom.Document オブジェクトを取得します。

フィールド

inputDOMDocument

```
protected org.w3c.dom.Document inputDOMDocument
```

jScrollPane

```
protected javax.swing.JScrollPane jScrollPane
```

jTextPane

```
protected javax.swing.JTextPane jTextPane
```

xmlStyledDocument

```
protected oracle.xml.srcviewer.XMLStyledDocument xmlStyledDocument
```

コンストラクタ

XMLSourceView()

```
public  XMLSourceView()
```

クラス・コンストラクタ。XMLSourceView 型のオブジェクトを作成します。

メソッド

fontGet(AttributeSet)

```
public static java.awt.Font fontGet(javax.swing.text.AttributeSet attributeset)
```

指定された attributeset（属性セット）からフォントを抽出して戻します。

パラメータ

attributeset ― 元となる属性セット

戻り値

抽出されたフォント

fontSet(MutableAttributeSet, Font)

```
public static void fontSet(javax.swing.text.MutableAttributeSet mutableattributeset,
java.awt.Font font)
```

可変属性セットのフォントを設定します。

パラメータ

mutableattributeset ― 更新する可変属性セット

font ― 可変属性セットの新しいフォント

getAttributeNameFont()

```
public java.awt.Font getAttributeNameFont()
```
属性名のフォントを戻します。

戻り値

Font オブジェクト

getAttributeNameForeground()

```
public java.awt.Color getAttributeNameForeground()
```
属性名のフォアグラウンド・カラーを戻します。

戻り値

Color オブジェクト

getAttributeValueFont()

```
public java.awt.Font getAttributeValueFont()
```
属性値のフォントを戻します。

戻り値

Font オブジェクト

getAttributeValueForeground()

```
public java.awt.Color getAttributeValueForeground()
```
属性値のフォアグラウンド・カラーを戻します。

戻り値

Color オブジェクト

getBackground()

```
public java.awt.Color getBackground()
```
バックグラウンド・カラーを戻します。

オーバーライド

クラス class java.awt.Component 内の java.awt.Component.getBackground()

戻り値

Color オブジェクト

getCDATAFont()

```
public java.awt.Font getCDATAFont()
```

CDATA のフォントを戻します。

戻り値

Font オブジェクト

getCDATAForeground()

```
public java.awt.Color getCDATAForeground()
```

CDATA のフォアグラウンド・カラーを戻します。

戻り値

Color オブジェクト

getCommentDataFont()

```
public java.awt.Font getCommentDataFont()
```

コメント・データのフォントを戻します。

戻り値

Font オブジェクト

getCommentDataForeground()

```
public java.awt.Color getCommentDataForeground()
```

コメント・データのフォアグラウンド・カラーを戻します。

戻り値

Color オブジェクト

getEditedText()

```
public java.lang.String getEditedText()
```

編集済みのテキストを戻します。

戻り値

編集済みのテキストを含んだ String オブジェクト

getJTextPane()

```
public javax.swing.JTextPane getJTextPane()
```
ビューア JTextPane コンポーネントを戻します。

戻り値

XMLSourceViewer によって使用される JTextPane オブジェクト

getMinimumSize()

```
public java.awt.Dimension getMinimumSize()
```
XMLSourceView の最小サイズを戻します。

オーバーライド

クラス javax.swing.JComponent 内の javax.swing.JComponent.getMinimumSize()

戻り値

XMLSourceView の最小サイズを含んだ Dimension オブジェクト

getNodeAtOffset(int)

```
public org.w3c.dom.Node getNodeAtOffset(int i)
```
指定されたオフセットにある XML ノードを戻します。

パラメータ

i ― ノードのオフセット

戻り値

オフセット i からの Node オブジェクト

getPCDATAFont()

```
public java.awt.Font getPCDATAFont()
```
PCDATA のフォントを戻します。

戻り値

Font オブジェクト

getPCDATAForeground()

```
public java.awt.Color getPCDATAForeground()
```

PCDATA のフォアグラウンド・カラーを戻します。

戻り値

Color オブジェクト

getPIDataFont()

```
public java.awt.Font getPIDataFont()
```

PI データのフォントを戻します。

戻り値

Font オブジェクト

getPIDataForeground()

```
public java.awt.Color getPIDataForeground()
```

PI データのフォアグラウンド・カラーを戻します。

戻り値

Color オブジェクト

getPINameFont()

```
public java.awt.Font getPINameFont()
```

PI 名のフォントを戻します。

戻り値

Font オブジェクト

getPINameForeground()

```
public java.awt.Color getPINameForeground()
```

PI データのフォアグラウンド・カラーを戻します。

戻り値

Color オブジェクト

getSymbolFont()

```
public java.awt.Font getSymbolFont()
```
NOTATION 記号のフォントを戻します。

戻り値

Font オブジェクト

getSymbolForeground()

```
public java.awt.Color getSymbolForeground()
```
NOTATION 記号のフォアグラウンド・カラーを戻します。

戻り値

Color オブジェクト

getTagFont()

```
public java.awt.Font getTagFont()
```
タグのフォントを戻します。

戻り値

Font オブジェクト

getTagForeground()

```
public java.awt.Color getTagForeground()
```
タグのフォアグラウンド・カラーを戻します。

戻り値

Color オブジェクト

getText()

```
public java.lang.String getText()
```
文字列として XML 文書を戻します。

戻り値

XML 文書を含んだ String オブジェクト

isEditable()

```
public boolean isEditable()
```
このオブジェクトが編集可能かどうかを示す boolean 型の値を戻します。

selectNodeAt(int)

```
public void selectNodeAt(int i)
```
カーソルをオフセット i にある XML ノードに移動します。

パラメータ

i ― ノードのオフセット

setAttributeNameFont(Font)

```
public void setAttributeNameFont(java.awt.Font font)
```
属性名のフォントを設定します。

パラメータ

font ― 属性名の新しいフォント

setAttributeNameForeground(Color)

```
public void setAttributeNameForeground(java.awt.Color color)
```
属性名のフォアグラウンド・カラーを設定します。

パラメータ

color ― 属性名の新しいカラー

setAttributeValueFont(Font)

```
public void setAttributeValueFont(java.awt.Font font)
```
属性値のフォントを設定します。

パラメータ

font ― 属性値の新しいフォント

setAttributeValueForeground(Color)

```
public void setAttributeValueForeground(java.awt.Color color)
```
属性値のフォアグラウンド・カラーを設定します。

パラメータ

color ― 属性値の新しいカラー

setBackground(Color)

```
public void setBackground(java.awt.Color color)
```
バックグラウンド・カラーを設定します。

オーバーライド

クラス javax.swing.JComponent 内の
javax.swing.JComponent.setBackground(java.awt.Color)

パラメータ

`color` − 新しいバックグラウンド・カラー

setCDATAFont(Font)

```
public void setCDATAFont(java.awt.Font font)
```
CDATA のフォントを設定します。

パラメータ

`font` − CDATA の新しいフォント

setCDATAForeground(Color)

```
public void setCDATAForeground(java.awt.Color color)
```
CDATA のフォアグラウンド・カラーを設定します。

パラメータ

`color` − CDATA の新しいカラー

setCommentDataFont(Font)

```
public void setCommentDataFont(java.awt.Font font)
```
コメントのフォントを設定します。

パラメータ

`font` − コメントの新しいフォント

setCommentDataForeground(Color)

```
public void setCommentDataForeground(java.awt.Color color)
```
コメントのフォアグラウンド・カラーを設定します。

パラメータ

`color` − コメントの新しいカラー

setEditable(boolean)

```
public void setEditable(boolean edit)
```
このオブジェクトを編集可能にするか否かを示す boolean 型の値を設定します。

パラメータ

`edit` − 新しい `boolean` 型の値

setPCDATAFont(Font)

```
public void setPCDATAFont(java.awt.Font font)
```
PCDATA のフォントを設定します。

パラメータ

`font` − PCDATA の新しいフォント

setPCDATAForeground(Color)

```
public void setPCDATAForeground(java.awt.Color color)
```
PCDATA のフォアグラウンド・カラーを設定します。

パラメータ

`color` − PCDATA の新しいカラー

setPIDataFont(Font)

```
public void setPIDataFont(java.awt.Font font)
```
PI データのフォントを設定します。

パラメータ

`font` − PI データの新しいフォント

setPIDataForeground(Color)

```
public void setPIDataForeground(java.awt.Color color)
```
PI データのフォアグラウンド・カラーを設定します。

パラメータ

`color` − PI データの新しいカラー

setPINameFont(Font)

```
public void setPINameFont(java.awt.Font font)
```
PI 名のフォントを設定します。

パラメータ

font − PI 名の新しいフォント

setPINameForeground(Color)

```
public void setPINameForeground(java.awt.Color color)
```
PI 名のフォアグラウンド・カラーを設定します。

パラメータ

color − PI 名の新しいカラー

setSelectedNode(Node)

```
public void setSelectedNode(org.w3c.dom.Node node)
```
選択された XML ノードにカーソル位置を設定します。

パラメータ

node − 選択されたノード

setSymbolFont(Font)

```
public void setSymbolFont(java.awt.Font font)
```
NOTATION 記号のフォントを設定します。

パラメータ

font − NOTATION 記号の新しいフォント

setSymbolForeground(Color)

```
public void setSymbolForeground(java.awt.Color color)
```
NOTATION 記号のフォアグラウンド・カラーを設定します。

パラメータ

color − NOTATION 記号の新しいカラー

setTagFont(Font)

```
public void setTagFont(java.awt.Font font)
```

タグのフォントを設定します。

パラメータ

`font` ― XML タグの新しいフォント

setTagForeground(Color)

```
public void setTagForeground(java.awt.Color color)
```

タグのフォアグラウンド・カラーを設定します。

パラメータ

`color` ― XML タグの新しいカラー

setXMLDocument(Document)

```
public void setXMLDocument(org.w3c.dom.Document document)
```

XMLviewer と XML 文書を対応付けます。

パラメータ

`document` ― 表示するドキュメント

関連項目

getText()

XMLSourceViewBeanInfo クラス

構文

```
public class XMLSourceViewBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.srcviewer.XMLSourceViewBeanInfo
```

実装される全インタフェース

```
java.beans.BeanInfo
```

コンストラクタ

XMLSourceViewBeanInfo()

```
public  XMLSourceViewBeanInfo()
```

メソッド

getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getIcon(int)

getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getPropertyDescriptors()

16

パッケージ oracle.xml.transviewer

この章では、Oracle XML Transviewer Beans のパッケージ oracle.xml.transviewer について説明します。Oracle XML Transviewer Beans は、XDK for JavaBeans の一部として提供されます。XML Transviewer Beans を使用すると、グラフィックで視覚的なインタフェースを XML アプリケーションに追加できます。

この章は、次の項目で構成されています。

パッケージ oracle.xml.transviewer の説明

パッケージ oracle.xml.transviewer は、CLOB 表の作成と削除、CLOB 表の内容のリスト表示、および指定した 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う Bean を提供します。この Bean には、XML 文書と XSL スタイルシートをデータベース内のファイル・システムまたは CLOB 表からロードし、取り出した XML と XSL のドキュメントまたはファイルを編集するためのファイル・インタフェースも含まれます。さらに、XML Transviewer Beans にはデータベース接続も含まれています。これによって、Bean は、JDBC 対応のデータベースに直接接続し、XML と XSL のファイルを取り出して格納できます。Oracle XML Transviewer Beans を使用したアプリケーション開発については、『Oracle9*i* XML Developer's Kit ガイド - XDK』を参照してください。

パッケージ oracle.xml.transviewer の概要

表 16-1　クラスの概要

クラス	説明
DBAccess	複数の XML 文書とテキスト・ドキュメントを保持できる CLOB 表を保持します。
DBAccessBeanInfo	DB アクセスの Bean 情報を提供します。
XMLTransformPanel	XML 文書に XSL 変換を適用します。
XMLTransformPanelBeanInfo	XMLTransormPanel の Bean 情報を提供します。
XMLTransViewer	コマンドラインから使用して、XML ファイルの編集および解析、XSL 変換の編集および適用、ファイル・システムまたは Oracle9*i* データベース内での XML、XSL および結果ファイルの取出しおよび保存を実行できます。

DBAccess

構文

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

java.lang.Object
  |
  +--oracle.xml.transviewer.DBAccess
```

説明

複数の XML 文書とテキスト・ドキュメントを保持できる CLOB 表を保持します。各表は、次のような文を使用して作成します。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）ドキュメントは、表に 1 行として格納され、FILENAME フィールドには、行を検索、更新または削除するための一意な文字列が保持されます。ドキュメント・テキストは、CLOB オブジェクトとして FILEDATA フィールドに格納されます。CLOB 表は、Transviewer Bean によって自動的に保持されます。このクラスによって保持される CLOB 表は、後で Transviewer Bean で使用できます。このクラスは、CLOB 表の作成と削除、CLOB 表の内容のリスト表示、および指定した 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。

コンストラクタ

DBAccess()

```
public  DBAccess()
```

メソッド

createBLOBTable(Connection, String)

```
public boolean createBLOBTable(java.sql.Connection con, java.lang.String tablename)
```

BLOB 表を作成します。

パラメータ

con − Connection オブジェクト

tablename − 表の名前

戻り値

正常に終了した場合は true

createXMLTable(Connection, String)

```
public boolean createXMLTable(java.sql.Connection con, java.lang.String tablename)
```

XML 表を作成します。

パラメータ

con − Connection オブジェクト

tablename − 表の名前

戻り値

正常に終了した場合は true

deleteBLOBName(Connection, String, String)

```
public boolean deleteBLOBName(java.sql.Connection con, java.lang.String tablename,
java.lang.String xmlname)
```

BLOB 表からバイナリ・ファイルを削除します。

パラメータ

con − Connection オブジェクト

tablename − 表の名前

xmlname − ファイルの名前

戻り値

正常に終了した場合は true

deleteXMLName(Connection, String, String)

```
public boolean deleteXMLName(java.sql.Connection con, java.lang.String tablename,
java.lang.String xmlname)
```

XML 表からファイルを削除します。

パラメータ

con − Connection オブジェクト

tablename − 表の名前

xmlname − ファイルの名前

戻り値

正常に終了した場合は true

dropBLOBTable(Connection, String)

```
public boolean dropBLOBTable(java.sql.Connection con, java.lang.String tablename)
```

BLOB 表を削除します。

パラメータ

con − Connection オブジェクト

tablename − 表の名前

戻り値

正常に終了した場合は true

dropXMLTable(Connection, String)

```
public boolean dropXMLTable(java.sql.Connection con, java.lang.String tablename)
```

XML 表を削除します。

パラメータ

con − Connection オブジェクト

tablename − 表の名前

戻り値

正常に終了した場合は true

getBLOBData(Connection, String, String)

```
public byte[] getBLOBData(java.sql.Connection con, java.lang.String tablename,
java.lang.String xmlname)
```

BLOB 表からバイナリ・ファイルを取り出します。

パラメータ

con − Connection オブジェクト

tablename − 表の名前

xmlname − ファイルの名前

戻り値

バイト配列としてのファイル

getNameSize()

```
public int getNameSize()
```
ファイル名が保持されるフィールドのサイズを戻します。

戻り値
ファイル名のサイズ

getXMLData(Connection, String, String)

```
public java.lang.String getXMLData(java.sql.Connection con, java.lang.String
tablename, java.lang.String xmlname)
```
XML 表からテキスト・ファイルを取り出します。

パラメータ
con − Connection オブジェクト

tablename − 表の名前

xmlname − ファイルの名前

戻り値
文字列としてのファイル

getXMLNames(Connection, String)

```
public java.lang.String[] getXMLNames(java.sql.Connection con, java.lang.String
tablename)
```
XML 表内のすべてのファイルの名前を戻します。

パラメータ
con − Connection オブジェクト

tablename − 表の名前

戻り値
この表内のすべてのファイルの名前を含んだ文字列配列

getXMLTableNames(Connection, String)

```
public java.lang.String[] getXMLTableNames(java.sql.Connection con, java.lang.String
tablePrefix)
```

名前が指定された文字列で始まるすべての XML 表を取得します。

パラメータ

con − Connection オブジェクト

tablePrefix − 表プレフィックスの文字列

戻り値

tablePrefix で始まるすべての XML 表の配列

insertBLOBData(Connection, String, String, byte[])

```
public boolean insertBLOBData(java.sql.Connection con, java.lang.String tablename,
java.lang.String xmlname, byte[] xmldata)
```

BLOB 表に 1 行としてバイナリ・ファイルを挿入します。

パラメータ

con − Connection オブジェクト

tablename − 表の名前

xmlname − ファイルの名前

xmldata − ファイル・データを含んだバイト配列

戻り値

正常に終了した場合は true

insertXMLData(Connection, String, String, String)

```
public boolean insertXMLData(java.sql.Connection con, java.lang.String tablename,
java.lang.String xmlname, java.lang.String xmldata)
```

XML 表に 1 行としてテキスト・ファイルを挿入します。

パラメータ

con − Connection オブジェクト

tablename − 表の名前

xmlname − ファイルの名前

xmldata − ファイル・データを含んだ文字列

戻り値

正常に終了した場合は true

isXMLTable(Connection, String)

```
public boolean isXMLTable(java.sql.Connection con, java.lang.String tablename)
```

表が XML 表かどうかをチェックします。

パラメータ

con − Connection オブジェクト

tableName − テストする表の名前

戻り値

この表が XML 表の場合は true

replaceXMLData(Connection, String, String, String)

```
public boolean replaceXMLData(java.sql.Connection con, java.lang.String tablename,
java.lang.String xmlname, java.lang.String xmldata)
```

XML 表の 1 行としてテキスト・ファイルを置換します。

パラメータ

con − Connection オブジェクト

tablename − 表の名前

xmlname − ファイルの名前

xmldata − ファイル・データを含んだ文字列

戻り値

正常に終了した場合は true

xmlTableExists(Connection, String)

```
public boolean xmlTableExists(java.sql.Connection con, java.lang.String tablename)
```

XML 表が存在するかどうかをチェックします。

パラメータ

con − Connection オブジェクト

tablename − 表の名前

戻り値

表が存在する場合は true

DBAccessBeanInfo

構文

```
public class DBAccessBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.transviewer.DBAccessBeanInfo
```

実装される全インタフェース

```
java.beans.BeanInfo
```

コンストラクタ

DBAccessBeanInfo()

```
public  DBAccessBeanInfo()
```

コンストラクタ

メソッド

getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getIcon(int)

getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getPropertyDescriptors()

XMLTransformPanel

構文

```
public class XMLTransformPanel extends javax.swing.JPanel

java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--oracle.xml.transviewer.XMLTransformPanel
```

実装される全インタフェース

```
javax.accessibility.Accessible, java.awt.image.ImageObserver,
java.awt.MenuContainer, java.io.Serializable
```

説明

XMLTransformPanel は可視 Bean です。XML 文書に XSL 変換を適用します。結果を視覚化します。入力 XML と XSL ドキュメント / ファイルの編集が可能になります。

コンストラクタ

XMLTransformPanel()

```
public  XMLTransformPanel()
```

クラス・コンストラクタ XMLTransformPanel タイプのオブジェクトを作成します。

XMLTransformPanelBeanInfo

構文

```
public class XMLTransformPanelBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.transviewer.XMLTransformPanelBeanInfo
```

実装される全インタフェース

```
java.beans.BeanInfo
```

コンストラクタ

XMLTransformPanelBeanInfo()

```
public  XMLTransformPanelBeanInfo()
```

メソッド

getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getIcon(int)

getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getPropertyDescriptors()

XMLTransViewer

構文

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

java.lang.Object
  |
  +--oracle.xml.transviewer.XMLTransViewer
```

説明

XMLTransformPanel を使用する簡単なアプリケーションです。コマンドラインから使用して、XML ファイルの編集および解析、XSL 変換の編集および適用、ファイル・システムまたは Oracle9*i* データベース内での XML、XSL および結果ファイルの取出しおよび保存を実行できます。

コンストラクタ

XMLTransViewer()

```
public  XMLTransViewer()
```

メソッド

getReleaseVersion()

```
public static java.lang.String getReleaseVersion()
```

Oracle XML Transviewer のバージョン番号を戻します。

戻り値

バージョン番号の文字列

main(String[ ])

```
public static void main(java.lang.String[] args)
```

17

パッケージ oracle.xml.treeviewer

この章では、パッケージ oracle.xml.treeviewer について説明します。

Oracle XML Transviewer Beans は、Oracle XDK for JavaBeans の一部として提供されます。XML Transviewer Beans を使用すると、グラフィックで視覚的なインタフェースを XML アプリケーションに追加できます。

この章は、次の項で構成されています。

パッケージ oracle.xml.treeviewer の説明

パッケージ oracle.xml.treeviewer は、フォーマット済みの XML ファイルをツリー状にグラフィック表示する Treeviewer Bean を実装します。このツリーのブランチとリーフは、マウスで操作できます。この章のクラス・リファレンスで説明するとおり、`org.w3c.dom.Document` オブジェクトを入力として取得し、XML DOM ノードを認識します。

Oracle XML Transviewer の Beans を使用したアプリケーション開発については、『Oracle9*i* XML Developer's Kit ガイド - XDK』を参照してください。

関連項目： Oracle XML Developer's Kit for JavaBeans:
`http://otn.oracle.com/tech/xml/xdk_jbeans/content.html`

パッケージ oracle.xml.treeviewer の概要

表 17-1　oracle.xml.treeviewer のクラスの概要

クラス	説明
XMLTreeView	XML 文書をツリー表示します。
XMLTreeViewBeanInfo	XMLTreeView Bean の情報を提供します。

XMLTreeView

構文

```
public class XMLTreeView extends javax.swing.JPanel

java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--oracle.xml.treeviewer.XMLTreeView
```

実装される全インタフェース

```
javax.accessibility.Accessible, java.awt.image.ImageObserver,
java.awt.MenuContainer, java.io.Serializable
```

説明

XML 文書をツリー表示します。次の XML DOM ノードを認識します。タグ、属性名、属性値、コメント、CDATA、PCDATA、PI データ、PI 名および NOTATION 記号。

入力として org.w3c.dom.Document オブジェクトを取得します。

フィールド

model

```
protected oracle.xml.treeviewer.XMLTreeModel model
```

scrollPane

```
protected transient javax.swing.JScrollPane scrollPane
```

theTree

```
protected transient javax.swing.JTree theTree
```

コンストラクタ

XMLTreeView()

```
public  XMLTreeView()
```

クラス・コンストラクタ XMLTreeView タイプのオブジェクトを作成します。

メソッド

getPreferredSize()

```
public java.awt.Dimension getPreferredSize()
```

XMLTreeView の最適サイズを戻します。

オーバーライド

クラス javax.swing.JComponent 内の javax.swing.JComponent.getPreferredSize()

戻り値

XMLTreeView の最適サイズを含んだ Dimension オブジェクト

getTree()

```
protected javax.swing.JTree getTree()
```

getXMLTreeModel()

```
protected oracle.xml.treeviewer.XMLTreeModel getXMLTreeModel()
```

setXMLDocument(Document)

```
public void setXMLDocument(org.w3c.dom.Document document)
```

XMLTreeViewer と XML 文書を対応付けます。

パラメータ

document ― 表示するドキュメント

updateUI()

```
public void updateUI()
```

XMLTreeView に UI の更新 / リフレッシュを強制実行させます。

オーバーライド

クラス javax.swing.JPanel 内の javax.swing.JPanel.updateUI()

XMLTreeViewBeanInfo

構文

```
public class XMLTreeViewBeanInfo extends java.beans.SimpleBeanInfo

java.lang.Object
  |
  +--java.beans.SimpleBeanInfo
        |
        +--oracle.xml.treeviewer.XMLTreeViewBeanInfo
```

実装される全インタフェース

java.beans.BeanInfo

コンストラクタ

XMLTreeViewBeanInfo()

構文

```
public  XMLTreeViewBeanInfo()
```

メソッド

getIcon(int)

```
public java.awt.Image getIcon(int iconKind)
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getIcon(int)

getPropertyDescriptors()

```
public java.beans.PropertyDescriptor[] getPropertyDescriptors()
```

オーバーライド

クラス java.beans.SimpleBeanInfo 内の java.beans.SimpleBeanInfo.getPropertyDescriptors()

第 IV 部

Oracle SOAP 用の Java パッケージ

第 IV 部で説明する Java パッケージは、XDK for Java で Oracle SOAP に対するサポートを実装します。

第 IV 部は、次の章で構成されています。

- 第 18 章「パッケージ oracle.soap.server」
- 第 19 章「パッケージ oracle.soap.transport」
- 第 20 章「パッケージ oracle.soap.transport.http」
- 第 21 章「パッケージ oracle.soap.util.xml」

関連項目： Oracle9*i* の Oracle SOAP の機能を使用したアプリケーション開発の詳細は、次の Oracle リソースを参照してください。

- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』

18

パッケージ oracle.soap.server

この章では、パッケージ oracle.soap.server について説明します。このパッケージには、XDK for Java で Oracle SOAP をサポートするクラスが含まれています。

Oracle SOAP は、Simple Object Access Protocol の実装です。Oracle SOAP は、Apache Software Foundation が開発した SOAP オープン・ソース実装に基づいています。

この章は、次の項で構成されています。

パッケージ oracle.soap.server の説明

Simple Object Access Protocol（SOAP）は、インターネット間でリクエストとレスポンスの送受信を行うトランスポート・プロトコルです。これは、XML と HTTP に基づいています。

SOAP は、トランスポート・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。すべてのアプリケーションで使用できる標準の XML メッセージ・フォーマットを提供します。SOAP では、World Wide Web Consortium（W3C）の XML Schema 規格を使用します。

パッケージ oracle.soap.server 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。管理クライアントは、新しいサービスとプロバイダの動的な配置をサポートするサービスです。

関連項目： SOAP のバックグラウンド情報は、次の Web サイトを参照してください。

- `http://www.w3.org/TR/SOAP/`
- `http://xml.apache.org/soap`

Oracle SOAP を使用したアプリケーション開発については、『Oracle9*i* XML Developer's Kit ガイド - XDK』を参照してください。

パッケージ oracle.soap.server の概要

表 18-1　パッケージ oracle.soap.server のインタフェース

インタフェース	説明
Handler インタフェース	SOAP サーバーのトランスポータブルなハンドラに対するインタフェースを定義します。
Provider インタフェース	各タイプのサービス・プロバイダに関してサポートする必要がある機能を定義します。
ProviderManager インタフェース	SOAP エンジンがプロバイダの配置、プロバイダの配置解除、およびプロバイダ配置情報へのアクセスを行うために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	SOAP エンジンがサービスの配置、サービスの配置解除、およびサービス配置情報へのアクセスを行うために使用するサービス・マネージャを定義します。

表 18-2　パッケージ oracle.soap.server のクラス

クラス	説明
ContainerContext クラス	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	ログ出力の実装によってサポートする必要がある機能を定義します。
ProviderDeploymentDescriptor クラス	特定のプロバイダの配置情報を定義します。
RequestContext クラス	SOAP リクエストの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が設定する必要がある情報も含まれます。
SOAPServerContext クラス	サーバーが実行されているコンテナのタイプに依存しない SOAP サーバーのコンテキストを定義します。
ServiceDeploymentDescriptor クラス	プロバイダ・タイプに依存しない SOAP サービスの配置情報を定義します。
UserContext クラス	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

oracle.soap.server.**Handler** インタフェース

```
public interface Handler
```

説明

Handler は、SOAP サーバーのトランスポータブルなハンドラに対する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。

ハンドラの実装では、次のことが必要です。

- 引数を使用しないコンストラクタを提供すること
- スレッド・セーフであること

表 18-3 Handler のメソッド

メソッド	説明
init	ハンドラを 1 回のみ初期化します。
setOptions	後続の init メソッドで使用するために、ハンドラのオプションを設定します。
getOptions	ハンドラのオプションを取得します。
getName	ハンドラの名前を取得します。
setName	ハンドラの名前を設定します。
destroy	ハンドラを 1 回のみクリーン・アップします。
invoke	リクエストのハンドラを指定した連鎖タイプの一部として起動します。

フィールド

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。

メソッド

init

```
public abstract void init(SOAPServerContext ssc)
                      throws SOAPException
```

ハンドラを 1 回のみ初期化します。このメソッドは、SOAP サーバーがハンドラで他の起動処理を行う前に、サーバーで 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な状態を設定できます。このメソッドは、setOptions を使用して事前に設定されたオプションを使用します。

パラメータ

`ssc` ― 情報メッセージ用のログ出力が含まれた、SOAP サーバーのコンテキスト

例外

`SOAPException` ― ハンドラを初期化できない場合

setOptions

```
public abstract void setOptions(Properties options)
```

後続の init メソッドで使用するために、ハンドラのオプションを設定します。このメソッドは、init メソッドの前にコールする必要があります。

パラメータ

options ― ハンドラの実装に固有のオプション

getOptions

```
public abstract Properties getOptions()
```

ハンドラのオプションを取得します。

戻り値

ハンドラの実装に固有のオプション

setName

```
public abstract void setName(String name)
```

ハンドラの名前を設定します。このメソッドは、init メソッドの前にコールする必要があります。

パラメータ

name ― ハンドラ・インスタンスの名前

getName

```
public abstract String getName()
```

ハンドラの名前を取得します。

戻り値

ハンドラ・インスタンスの名前

destroy

```
public abstract void destroy()
                        throws SOAPException
```

ハンドラを 1 回のみクリーン・アップします。このメソッドは、サーバーの終了前に、SOAP サーバーで 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な状態をクリーン・アップできます。

例外

SOAPException ― 破棄できない場合

invoke

```
public abstract void invoke(int chainType,
                            RequestContext requestContext)
                              throws SOAPException
```

リクエストのハンドラを指定した連鎖タイプの一部として起動します。いずれかのハンドラで SOAPException 例外が発生した場合は、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即時に終了することに注意してください。一方、エラー連鎖内のすべてのハンドラは、いずれかのハンドラで例外が発生しているかどうかに関係なく起動されます。エラー・ハンドラ内で例外が発生した場合、その例外は記録されて廃棄されます。

パラメータ

chainType ― 次のとおりです。

- Handler.REQUEST_TYPE ― ハンドラがリクエスト連鎖の一部として起動される場合（つまり、サービスが起動する前）
- Handler.RESPONSE_TYPE ― ハンドラがレスポンス連鎖の一部として起動される場合（つまり、サービスが起動された後）
- Handler.ERROR_TYPE ― ハンドラがエラー連鎖の一部として起動された場合（つまり、リクエスト連鎖、サービス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）

requestContext ― 関連するリクエスト・コンテキスト

例外

SOAPException ― ハンドラの起動に失敗した場合

Provider インタフェース

oracle.soap.server.**Provider** インタフェース

```
public interface Provider
```

Provider インタフェースは、各サービス・プロバイダについてサポートする必要がある機能（Java クラスやストアド・プロシージャなど）を定義します。Provider インタフェースは、サービス認可とパラメータのアンマーシャリング / マーシャリングを行います。

Provider インタフェース（別名はプロバイダ・インスタンス）は、SOAP ハンドラに配置する必要があります。各プロバイダの配置では、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ことが必要です）、およびプロバイダ固有のキー - 値のペア（任意の数）を定義する必要があります。プロバイダの配置情報を指定すると、SOAP ハンドラは、このインタフェースのみを使用してプロバイダと対話します。

SOAP ハンドラは、配置されたプロバイダ・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関して、1 つ以上のインスタンスを持つことができます（必ずしもお薦めしません）。あらゆるイベントで、プロバイダの各インスタンスが複数のリクエストを同時に処理できる必要があります。

プロバイダの実装では、次のことが必要です。

- 引数を使用しないコンストラクタを提供すること
- スレッド・セーフであること

表 18-4 Provider のメソッド

メソッド	説明
destroy	プロバイダ・インスタンスを 1 回のみクリーン・アップします。
getId	このプロバイダの一意の名前を取得します。
init	プロバイダ・インスタンスを 1 回のみ初期化します。
invoke	SOAP リクエストがリクエスト・コンテキストに完全に記述されている指定のサービスで、要求されたメソッドを起動します。

init

```
public abstract void init(ProviderDeploymentDescriptor pd,
                          SOAPServerContext ssc)
                   throws SOAPException
```

プロバイダ・インスタンスを 1 回のみ初期化します。このメソッドは、プロバイダがサポートするサービスに対してハンドラがリクエストを行う前に、SOAP ハンドラで 1 回のみ起動されます。これによって、プロバイダはプロバイダのグローバルなコンテキストを設定できます。

パラメータ

pd ― プロバイダの配置情報が含まれたプロバイダ・ディスクリプタ

ssc ― 情報メッセージ用のログ出力が含まれた、SOAP サーバーのコンテキスト

例外

SOAPException ― 初期化できないためにサービスを提供できない場合

destroy

```
public abstract void destroy() throws SOAPException
```

プロバイダ・インスタンスを 1 回のみクリーン・アップします。このメソッドは、SOAP ハンドラの終了前に、ハンドラで 1 回のみ起動されます。この起動によって、プロバイダは、プロバイダのグローバルな状態をクリーン・アップできます。

例外

SOAPException ― 破棄できない場合

関連項目

init

getId

```
public abstract String getId()
```

このプロバイダの一意の名前を取得します。

戻り値

このプロバイダの名前（SOAP ハンドラ内で一意の名前）

invoke

```
public abstract void invoke(RequestContext requestContext)
                        throws SOAPException
```

SOAP リクエストがリクエスト・コンテキストに完全に記述されている指定のサービスで、要求されたメソッドを起動します。

パラメータ

`requestContext` ― プロバイダによるリクエストの処理に必要な情報が含まれた RequestContext

例外

`SOAPException` ― なんらかの理由で（ユーザーに権限がない、メソッドが存在しないなど）メソッドの起動中にエラーが発生した場合

ProviderManager インタフェース

```
Interface oracle.soap.server.ProviderManager
public interface ProviderManager
```

ProviderManager 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、SOAP エンジンがプロバイダの配置、プロバイダの配置解除、およびプロバイダ配置情報へのアクセスを行うために使用します。プロバイダ・マネージャは、配置情報をキャッシュして、その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・マネージャに対してセキュリティを提供します。プロバイダ・マネージャは、リクエストを受け付けるために、リクエストを行う URL を使用して構成できます。プロバイダ・マネージャの SOAP リクエストが他の URL に対して行われると、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であることが必要です。また、HTTP セキュリティを設定して、URL に転送できるユーザーを制御できます。

表 18-5 ProviderManager のメソッド

メソッド	説明
deploy	指定したプロバイダを配置します。
destroy	プロバイダ・マネージャをクリーン・アップします。
getRequiredRequestURI	リクエストを受け付けるために、プロバイダ・マネージャのリクエストを行う URI を取得します。
init	プロバイダ・マネージャを初期化します。
list	配置された全プロバイダのプロバイダ ID のリストを取得します。
query	指定したプロバイダのデプロイメント・ディスクリプタを取得します。
setServiceManager	プロバイダ・マネージャが使用できるサービス配置情報を管理するためのサービス・マネージャを作成します。
undeploy	指定のプロバイダの配置を解除して、そのディスクリプタを戻します。

メソッド

init

```
public abstract void init(Properties options) throws SOAPException
```

プロバイダ・マネージャを初期化します。

パラメータ

options ― 配置情報へのアクセスの設定に必要なオプション

例外

SOAPException ― 配置情報にアクセスできない場合

destroy

```
public abstract void destroy() throws SOAPException
```

プロバイダ・マネージャをクリーン・アップします。

例外

SOAPException ― プロバイダ・マネージャをクリーン・アップできない場合

setServiceManager

```
public abstract void setServiceManager(ServiceManager serviceManager)
```

プロバイダ・マネージャが使用できるサービス配置情報を管理するためのサービス・マネージャを作成します。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、プロバイダにサービスが配置されているかぎり、そのプロバイダが配置解除されないようにできます。

パラメータ

providerManager ― SOAP サーバーのプロバイダ配置情報を管理するプロバイダ・マネージャ

getRequiredRequestURI

```
public abstract String getRequiredRequestURI()
```

リクエストを受け付けるために、プロバイダ・マネージャのリクエストを行う URI を取得します。他の URI に対して行われたリクエストは拒否されます。

戻り値

プロバイダ・マネージャのリクエストを行うリクエスト URI、またはすべての URI が使用できる場合は null

undeploy

```
public abstract ProviderDeploymentDescriptor undeploy(String providerId)
                            throws SOAPException
```

指定のプロバイダの配置を解除して、そのディスクリプタを戻します。

パラメータ

`providerId` ― 配置を解除するプロバイダの ID

戻り値

配置を解除されたプロバイダの配置情報が含まれるディスクリプタ

例外

`SOAPException` ― プロバイダが検出されない場合、または配置解除が失敗した場合

deploy

```
public abstract void deploy(ProviderDeploymentDescriptor pd)
                         throws SOAPException
```

指定したプロバイダを配置します。

パラメータ

`pd` ― 配置するプロバイダのプロバイダ・ディスクリプタ

例外

`SOAPException` ― 配置できない場合

query

```
public abstract ProviderDeploymentDescriptor query(String providerId) throws
SOAPException
```

指定したプロバイダのデプロイメント・ディスクリプタを取得します。

パラメータ

`providerId` − プロバイダの ID

戻り値

指定したプロバイダの配置情報が含まれるディスクリプタ

例外

`SOAPException` − プロバイダが検出されない場合

list

```
public abstract String[] list() throws SOAPException
```

配置された全プロバイダのプロバイダ ID のリストを取得します。

戻り値

配置されたプロバイダの ID の配列

例外

`SOAPException` − プロバイダ ID をリストできない場合

ServiceManager インタフェース

```
Interface oracle.soap.server.ServiceManager
public interface ServiceManager
```

ServiceManager 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、SOAP エンジンがサービスの配置、サービスの配置解除、およびサービス配置情報へのアクセスを行うために使用します。サービス・マネージャでは、配置情報をキャッシュして、その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・マネージャに対してセキュリティを提供します。サービス・マネージャは、リクエストを受け付けるために、リクエストを行う URL を使用して構成できます。サービス・マネージャの SOAP リクエストが他の URL に対して行われると、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であることが必要です、また、HTTP セキュリティを設定して、指定の URL に転送できるユーザーを制御できます。

メソッド

init

```
public abstract void init(Properties options,
                          ProviderManager providerManager)
                   throws SOAPException
```

サービス・マネージャを初期化します。この実装では、プロバイダ・マネージャで null 値を処理できる必要があります。

パラメータ

`options` − サービス配置情報へのアクセスの設定に必要なオプション

`providerManager` − SOAP サーバーのプロバイダ配置情報を管理するプロバイダ・マネージャ、またはプロバイダ・マネージャが指定されていない場合は null。サービス・マネージャは、新しいサービスが配置されたとき、プロバイダ・マネージャを使用してプロバイダの存在を確認できます。

例外

`SOAPException` − サービス配置情報にアクセスできない場合

destroy

```
public abstract void destroy() throws SOAPException
```

サービス・マネージャをクリーン・アップします。

例外

SOAPException － サービス・マネージャをクリーン・アップできない場合

getRequiredRequestURI

```
public abstract String getRequiredRequestURI()
```

リクエストを受け付けるために、サービス・マネージャのリクエストを行う URI を取得します。他の URI に対して行われたリクエストは拒否されます。

戻り値

サービス・マネージャのリクエストを行うリクエスト URI、またはすべての URI が使用できる場合は null

undeploy

```
public abstract ServiceDeploymentDescriptor undeploy(String serviceId)
                    throws SOAPException
```

指定のサービスの配置を解除して、そのディスクリプタを戻します。

パラメータ

serviceId － 配置を解除するサービスの URI

戻り値

配置を解除されたサービスの配置情報が含まれるディスクリプタ

例外

SOAPException － サービスが検出されない場合、または配置解除が失敗した場合

deploy

```
public abstract void deploy(ServiceDeploymentDescriptor sd) throws SOAPException
```

指定したサービスを配置します。

パラメータ

sd ― 配置するサービスのサービス・ディスクリプタ

例外

SOAPException ― 配置できない場合

query

```
public abstract ServiceDeploymentDescriptor query(String serviceId) throws
SOAPException
```

指定したサービスのデプロイメント・ディスクリプタを取得します。

パラメータ

serviceId ― サービスの一意の URI

戻り値

指定したサービスの配置情報が含まれるディスクリプタ

例外

SOAPException ― サービスが検出されない場合

list

```
public abstract String[] list() throws SOAPException
```

プロバイダに関係なく、配置された全サービスのサービス ID のリストを取得します。

戻り値

配置されたサービスの ID の配列

例外

SOAPException ― サービス ID をリストできない場合

ContainerContext クラス

oracle.soap.server.**ContainerContext** クラス

```
java.lang.Object
   |
   +----oracle.soap.server.ContainerContext

public class ContainerContext
extends Object
```

ContainerContext 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、サーバーが実行されている環境（サーブレット・エンジン内など）によって異なります。このクラスに含まれるのは、コンテナ固有の内容のみです。

フィールド

SERVLET_CONTAINER

```
public static final String SERVLET_CONTAINER
```

Servlet コンテナ・タイプの値

コンストラクタ

ContainerContext

```
public ContainerContext()
```

メソッド

setContainerType

```
public void setContainerType(String containerType)
```

コンテナ・タイプを設定します。

パラメータ

`containerType` − SOAP サーバーが実行されているコンテナのタイプ

getContainerType

```
public String getContainerType()
```

コンテナ・タイプを戻します。

戻り値

SOAP サーバーが実行されているコンテナのタイプ

getHttpServlet

```
public HttpServlet getHttpServlet()
```

コンテナのタイプが SERVLET_CONTAINER の場合は、HTTP サーブレットを戻します。

戻り値

SOAP リクエストを処理している HttpServlet、またはサーブレットの属性が設定されていない場合は null

setHttpServlet

```
public void setHttpServlet(HttpServlet servlet)
```

コンテナ・タイプが SERVLET_CONTAINER で実行されている SOAP サーバーに HTTP サーブレットを設定します。

パラメータ

`servlet` − SOAP リクエストを処理する HttpServlet

getAttribute

```
public Object getAttribute(String name)
```

指定した名前の属性を戻します。指定した名前の属性がない場合は null を戻します。

パラメータ

`name` − 属性の名前を指定する文字列

戻り値

属性の値が含まれるオブジェクト、または指定した名前と一致する属性が存在しない場合は null

関連項目

`getAttributeNames`

getAttributeNames

```
public Enumeration getAttributeNames()
```

SOAP のコンテキスト内で使用可能な属性名の一覧を戻します。

戻り値

属性名の一覧

関連項目

getAttribute

setAttribute

```
public void setAttribute(String name,
                         Object object)
```

SOAP のコンテキスト内で、オブジェクトを指定の属性名にバインドします。指定した名前がすでに属性で使用されている場合、このメソッドは、古い属性を削除し、新しい属性にその名前をバインドします。名前およびオブジェクトには null を設定できません。

パラメータ

name ― 属性の名前を指定する文字列（null 以外）

object ― バインドする属性を表すオブジェクト（null 以外）

removeAttribute

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

指定した名前の属性をコンテキストから削除します。削除した後、属性の値を取得する getAttribute(java.lang.String) をコールすると、null が戻されます。

パラメータ

name ― 削除する属性の名前を指定する文字列

Logger クラス

oracle.soap.server.**Logger** クラス

```
java.lang.Object
   |
   +----oracle.soap.server.Logger

public abstract class Logger
extends Object
```

Logger クラスは、ログ出力の実装によってサポートする必要がある機能を定義します。ログ出力を使用して、エラー・メッセージと情報メッセージを永続的に記録します。

各ログ・リクエストには重大度が指定され、重大度が指定の重大度以上の場合は情報を記録する必要があります。

重大度の順位は、高い順に次のとおりです。

- SEVERITY_ERROR
- SEVERITY_STATUS
- SEVERITY_DEBUG

たとえば、重大度が SEVERITY_STATUS に設定されている場合は、重大度が SEVERITY_STATUS または SEVERITY_ERROR のログ・リクエストはすべて記録されます。

フィールド

SEVERITY_ERROR

```
public static final int SEVERITY_ERROR
```

SEVERITY_STATUS

```
public static final int SEVERITY_STATUS
```

SEVERITY_DEBUG

```
public static final int SEVERITY_DEBUG
```

SEVERITY_INVALID

```
protected static final int SEVERITY_INVALID
```

SEVERITY_NAMES

```
public static String SEVERITY_NAMES[]
```

DEFAULT_SEVERITY

```
public static final int DEFAULT_SEVERITY
```

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です。

m_severity

```
protected int m_severity
```

ログ出力の重大度の設定です。

コンストラクタ

Logger

```
public Logger()
```

メソッド

getSeverityValue

```
protected final int getSeverityValue(String severityName)
```

指定した重大度の名前に対応する重大度の値を取得します。

パラメータ

`severityName` ― 重大度レベルの名前（error など）

戻り値

重大度（`SEVERITY_xxx`）

getSeverityName

```
protected final String getSeverityName(int severity)
```

指定した重大度に対応する重大度の名前を取得します。

パラメータ

severity − 重大度レベル（SEVERITY_xxx）

戻り値

重大度の名前

init

```
public abstract void init(Properties options,
                          ContainerContext context)
                   throws SOAPException
```

ログ出力の構成パラメータを使用して、ログ出力を 1 回のみ初期化します。

パラメータ

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）

例外

SOAPException − ログ出力を初期化できない場合

getSeverity

```
public int getSeverity()
```

現行の重要度を取得します。

戻り値

ログ出力の重要度の現行設定

setSeverity

```
public void setSeverity(int severity)
```

現行の重要度を設定します。

パラメータ

severity − ログ出力の重要度の新しい設定

isLoggable

```
public boolean isLoggable(int severity)
```

指定した重要度レベルでメッセージを記録するかどうかを決定します。

パラメータ

severity − チェックする重要度レベル

戻り値

指定した重要度レベルでメッセージを記録する場合は true、それ以外の場合は false

log

```
public abstract void log(String msg,
                         int severity)
```

指定の重要度で指定のメッセージを記録します。

パラメータ

msg − 記録するメッセージ

severity − 情報を記録する際の重要度

log

```
public abstract void log(String msg,
                         Throwable t,
                         int severity)
```

指定の重要度で、指定のメッセージと例外を記録します。

パラメータ

msg ― 記録するメッセージ

t ― 記録するスロー可能な例外

severity ― 情報を記録する際の重要度

log

```
public abstract void log(Throwable t,
                         int severity)
```

指定の重要度で指定の例外を記録します。

パラメータ

t ― 記録するスロー可能な例外

severity ― 情報を記録する際の重要度

ProviderDeploymentDescriptor クラス

oracle.soap.server.**ProviderDeploymentDescriptor** クラス

```
java.lang.Object
   |
   +----oracle.soap.server.ProviderDeploymentDescriptor

public final class ProviderDeploymentDescriptor
extends Object
implements Serializable
```

ProviderDeploymentDescriptor クラスは、特定のプロバイダの配置情報を定義します。同じ実装を使用して異なるプロバイダを配置できます。プロバイダはプロバイダ・ディスクリプタによってのみ区別されます。

表 18-6 ProviderDeploymentDescriptor のクラス・メンバー

クラス・メンバー	説明
ProviderDeploymentDescriptor	クラス・コンストラクタです。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	指定したドキュメントからプロバイダ・ディスクリプタを作成します。
getClassname	このプロバイダを実装するクラスの名前を戻します。
getId	このプロバイダの一意の ID を戻します。
getOptions	プロバイダ固有のオプションを取得します。
getProviderType	プロバイダ・タイプを戻します。
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。
setId	プロバイダ ID を設定します。
setOptions	オプションを設定します。
setProviderType	プロバイダ・タイプを設定します。
toString	クラス Object の toString をオーバーライドします。
toXML	サービス・デプロイメント・ディスクリプタを XML として書き込みます。

コンストラクタ

ProviderDeploymentDescriptor

```
public 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。

メソッド

setId

```
public void setId(String id)
```

プロバイダ ID を設定します。

パラメータ

`id` ― 一意のプロバイダ ID

getId

```
public String getId()
```

このプロバイダの一意の ID を戻します。

戻り値

このプロバイダの一意の ID

setClassname

```
public void setClassname(String classname)
```

このプロバイダを実装するクラスの名前を設定します。

パラメータ

`classname` ― 実装するクラスの名前

getClassname

```
public String getClassname()
```

このプロバイダを実装するクラスの名前を戻します。

戻り値

クラスの名前

setProviderType

```
public void setProviderType(String providerType)
```

プロバイダ・タイプを設定します。

パラメータ

providerType ― プロバイダ・タイプ

getProviderType

```
public String getProviderType()
```

プロバイダ・タイプを戻します。

戻り値

このプロバイダのタイプ

setOptions

```
public void setOptions(Hashtable options)
```

オプションを設定します。

パラメータ

options ― このサービスに対するプロバイダ実装固有のオプションを表す名前 - 値のペア

getOptions

```
public Hashtable getOptions()
```

プロバイダ固有のオプションを取得します。

戻り値

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名前 - 値のペア

fromXML

```
public static ProviderDeploymentDescriptor fromXML(Element root)
```

指定したドキュメントからプロバイダ・ディスクリプタを作成します。

パラメータ

root − XML プロバイダ・ディスクリプタを表すドキュメントのルート

戻り値

指定した XML の ProviderDeploymentDescriptor

toXML

```
public void toXML(Writer pr)
```

サービス・デプロイメント・ディスクリプタを XML として書き込みます。

パラメータ

pr − XML 出力用のライター

toString

```
public String toString()
```

オーバーライド

クラス Object 内の toString

RequestContext クラス

oracle.soap.server.**RequestContext** クラス

```
java.lang.Object
   |
   +----oracle.soap.server.RequestContext

public class RequestContext
extends Object
```

RequestContext クラスは、SOAP リクエストの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が設定する必要がある情報も含まれます。プロバイダにはリクエスト・エンベロープが渡されるため、プロバイダはリクエスト・パラメータをアンマーシャリングする必要があることに注意してください。同様に、プロバイダは、レスポンスをマーシャリングする必要があります（ただし、プロバイダは、レスポンス・エンベロープも設定する必要があります）。これは、トランスポータブルなハンドラで必要になる場合があるためです。

SOAP エンジンはプロバイダに次の情報を提供します。つまり、プロバイダは、この情報を `Provider.invoke` で利用できます。

- `getEnvelope` ― リクエストが含まれるエンベロープ
- `getServiceDeploymentDescriptor` ― メソッドが起動し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` ― サービス内でメソッドを起動するユーザーを記述したセキュリティ・コンテキスト
- `getMethodName` ― サービス内で起動しているメソッドの名前

プロバイダは、次の情報を SOAP エンジンに渡す必要があります。

- `setResponseBytes` ― マーシャリングされたレスポンス。これは、レスポンスがあると、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で作成できます。
- `setResponseEnvelope` ― レスポンス・エンベロープ。レスポンス・バイトと論理的に等価です。
- `getRequestEncodingStyle` ― エラーが発生した場合にレスポンスで使用するエンコーディング・スタイル。設定されていない場合は、Constants.NS_URI_SOAP_ENC（SOAP エンコーディング）にデフォルト設定されます。プロバイダは、このデフォルト設定を使用しない場合、例外の発生に備えてできるかぎり早くこの値を設定する必要が

あります。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。

表 18-7 RequestContext のクラス・メンバー

メンバー	説明
RequestContext	デフォルトのコンストラクタです。
getMethodName	この SOAP リクエストのメソッド名を取得します。
getRequestEncodingStyle	リクエストで使用されたエンコーディング・スタイルを取得します。
getRequestEnvelope	実際の SOAP リクエストを表すエンベロープを取得します。
getResponseBytes	この SOAP リクエストのレスポンス・ストリームを取得します。
getResponseEnvelope	SOAP レスポンスを表すエンベロープを取得します。
getResponseMap	SOAP レスポンスのシリアル化に使用するマッピング・レジストリを取得します。
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・ディスクリプタを取得します。
getServiceId	この SOAP リクエストのサービス ID（URI）を取得します。
getUserContext	この SOAP リクエストのユーザー・コンテキストを取得します。
setMethodName	この SOAP リクエストのメソッド名を設定します。
setRequestEncodingStyle	リクエストで使用されたエンコーディング・スタイルを設定します。
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。
setResponseBytes	この SOAP リクエストのレスポンス・ストリームを設定します。
setResponseEnvelope	SOAP レスポンスを表すエンベロープを設定します。
setResponseMap	SOAP レスポンス・エンベロープのシリアル化に使用するマッピング・レジストリを設定します。
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。

表 18-7 RequestContext のクラス・メンバー（続き）

メンバー	説明
setServiceId	この SOAP リクエストのサービス ID（URI）を設定します。
setUserContext	この SOAP リクエストのユーザー・コンテキストを設定します。

コンストラクタ

RequestContext

```
public RequestContext()
```

このクラスのデフォルトのコンストラクタです。

メソッド

setRequestEnvelope

```
public void setRequestEnvelope(Envelope envelope)
```

実際の SOAP リクエストを表すエンベロープを設定します。

パラメータ

`envelope` ― SOAP エンベロープ

getRequestEnvelope

```
public Envelope getRequestEnvelope()
```

実際の SOAP リクエストを表すエンベロープを取得します。

戻り値

SOAP エンベロープ

setResponseEnvelope

```
public void setResponseEnvelope(Envelope envelope)
```

SOAP レスポンスを表すエンベロープを設定します。

パラメータ

envelope − SOAP レスポンス・エンベロープ

getResponseEnvelope

```
public Envelope getResponseEnvelope()
```

SOAP レスポンスを表すエンベロープを取得します。

戻り値

SOAP レスポンス・エンベロープ

setResponseMap

```
public void setResponseMap(SOAPMappingRegistry smr)
```

SOAP レスポンス・エンベロープのシリアル化に使用するマッピング・レジストリを設定します。

パラメータ

smr − SOAP レスポンス・エンベロープ用のマッピング・レジストリ

getResponseMap

```
public SOAPMappingRegistry getResponseMap()
```

SOAP レスポンスのシリアル化に使用するマッピング・レジストリを取得します。

戻り値

SOAP レスポンス・エンベロープ用のマッピング・レジストリ

setResponseBytes

```
public void setResponseBytes(ByteArrayOutputStream bytes)
```

この SOAP リクエストのレスポンス・ストリームを設定します。

パラメータ

`bytes` ― レスポンスが含まれる ByteArrayOutputStream

getResponseBytes

```
public ByteArrayOutputStream getResponseBytes()
```

この SOAP リクエストのレスポンス・ストリームを取得します。

戻り値

レスポンスが含まれる ByteArrayOutputStream

setRequestEncodingStyle

```
public void setRequestEncodingStyle(String requestEncodingStyle)
```

リクエストで使用されたエンコーディング・スタイルを設定します。

パラメータ

`requestEncodingStyle` ― リクエスト・エンコーディング・スタイル

getRequestEncodingStyle

```
public String getRequestEncodingStyle()
```

リクエストで使用されたエンコーディング・スタイルを取得します。

戻り値

リクエスト・エンコーディング・スタイル

setServiceDeploymentDescriptor

```
public void setServiceDeploymentDescriptor(ServiceDeploymentDescriptor
serviceDeploymentDescriptor)
```

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。

パラメータ

`serviceDeploymentDescriptor` ― このリクエストのサービス・デプロイメント・ディスクリプタ

getServiceDeploymentDescriptor

```
public ServiceDeploymentDescriptor 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・ディスクリプタを取得します。

戻り値

このリクエストのサービス・デプロイメント・ディスクリプタ、またはプロバイダが AutonomousProvider の場合は null

setMethodName

```
public void setMethodName(String methodName)
```

この SOAP リクエストのメソッド名を設定します。メソッド名はエンベロープ内に格納されていますが、サーバーはここでメソッド名をキャッシュできるので便利です。

パラメータ

`methodName` ― サービス内で起動しているメソッドの名前

getMethodName

```
public String getMethodName()
```

この SOAP リクエストのメソッド名を取得します。

戻り値

起動しているメソッドの名前

setServiceId

```
public void setServiceId(String serviceId)
```

この SOAP リクエストのサービス ID（URI）を設定します。

パラメータ

`serviceId` ― このリクエストが向けられるサービスの URI

getServiceId

```
public String getServiceId()
```

この SOAP リクエストのサービス ID（URI）を取得します。

戻り値

このリクエストが向けられるサービスの URI

setUserContext

```
public void setUserContext(UserContext userContext)
```

この SOAP リクエストのユーザー・コンテキストを設定します。

パラメータ

`userContext` ― ユーザー・コンテキスト

getUserContext

```
public UserContext getUserContext()
```

この SOAP リクエストのユーザー・コンテキストを取得します。

戻り値

ユーザー・コンテキスト

SOAPServerContext クラス

oracle.soap.server.**SOAPServerContext** クラス

```
java.lang.Object
   |
   +----oracle.soap.server.SOAPServerContext

public class SOAPServerContext
extends Object
```

SOAPServerContext クラスは、サーバーが実行されているコンテナのタイプに依存しない SOAP サーバーのコンテキストを定義します。

表 18-8 SOAPServerContext のクラス・メンバー

クラス・メンバー	説明
SOAPServerContext	デフォルトのコンストラクタです。
getAttribute	指定した名前の属性を戻します。指定した名前の属性がない場合は null を戻します。
getAttributeNames	SOAP のコンテキスト内で使用可能な属性名の一覧を戻します。
getGlobalContext	グローバルなコンテキストを戻します。
getLogger	SOAP のログ出力を戻します。
removeAttribute	指定した名前の属性をコンテキストから削除します。
setAttribute	SOAP のコンテキスト内で、オブジェクトを指定の属性名にバインドします。
setGlobalContext	SOAP サーバー全体のオブジェクトが含まれるグローバルなコンテキストを設定します。
setLogger	情報メッセージとデバッグ・メッセージのテキスト・ベースのロギングで使用するログ出力を設定します。

コンストラクタ

SOAPServerContext

```
public SOAPServerContext()
```

デフォルトのコンストラクタです。

メソッド

getGlobalContext

```
public Hashtable getGlobalContext()
```

グローバルなコンテキストを戻します。

戻り値

SOAP サーバー全体のオブジェクトが含まれるグローバルなコンテキスト、または属性が設定されていない場合は null

setGlobalContext

```
public void setGlobalContext(Hashtable globalContext)
```

SOAP サーバー全体のオブジェクトが含まれるグローバルなコンテキストを設定します。

パラメータ

`globalContext` − グローバルなコンテキスト

setLogger

```
public void setLogger(Logger logger)
```

情報メッセージとデバッグ・メッセージのテキスト・ベースのロギングで使用するログ出力を設定します。

パラメータ

`logger` − SOAP のログ出力

getLogger

```
public Logger getLogger()
```

SOAP のログ出力を戻します。

戻り値

情報メッセージとデバッグ・メッセージのログに使用する、SOAP のログ出力

getAttribute

```
public Object getAttribute(String name)
```

指定した名前の属性を戻します。指定した名前の属性がない場合は null を戻します。

パラメータ

name ― 属性の名前を指定する文字列

戻り値

属性の値が含まれるオブジェクト、または指定した名前と一致する属性が存在しない場合は null

getAttributeNames

```
public Enumeration getAttributeNames()
```

SOAP のコンテキスト内で使用可能な属性名の一覧を戻します。

戻り値

属性名の一覧

removeAttribute

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

指定した名前の属性をコンテキストから削除します。削除した後、属性の値を取得する getAttribute(java.lang.String) をコールすると、null が戻されます。

パラメータ

name ― 削除する属性の名前を指定する文字列

setAttribute

```
public void setAttribute(String name,
                         Object object)
```

SOAP のコンテキスト内で、オブジェクトを指定の属性名にバインドします。指定した名前がすでに属性で使用されている場合、このメソッドは、古い属性を削除し、新しい属性にその名前をバインドします。名前およびオブジェクトには null を設定できません。

パラメータ

name ― 属性の名前を指定する文字列（null 以外）

object ― バインドする属性を表すオブジェクト（null 以外）

ServiceDeploymentDescriptor クラス

oracle.soap.server.**ServiceDeploymentDescriptor** クラス

```
java.lang.Object
   |
   +----oracle.soap.server.ServiceDeploymentDescriptor

public final class ServiceDeploymentDescriptor
extends Object
implements Serializable
```

ServiceDeploymentDescriptor クラスは、プロバイダ・タイプに依存しない SOAP サービスの配置情報を定義します。このクラスは、指定した任意の数のプロバイダ・オプションをサポートします。これによって、新しいタイプのプロバイダについてコードを変更せずにディスクリプタを簡単に拡張できます。

フィールド

SERVICE_TYPE_RPC

```
public static final int SERVICE_TYPE_RPC
```

SERVICE_TYPE_MESSAGE

```
public static final int SERVICE_TYPE_MESSAGE
```

SCOPE_REQUEST

```
public static final int SCOPE_REQUEST
```

SCOPE_SESSION

```
public static final int SCOPE_SESSION
```

SCOPE_APPLICATION

```
public static final int SCOPE_APPLICATION
```

コンストラクタ

ServiceDeploymentDescriptor

```
public ServiceDeploymentDescriptor()
```

新しいサービス・ディスクリプタを構成します。

メソッド

setId

```
public void setId(String id)
```

サービス ID を設定します。これは、有効な URI であることが必要です。

パラメータ

id ― サービス URI

getId

```
public String getId()
```

サービス ID を取得します。

戻り値

サービス ID（URI）

setProviderId

```
public void setProviderId(String providerId)
```

このサービスのプロバイダの ID を設定します。

パラメータ

providerId ― このサービスのプロバイダの ID

getProviderId

```
public String getProviderId()
```

このサービスのプロバイダの ID を取得します。

戻り値

プロバイダ ID

setMethods

```
public void setMethods(String methods[])
```

このサービスで提供されるメソッドのリストを設定します。

パラメータ

methods ― 提供されるメソッドのリスト

getMethods

```
public String[] getMethods()
```

このサービスで提供されるメソッドのリストを取得します。

戻り値

提供されるメソッドのリスト

setScope

```
public void setScope(int scope)
```

実行範囲を設定します。

パラメータ

scope ― 実行範囲。定数 SCOPE_xxx の 1 つです。

getScope

```
public int getScope()
```

有効範囲を取得します。

戻り値

有効範囲。定数 SCOPE_xxx の 1 つです。

setServiceType

```
public void setServiceType(int serviceType)
```

サービス・タイプを設定します。DLD: RPC と一方向メッセージの違いを説明します。

パラメータ

`serviceType` ― サービス・タイプ。定数 SERVICE_TYPE_xxx の 1 つです。

getServiceType

```
public int getServiceType()
```

サービス・タイプを取得します。

戻り値

サービス・タイプ。定数 SERVICE_TYPE_xxx の 1 つです。

setProviderType

```
public void setProviderType(String providerType)
```

プロバイダ・タイプを設定します。

パラメータ

`providerType` ― プロバイダ・タイプ。文字列も設定できます。プロバイダ・タイプを使用して、プロバイダ固有のオプションに対して XML サービス・ディスクリプタの妥当性をチェックします。

getProviderType

```
public String getProviderType()
```

プロバイダ・タイプを取得します。

戻り値

プロバイダ・タイプ

setProviderOptions

```
public void setProviderOptions(Hashtable providerOptions)
```

プロバイダ固有のオプションを設定します。

パラメータ

`providerOptions` ― 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名前 - 値のペア

getProviderOptions

```
public Hashtable getProviderOptions()
```

プロバイダ固有のオプションを取得します。

戻り値

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名前 - 値のペア

setFaultListener

```
public void setFaultListener(String faultListener[])
```

障害リスナーのリストを設定します。

パラメータ

`faultListener` ― このサービスの障害リスナーであるクラスの名前リスト

getFaultListener

```
public String[] getFaultListener()
```

障害リスナーのリストを取得します。

戻り値

このサービスの障害リスナーであるクラスの名前リスト

buildFaultRouter

```
public SOAPFaultRouter buildFaultRouter()
```

障害ルーターを取得します。

戻り値

サービスの障害リスナーから作成される障害ルーター

setTypeMappings

```
public void setTypeMappings(TypeMapping typeMappings[])
```

XML から Java への非シリアル化、Java から XML へのシリアル化の方法を定義する、XML と Java 間のタイプ・マッピングを設定します。

パラメータ

typeMappings − タイプ・マッピング

getTypeMappings

```
public TypeMapping[] getTypeMappings()
```

XML から Java への非シリアル化、Java から XML へのシリアル化の方法を定義する、XML と Java 間のタイプ・マッピングを取得します。

戻り値

タイプ・マッピング

setSqlMap

```
public void setSqlMap(Hashtable sqlMap)
```

SQL タイプから Java タイプへのマップを設定します。

パラメータ

sqlMap − SQL タイプから Java タイプへのマップ

getSqlMap

```
public Hashtable getSqlMap()
```

SQL タイプから Java タイプへのマップを取得します。

戻り値

SQL タイプから Java タイプへのマップ

setDefaultSMRClass

```
public void setDefaultSMRClass(String defaultSMRClass)
```

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。

パラメータ

`defaultSMRClass` ― デフォルトの SOAP マッピング・レジストリ・クラス

getDefaultSMRClass

```
public String getDefaultSMRClass()
```

デフォルトの SOAP マッピング・レジストリ・クラスを取得します。

戻り値

デフォルトの SOAP マッピング・レジストリ・クラス

isMethodValid

```
public boolean isMethodValid(String methodName)
```

指定したメソッドがこのサービスに対して有効かどうかを判断します。

戻り値

メソッドがこのサービスに対して有効な場合は true、無効な場合は false

fromXML

```
public static ServiceDeploymentDescriptor fromXML(Element root)
```

指定したドキュメントの情報を ServiceDeploymentDescriptor に移入します。このドキュメントはディスクリプタを表す XML 文書です。

パラメータ

`root` ― サービス・ディスクリプタを表す XML 文書のルート

戻り値

ドキュメントの情報が含まれる ServiceDeploymentDescriptor

例外

`IllegalArgumentException` ― ドキュメントが無効な場合

toXML

```
public void toXML(Writer pr)
```

サービス・デプロイメント・ディスクリプタを XML として書き込みます。

パラメータ

`pr` ― XML 出力用のライター

toString

```
public String toString()
```

ディスクリプタの出力可能な表現を取得します。

オーバーライド

クラス Object 内の toString

buildSOAPMappingRegistry

```
public static SOAPMappingRegistry
buildSOAPMappingRegistry(ServiceDeploymentDescriptor sdd)
```

XML シリアル化レジストリを、デプロイメント・ディスクリプタに登録されたすべてのタイプ・マッピングから生成するユーティリティです。

パラメータ

sdd ― デプロイメント・ディスクリプタ

戻り値

XML シリアル化レジストリ

buildSqlClassMap

```
public static Hashtable buildSqlClassMap(ServiceDeploymentDescriptor sdd)
                            throws SOAPException
```

デプロイメント・ディスクリプタのタイプ・マッピング情報を使用して、SQL タイプから Java クラスへのマップを生成するユーティリティです。

パラメータ

sdd ― 使用するサービス・デプロイメント・ディスクリプタ

戻り値

タイプからクラスへのマップ

例外

SOAPException ― マップの生成に失敗した場合

UserContext クラス

oracle.soap.server.**UserContext** クラス

```
java.lang.Object
   |
   +----oracle.soap.server.UserContext

public class UserContext
extends Object
```

UserContext クラスは、SOAP サービス・リクエストのユーザー・コンテキストを定義します。一部の属性は事前定義されており、それらの属性を使用する set メソッドと get メソッドが用意されています。さらに、プロバイダは、getAttribute と setAttribute を使用して追加の属性を定義できます。

`HttpServlet` と `HttpSession` は実際にはこのクラスに属していませんが、JavaProvider には必要です。

コンストラクタ

UserContext

```
public UserContext()
```

デフォルトのコンストラクタです。

メソッド

getRequestURI

```
public String getRequestURI()
```

リクエストの URI を戻します。

戻り値

リクエストの URI

setRequestURI

```
public void setRequestURI(String uri)
```

リクエストの URI を設定します。

パラメータ

`uri` − リクエストの URI

getCertificate

```
public Object getCertificate()
```

ユーザー証明書を戻します。

戻り値

SOAP リクエストを行うユーザーのユーザー証明書、またはこの属性が設定されていない場合は null

setCertificate

```
public void setCertificate(Object certificate)
```

ユーザー証明書を設定します。

パラメータ

`certificate` − SOAP リクエストを行うユーザーのユーザー証明書

getHttpServlet

```
public HttpServlet getHttpServlet()
```

HTTP サーブレットを戻します

戻り値

SOAP リクエストを処理している HttpServlet、またはサーブレットの属性が設定されていない場合は null

setHttpServlet

```
public void setHttpServlet(HttpServlet servlet)
```

HTTP サーブレットを設定します

パラメータ

`servlet` − SOAP リクエストを処理する HttpServlet

getHttpSession

```
public HttpSession getHttpSession()
```

HTTP セッションを戻します

戻り値

SOAP リクエストの `HttpSession`、またはセッションの属性が設定されていない場合は null

setHttpSession

```
public void setHttpSession(HttpSession session)
```

HTTP セッションを設定します

パラメータ

`servlet` − SOAP リクエストの HttpSession

getRemoteAddress

```
public String getRemoteAddress()
```

リクエストを送信するクライアントのインターネット・プロトコル（IP）アドレスを戻します。

戻り値

リモート・クライアントの IP アドレス

setRemoteAddress

```
public void setRemoteAddress(String remoteAddress)
```

クライアントのリモート IP アドレスを設定します。

パラメータ

`remoteAddress` − SOAP リクエストを行うクライアントの IP アドレス

getRemoteHost

```
public String getRemoteHost()
```

リクエストを送信するクライアントのホスト名を戻します。

戻り値

リモート・クライアントのホスト名

setRemoteHost

```
public void setRemoteHost(String remoteHost)
```

SOAP リクエストを行うクライアントのホスト名を設定します。

パラメータ

`remoteHost` − SOAP リクエストを行うクライアントのホスト名

getSecureChannel

```
public boolean getSecureChannel()
```

チャネルが保護されているかどうかを戻します。

戻り値

チャネルが保護されている場合は `true`、または保護されていない場合は `false`

setSecureChannel

```
public void setSecureChannel(boolean secureChannel)
```

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。

パラメータ

`secureChannel` − チャネルが保護されている場合は true、または保護されていない場合は false

getUsername

```
public String getUsername()
```

プロトコル固有のユーザー名を戻します。

戻り値

SOAP リクエストのプロトコル固有のユーザー名、またはこの属性が設定されていない場合は null

setUsername

```
public void setUsername(String username)
```

プロトコル固有のユーザー名を設定します。

パラメータ

`username` − SOAP リクエストのプロトコル固有のユーザー名

getAttribute

```
public Object getAttribute(String name)
```

指定した名前の属性を戻します。指定した名前の属性がない場合は null を戻します。

パラメータ

`name` − 属性の名前を指定する文字列

戻り値

属性の値が含まれるオブジェクト、または指定した名前と一致する属性が存在しない場合は null

関連項目

`getAttributeNames`

getAttributeNames

```
public Enumeration getAttributeNames()
```

SOAP のコンテキスト内で使用可能な属性名の一覧を戻します。

戻り値

属性名の一覧

関連項目

`getAttribute`

setAttribute

```
public void setAttribute(String name,
                         Object object)
```

SOAP のコンテキスト内で、オブジェクトを指定の属性名にバインドします。指定した名前がすでに属性で使用されている場合、このメソッドは、古い属性を削除し、新しい属性にその名前をバインドします。名前およびオブジェクトには null を設定できません。

パラメータ

`name` ― 属性の名前を指定する文字列（null 以外）

`object` ― バインドする属性を表すオブジェクト（null 以外）

removeAttribute

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

指定した名前の属性をコンテキストから削除します。削除した後、属性の値を取得する getAttribute(java.lang.String) をコールすると、null が戻されます。

パラメータ

`name` ― 削除する属性の名前を指定する文字列

19

パッケージ oracle.soap.transport

この章では、パッケージ oracle.soap.transport について説明します。このパッケージには、XDK for Java で Oracle SOAP をサポートするクラスが含まれています。

Oracle SOAP は、Simple Object Access Protocol の実装です。Oracle SOAP は、Apache Software Foundation が開発した SOAP オープン・ソース実装に基づいています。

この章は、次の項で構成されています。

パッケージ oracle.soap.transport の説明

Simple Object Access Protocol（SOAP）は、インターネット間でリクエストとレスポンスの送受信を行うトランスポート・プロトコルです。これは、XML と HTTP に基づいています。

SOAP は、トランスポート・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。すべてのアプリケーションで使用できる標準の XML メッセージ・フォーマットを提供します。SOAP では、World Wide Web Consortium（W3C）の XML Schema 規格を使用します。

関連項目：

- `http://www.w3.org/TR/SOAP/`
- `http://xml.apache.org/soap`
- 『Oracle9*i* XML Developer's Kit ガイド - XDK』

パッケージ oracle.soap.transport の概要

パッケージ oracle.soap.transport には、oracle.soap.transport.**OracleSOAPTransport** インタフェースが含まれています。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。

表 19-1　OracleSOAPTransport インタフェースのメソッド

メソッド	説明
getProperties	接続プロパティを取得します。
setProperties	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、クリーン・アップを実行します。

OracleSOAPTransport インタフェース

oracle.soap.transport.**OracleSOAPTransport** インタフェース

```
public interface OracleSOAPTransport
extends SOAPTransport
```

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。

getProperties

```
public abstract Properties getProperties()
```

接続プロパティを取得します。

戻り値

接続プロパティ

setProperties

```
public abstract void setProperties(Properties prop)
```

接続プロパティを設定します。

パラメータ

prop ― 接続プロパティ

close

```
public abstract void close()
```

トランスポートをクローズし、クリーン・アップを実行します。

20

パッケージ oracle.soap.transport.http

この章では、パッケージ oracle.soap.transport.http について説明します。このパッケージには、XDK for Java で Oracle SOAP をサポートするクラスが含まれています。

Oracle SOAP は、Simple Object Access Protocol の実装です。Oracle SOAP は、Apache Software Foundation が開発した SOAP オープン・ソース実装に基づいています。

この章は、次の項で構成されています。

パッケージ oracle.soap.transport.http の説明

Simple Object Access Protocol（SOAP）は、インターネット間でリクエストとレスポンスの送受信を行うトランスポート・プロトコルです。これは、XML と HTTP に基づいています。

SOAP は、トランスポート・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。すべてのアプリケーションで使用できる標準の XML メッセージ・フォーマットを提供します。SOAP では、World Wide Web Consortium（W3C）の XML Schema 規格を使用します。

パッケージ oracle.soap.transport.http には、`OracleSOAPTransport` を実装する `OracleSOAPHTTPConnection` クラスが含まれています。

Oracle Soap Client API は、トランスポータブルなトランスポートをサポートします。これによって、クライアントはトランスポートを簡単に変更できます。HTTP および HTTPS（保護されている HTTP）も使用可能なトランスポートです。

関連項目：

- `http://www.w3.org/TR/SOAP/`
- `http://xml.apache.org/soap`
- 『Oracle9*i* XML Developer's Kit ガイド - XDK』

パッケージ oracle.soap.transport.http の概要

パッケージ oracle.soap.transport.http には、oracle.soap.transport.http.OracleSOAPHTTPConnection クラスが含まれています。このクラスは、`OracleSOAPTransport` 内の Oracle 固有のトランスポート拡張機能を実装します。

表 20-1　OracleSOAPHTTPConnection クラスのフィールド

フィールド	説明
ALLOW_USER_INTERACTION	ユーザーとの対話を設定します。
AUTH_TYPE	HTTP の認証タイプ（Basic/Digest）を定義します。
CIPHERS	HTTPS で使用する Cipher Suite（Cipher Suite のコロン区切りリスト）を定義します。
PASSWORD	HTTP のパスワードを定義します。
PROXY_AUTH_TYPE	プロキシの認証タイプ（Basic/Digest）を定義します。
PROXY_HOST	プロキシのホストを定義します。
PROXY_PASSWORD	プロキシのパスワードを定義します。
PROXY_PORT	プロキシのポートを定義します。
PROXY_USERNAME	プロキシのユーザー名を定義します。
STATUS_LINE	HTTP ヘッダーから HTTP ステータス行を取得する（getHeaders）場合に使用します。
USERNAME	HTTP のユーザー名を定義します。
WALLET_LOCATION	HTTPS で使用する Wallet 位置を定義します。
WALLET_PASSWORD	HTTPS で使用するパスワードを定義します。

表 20-2　OracleSOAPHTTPConnection クラスのコンストラクタとメソッド

メンバー	説明
コンストラクタ	--
OracleSOAPHTTPConnection	プロパティを引数として取得するコンストラクタです。
メソッド	--
getProperties	接続プロパティを取得します。

表 20-2 OracleSOAPHTTPConnection クラスのコンストラクタとメソッド（続き）

メンバー	説明
close	トランスポートをクローズし、クリーン・アップを実行します。
finalize	クラス Object の finalize をオーバーライドします。
getHeaders	プロトコルで生成されたヘッダーへのアクセスを戻します。
getProperties	接続プロパティを取得します。
receive	バッファされている Reader を戻し、x に送信した内容に対するレスポンスを再受信します。
send	エンベロープの指定の URL への転送をリクエストします。
setProperties	接続プロパティを設定します。

OracleSOAPHTTPConnection クラス

oracle.soap.transport.http.**OracleSOAPHTTPConnection**

```
java.lang.Object
   |
   +----oracle.soap.transport.http.OracleSOAPHTTPConnection

public class OracleSOAPHTTPConnection
extends Object
```

OracleSOAPTransport を実装します。

フィールド

ALLOW_USER_INTERACTION

```
public static final String ALLOW_USER_INTERACTION
```

ユーザーとの対話を設定するプロパティ

STATUS_LINE

```
public static final String STATUS_LINE
```

HTTP ヘッダーから HTTP ステータス行を取得する（getHeaders）場合に使用するプロパティ

PROXY_HOST

```
public static final String PROXY_HOST
```

プロキシのホストを定義するために使用するプロパティ

PROXY_PORT

```
public static final String PROXY_PORT
```

プロキシのポートを定義するために使用するプロパティ

PROXY_AUTH_TYPE

```
public static final String PROXY_AUTH_TYPE
```

プロキシの認証タイプ（Basic/Digest）を定義するために使用するプロパティ

PROXY_USERNAME

```
public static final String PROXY_USERNAME
```

プロキシのユーザー名を定義するために使用するプロパティ

PROXY_PASSWORD

```
public static final String PROXY_PASSWORD
```

プロキシのパスワードを定義するために使用するプロパティ

AUTH_TYPE

```
public static final String AUTH_TYPE
```

HTTP の認証タイプ（Basic/Digest）を定義するために使用するプロパティ

USERNAME

```
public static final String USERNAME
```

HTTP のユーザー名を定義するために使用するプロパティ

PASSWORD

```
public static final String PASSWORD
```

HTTP のパスワードを定義するために使用するプロパティ

WALLET_LOCATION

```
public static final String WALLET_LOCATION
```

HTTPS で使用する Wallet 位置を定義するために使用するプロパティ

WALLET_PASSWORD

```
public static final String WALLET_PASSWORD
```

HTTPS で使用する Wallet のパスワードを定義するために使用するプロパティ

CIPHERS

```
public static final String CIPHERS
```

HTTPS で使用する Cipher Suite（Cipher Suite のコロン区切りリスト）を定義するために使用するプロパティ

コンストラクタ

OracleSOAPHTTPConnection

```
public OracleSOAPHTTPConnection(Properties prop)
```

プロパティを引数として取得するコンストラクタです。

パラメータ

prop ― 接続プロパティ

メソッド

setProperties

```
public void setProperties(Properties prop)
```

接続プロパティを設定します。

パラメータ

prop ― 接続プロパティ

getProperties

```
public Properties getProperties()
```

接続プロパティを取得します。

戻り値

接続プロパティ

send

```
public void send(URL sendTo,
                 String action,
                 Hashtable headers,
                 Envelope env,
                 SOAPMappingRegistry smr,
                 int timeout) throws SOAPException
```

このメソッドを使用して、エンベロープを指定の URL に転送するようにリクエストします。レスポンス（存在する場合）は、receive() 関数をコールして取得する必要があります。SOAP クライアントはこのメソッドを直接使用できませんが、かわりに org.apache.soap.rpc.Call を使用してください。

パラメータ

sendTo ― エンベロープの送信先の URL

action ― SOAPAction ヘッダー・フィールドの値

headers ― プロトコル・ヘッダーとして送信するその他のヘッダー・フィールド

env ― 送信するエンベロープ

smr ― （渡される）XML<->Java 間のタイプ・マッピング・レジストリ

timeout ―タイムアウト

例外

SOAPException ― 問題が発生した場合。該当する理由コードが付きます。

receive

```
public BufferedReader receive()
```

バッファされている Reader を戻し、送信した内容に対するレスポンスを再受信します。SOAP クライアントはこのメソッドを直接使用できませんが、かわりに org.apache.soap.rpc.Call を使用してください。

戻り値

結果を読み込む Reader、または結果が読み込めない場合は null

getHeaders

```
public Hashtable getHeaders()
```

プロトコルで生成されたヘッダーへのアクセスを戻します。SOAP クライアントはこのメソッドを直接使用できませんが、かわりに org.apache.soap.rpc.Call を使用してください。

戻り値

すべてのヘッダーが含まれるハッシュテーブル

close

```
public void close()
```

接続をクローズします。このメソッドをコールすると、receive メソッドによって戻された BufferedReader もクローズされて使用できなくなります。このメソッドをコールすると、ガベージ・コレクションを実行せずに、リソースが解放されます。

finalize

```
public void finalize()
             throws Throwable
```

オーバーライド

クラス Object 内の finalize

21

パッケージ oracle.soap.util.xml

この章では、パッケージ oracle.soap.util.xml について説明します。このパッケージには、XDK for Java で Oracle SOAP をサポートするクラスが含まれています。

Oracle SOAP は、Simple Object Access Protocol の実装です。Oracle SOAP は、Apache Software Foundation が開発した SOAP オープン・ソース実装に基づいています。

この章は、次の項で構成されています。

パッケージ oracle.soap.util.xml の説明

Simple Object Access Protocol（SOAP）は、インターネット間でリクエストとレスポンスの送受信を行うトランスポート・プロトコルです。これは、XML と HTTP に基づいています。Oracle SOAP は、Apache Software Foundation が開発した SOAP オープン・ソース実装に基づいています。

SOAP は、トランスポート・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。すべてのアプリケーションで使用できる標準の XML メッセージ・フォーマットを提供します。SOAP では、World Wide Web Consortium（W3C）の XML Schema 規格を使用します。

パッケージ oracle.soap.util.xml には、`XmlUtils` クラスが含まれています。このクラスは、SOAP クライアント用の API を実装し、SOAP サービスのリクエストを構成して SOAP レスポンスを処理する XML 文書を生成します。Oracle SOAP は、有効な SOAP リクエストを送信するクライアントからのリクエストを処理します。

関連項目：

- `http://www.w3.org/TR/SOAP/`
- `http://xml.apache.org/soap`
- 『Oracle9*i* XML Developer's Kit ガイド - XDK』

パッケージ oracle.soap.util.xml の概要

パッケージ oracle.soap.util.xml には、oracle.soap.util.xml.OracleSOAPHTTPConnection クラスが含まれています。このクラスは、`OracleSOAPTransport` 内の Oracle 固有のトランスポート拡張機能を実装します。

表 21-1　XmlUtils クラスのメンバー

メンバー	説明
コンストラクタ	--
XmlUtils	デフォルトのコンストラクタです。
メソッド	--
extractServiceId	エンベロープからサービス ID を取得します。
extractMethodName	エンベロープからメソッド名を取得します。
parseXml(String)	指定した XML ファイルを解析し、XML 文書を戻します。
parseXml(Reader)	指定した XML ソースを解析し、XML 文書を戻します。
parseXml(InputStream)	XML InputStream の内容を解析し、XML 文書を戻します。
createDocument	Document を作成します。

XmlUtils クラス

oracle.soap.util.xml.**XmlUtils** クラス

```
java.lang.Object
   |
   +----oracle.soap.util.xml.XmlUtils

public class XmlUtils
```

Object を拡張します。

コンストラクタ

XmlUtils

```
public XmlUtils()
```

デフォルトのコンストラクタです。

メソッド

extractServiceId

```
public static String extractServiceId(Envelope envelope) throws SOAPException
```

エンベロープからサービス ID を取得します。これは、最初の本体エントリの名前空間 URI です。

例外

SOAPException ― エンベロープからサービス URI を取得できない場合

extractMethodName

```
public static String extractMethodName(Envelope envelope) throws SOAPException
```

エンベロープからメソッド名を取得します。これは、最初の本体エントリの名前です。

例外

SOAPException ― エンベロープからメソッド名を取得できない場合

parseXml

```
public static Document parseXml(String filename)
                          throws SOAPException
```

指定した XML ファイルを解析し、XML 文書を戻します。

パラメータ

`filename` − XML ファイルへのフルパス

例外

`SOAPException` − ファイルが検出されない場合、または解析エラーの場合

parseXml

```
public static Document parseXml(Reader reader)
                           throws SOAPException
```

指定した XML ソースを解析し、XML 文書を戻します。

パラメータ

`reader` − XML 用の Reader

例外

`SOAPException` − ファイルが検出されない場合、または解析エラーの場合

parseXml

```
public static Document parseXml(InputStream is)
                           throws SOAPException
```

XML `InputStream` の内容を解析し、XML 文書を戻します。

パラメータ

`is` − 入力ストリーム・ソース

例外

`SOAPException` − 解析エラーまたは I/O エラーが発生した場合

createDocument

```
public static Document createDocument()
                        throws SOAPException
```

Document を作成します。

例外

SOAPException ― Document を作成できない場合

第 V 部

Oracle XML DB 用の Java パッケージ

第 V 部では、ドキュメント・オブジェクト・モデル（DOM）と Oracle XML DB の Service Provider Interface（SPI）を実装する Java パッケージについて説明します。

第 V 部は、次の章で構成されています。

- 第 22 章「パッケージ oracle.xdb.dom」
- 第 23 章「パッケージ oracle.xdb.spi」

関連項目： Oracle XML DB の機能を使用したアプリケーションの開発方法については、次のマニュアルを参照してください。

- 『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』
- 『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』

22

パッケージ oracle.xdb.dom

この章で説明するクラスはパッケージ oracle.xdb.dom に含まれており、XMLType の Java DOM API を実装します。

この章は、次の項で構成されています。

パッケージ oracle.xdb.dom の説明

この章で説明するクラスはパッケージ oracle.xdb.dom に含まれており、XMLType の Java DOM API を実装します。Oracle XML DB は W3C DOM 勧告の規定に従って、ドキュメント・オブジェクト・モデル（DOM）をサポートします。DOM の詳細は、http://www.w3.org/DOM/ を参照してください。

> **関連項目：** Oracle XML DB の機能を使用したアプリケーションの開発方法については、『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』を参照してください。

W3C DOM 勧告の実装の他に、Oracle XML DB 用の DOM API には、Oracle 固有の拡張機能が用意されています。

パッケージ oracle.xdb.dom クラスの概要

表 22-1 oracle.xdb.dom のクラス

クラス	説明
XDBAttribute クラス	XOB とのインタフェースで使用する W3C の DOM Node インタフェースを実装します。
XDBCData クラス	W3C の Text インタフェースを実装します。
XDBCharData クラス	W3C の CharacterData インタフェースを実装します。
XDBComment クラス	W3C の Comment インタフェースを実装します。
XDBDocument クラス	W3C の Document インタフェースを実装します。
XDBDomImplementation クラス	サーバーへの JDBC 接続をオープンする W3C の DomImplementation インタフェースを実装します。
XDBElement クラス	W3C の Element インタフェースを実装します。
XDBNamedNodeMap クラス	W3C の Named Node Map インタフェースを実装します。
XDBNode クラス	W3C の Node インタフェースを実装します。
XDBNodeList クラス	W3C の Node List インタフェースを実装します。
XDBProcInst クラス	W3C の W3C DOM ProcessingInstruction インタフェースを実装します。
XDBText クラス	W3C の Text インタフェースを実装します。
XMLType クラス	システム固有のデータ型 XMLType を Oracle XML DB で実装します。

表 22-2 org.w3c.dom 実装にマップされる oracle.xdb.dom のクラス

クラス	実装
XDBAttribute クラス	org.w3c.dom.Attribute
XDBCData クラス	org.w3c.dom.CData
XDBCharData クラス	org.w3c.dom.CharData
XDBComment クラス	org.w3c.dom.Comment
XDBDocument クラス	org.w3c.dom.Document
XDBDomImplementation クラス	org.w3c.dom.DomImplementation
XDBElement クラス	org.w3c.dom.Element
XDBNamedNodeMap クラス	org.w3c.dom.NamedNodeMap
XDBNode クラス	org.w3c.dom.Node
XDBNodeList クラス	org.w3c.dom.NodeList
XDBProcInst クラス	org.w3c.dom.ProcInst
XDBText クラス	org.w3c.dom.Text
XMLType クラス	該当なし。Oracle 固有のクラスです。

XDBAttribute クラス

構文

```
public class XDBAttribute
```

oracle.xdb.dom.XDBAttribute

説明

このクラスは org.w3c.dom.Attribute（XOB とのインタフェースで使用する W3C の DOM Node インタフェース）を実装します。

XDBCData クラス

構文

```
public class XDBCData
```

oracle.xdb.dom.XDBCData

説明

このクラスは org.w3c.dom.CData（W3C の Text インタフェース）を実装します。

XDBCharData クラス

構文

```
public class XDBCharData
```

oracle.xdb.dom.XDBCharData

説明

このクラスは org.w3c.dom.CharData（W3C の CharacterData インタフェース）を実装します。

XDBComment クラス

構文

```
public class XDBComment
```

oracle.xdb.dom.XDBComment

説明

このクラスは org.w3c.dom.Comment インタフェースを実装します。

XDBDocument クラス

構文

```
public class XDBDocument
```

oracle.xdb.dom.XDBDocument

説明

このクラスは org.w3c.dom.Document インタフェースを実装します。

メソッド

XDBDocument()

構文

```
public  XDBDocument()
```

説明

新しいドキュメントを作成します。使用できるのはサーバー内のみです。

XDBDocument(byte[ ])

構文

```
public  XDBDocument(byte[] source)
```

説明

ドキュメントをソースから移入します。使用できるのはサーバー内のみです。

パラメータ

source − XML テキストを含んだバイト

XDBDocument(Connection)

構文

```
public  XDBDocument(java.sql.Connection conn)
```

説明

ドキュメント・ソースのキャッシュに使用する接続をオープンします。

パラメータ

conn ― ドキュメントに使用する接続

XDBDocument(Connection, byte[ ])

構文

```
public  XDBDocument(java.sql.Connection conn, byte[] source)
```

説明

ドキュメント・ソースのバイトのキャッシュに使用する接続。

パラメータ

conn ― ドキュメントに使用する接続

source ― XML テキストを含んだバイト

XDBDocument(Connection, String)

構文

```
public  XDBDocument(java.sql.Connection conn, java.lang.String source)
```

説明

XML テキストを含んだ文字列のキャッシュに使用する接続をオープンします。

パラメータ

source ― XML テキストを含んだ文字列

conn ― 文字列に使用する接続

XDBDocument(String)

構文

```
public  XDBDocument(java.lang.String source)
```

説明

XML テキストを含んだ文字列。使用できるのはサーバー内のみです。

XDBDomImplementation クラス

構文

```
public class XDBDomImplementation
```

oracle.xdb.dom.XDBDomImplementation

説明

このクラスは org.w3c.dom.DomImplementation を実装します。

メソッド

XDBDomImplementation()

構文

```
public  XDBDomImplementation()
```

説明

サーバーへの JDBC 接続をオープンします。

XDBElement クラス

構文

```
public class XDBElement
```

oracle.xdb.dom.XDBElement

説明

このクラスは org.w3c.dom.Element を実装します。

XDBNamedNodeMap クラス

構文

```
public class XDBNamedNodeMap
```

oracle.xdb.dom.XDBNamedNodeMap

説明

このクラスは org.w3c.dom.NamedNodeMap を実装します。

XDBNode クラス

構文

```
public abstract class XDBNode
```

oracle.xdb.dom.XDBNode

説明

このクラスは org.w3c.dom.Node（XOB とのインタフェースで使用する W3C の DOM Node インタフェース）を実装します。

メソッド

write(OutputStream, String, short)

説明

このノード（およびすべてのサブノード）の XML を OutputStream に書き込みます。OutputStream が ServletOutputStream の場合、サーブレットの出力はコミットされ、システム固有のストリーム機能を使用してデータが書き込まれます。

構文

```
public void write(java.io.OutputStream s, java.lang.String charEncoding, short
indent)
```

パラメータ

s − 出力の書込み先ストリーム

charEncoding − IANA 文字コード（「ISO-8859」など）

indent − ネストされた要素のインデントに使用する文字数

XDBNodeList クラス

構文

```
public class XDBNodeList
```

oracle.xdb.dom.XDBNodeList

説明

このクラスは org.w3c.dom.NodeList を実装します。

XDBProcInst クラス

構文

```
public class XDBProcInst
```

oracle.xdb.dom.XDBProcInst

説明

このクラスは org.w3c.dom.ProcInst（W3C の DOM ProcessingInstruction インタフェース）を実装します。

XDBText クラス

構文

```
public class XDBText
```

oracle.xdb.dom.XDBText

説明

このクラスは org.w3c.dom.Text を実装します。

XMLType クラス

```
oracle.xdb.XMLType

public class XMLType
```

説明

XMLType クラスは、システム固有のデータ型 XMLType を Oracle XML DB で実装し、サーバー内での XML の格納と操作をサポートします。XMLType では、構造化された XML やキャラクタ・ラージ・オブジェクト（CLOB）などの複数の記憶領域オプションが使用できます。

構造化されたシステム固有の XML 記憶域は、実際のオブジェクト・リレーショナル構造（Oracle によって自動的に作成および管理されます）への XML の一種の分解です。この記憶域によって、SQL 問合せの効率が向上します。CLOB 記憶域は分解されない記憶域で、空白も含めて元の XML のイメージを保持しています。

フィールド

表 22-3 XMLType クラスのフィールドの概要

フィールド	説明
SQL_TYPECODE	静的な int 型
SQL_TYPENAME	静的な java.lang.String

_SQL_TYPECODE

```
public static final int _SQL_TYPECODE
```

_SQL_TYPENAME

```
public static final java.lang.String _SQL_TYPENAME
```

コンストラクタ

XMLType

```
public XMLType()
      throws java.sql.SQLException
```

メソッド

表 22-4　XMLType のメソッドの概要

メソッド	説明
getORADataFactory	この XMLType の ORAData ファクトリを取得します。
toDatum	XMLType データからイメージを構成します。
createXML	接続（conn）、データ型およびオブジェクトを指定して、XMLType を作成します。
getStringVal	XML データを含んだ文字列値を XMLType から取得します。
getClobVal	XML データを含んだ CLOB 値を XMLType から取得します。
extract	指定したノードのセットを XMLType から抽出する関数です。
existsNode	指定したノードのセットが XMLType 内にあることをチェックする関数です。
transform	指定の XSL ドキュメントを使用して XMLType を変換する関数です。
isFragment	XMLType が通常のドキュメントか、またはドキュメント・フラグメントであるかをチェックする関数です。
isSchemaValid	XMLType がスキーマ・ベースであるかどうかをチェックする関数です。
getDOM	XMLType に関連付けられている DOM 文書を取得します。
getBytesValue	XML データを含んだバイト値を XMLType から取得します。

getORADataFactory

```
public static oracle.sql.ORADataFactory getORADataFactory()
```

この XMLType の ORAData ファクトリを取得します。これは ORAData インタフェースの一部として必須です。

戻り値

この XMLType に関連付けられている ORADataFactory

toDatum

```
public oracle.sql.Datum toDatum(java.sql.Connection conn)
                         throws java.sql.SQLException
```

XMLType データからイメージを構成します。この関数は XMLType バイトを処理し、1 つのデータとして戻します。

パラメータ

conn − データ作成に使用する接続

戻り値

処理されたイメージ

例外

java.sql.SQLException

createXML

```
public static XMLType createXML(oracle.sql.OPAQUE opq)
                         throws java.sql.SQLException
```

XMLType バイトを含んだ不透明な型を指定して、XMLType を作成します。

パラメータ

opq − XMLType 生成の元となる不透明なデータ・オブジェクト

戻り値

作成された XMLType

例外

java.sql.SQLException -

createXML

```
public static XMLType createXML(java.sql.Connection conn,
                                java.lang.String xmlval)
                         throws java.sql.SQLException
```

XML データを含んだ文字列を指定して、XMLType を作成します。

パラメータ

conn − 使用する接続オブジェクト

xmlval − XML データを含んだ文字列

戻り値
作成された XMLType

例外
java.sql.SQLException

createXML

```
public static XMLType createXML(java.sql.Connection conn,
                                oracle.sql.CLOB xmlval)
                         throws java.sql.SQLException
```
XML データを含んだ CLOB を指定して、XMLType を作成します。

パラメータ
conn − 使用する接続オブジェクト

xmlval − XML データを含んだ CLOB

戻り値
作成された XMLType

例外
java.sql.SQLException

getStringVal

```
public java.lang.String getStringVal()
                               throws java.sql.SQLException
```
XML データを含んだ文字列値を XMLType から取得します。

戻り値
XML データ・バイトを含んだ文字列

例外
java.sql.SQLException

getClobVal

```
public oracle.sql.CLOB getClobVal()
                           throws java.sql.SQLException
```

XML データを含んだ CLOB 値を XMLType から取得します。

戻り値

XML データ・バイトを含んだ CLOB

例外

java.sql.SQLException -

createXML

```
public static XMLType createXML(java.sql.Connection conn,
                                org.w3c.dom.Document domdoc)
                         throws java.sql.SQLException
```

DOM 文書のインスタンスを指定して、XMLType を作成します。

パラメータ

domdoc — DOM ツリーを表す DOM 文書

戻り値

構成された XMLType

例外

java.sql.SQLException -

extract

```
public XMLType extract(java.lang.String xpath,
                       java.lang.String nsmap)
                 throws java.sql.SQLException
```

指定したノードのセットを XMLType から抽出する関数です。このノードのセットは XPath 式によって指定されます。元の XMLType は変更されず、そのままです。Thick の場合のみ機能します。

パラメータ

xpath − 検索するノードを指定する XPath 式

nsmap − XPath 式内の接頭辞を解決する名前空間のマップ。書式は「xmlns=a.com xmlns:b=b.com」です。

戻り値

抽出したノードを含んだ XMLType。指定の式と一致するノードがない場合は null です。

existsNode

```
public boolean existsNode(java.lang.String xpath,
                          java.lang.String nsmap)
                   throws java.sql.SQLException
```

指定したノードのセットが XMLType 内にあることをチェックする関数です。このノードのセットは XPath 式によって指定されます。

パラメータ

xpath −検索するノードを指定する XPath 式

nsmap − XPath 式内の接頭辞を解決する名前空間のマップ。書式は「xmlns=a.com xmlns:b=b.com」です。

戻り値

指定のノードが XMLType にある場合は true、ない場合は false

transform

```
public XMLType transform(XMLType xsldoc,
                         java.lang.String parammap)
                  throws java.sql.SQLException
```

指定の XSL ドキュメントを使用して XMLType を変換する関数です。新しい（変換された）XML 文書が戻されます。

パラメータ

xsldoc − XMLType に適用する XSL ドキュメント

parammap − XSL 変換に渡す最上位レベルのパラメータ。必要な書式は「a=b c=d e=f」です。null も可能です。

戻り値

変換された XMLType

isFragment

```
public boolean isFragment()
                    throws java.sql.SQLException
```

XMLType が通常のドキュメントか、またはドキュメント・フラグメントであるかをチェックする関数です。

戻り値

ドキュメントがフラグメントの場合は true、そうでない場合は false

isSchemaValid

```
public boolean isSchemaValid(java.lang.String schurl,
                             java.lang.String elname)
                     throws java.sql.SQLException
```

XMLType がスキーマ・ベースであるかどうかをチェックする関数です。

パラメータ

schurl − 検証対象のスキーマの URL。null の場合は、ドキュメント自体のスキーマ URL が使用されます（存在している場合）。

戻り値

ドキュメントがスキーマ・ベースの場合は true、そうでない場合は false

getDOM

```
public org.w3c.dom.Document getDOM()
                           throws java.sql.SQLException
```

XMLType に関連付けられている DOM 文書を取得します。このドキュメントが org.w3c.dom.Document です。コール元では、このドキュメントに関するすべての DOM 操作を実行できます。ドキュメントがバイナリ・ドキュメントの場合、この getDOM 関数は null を戻します。

戻り値

XMLType に関連付けられている DOM 文書オブジェクト

getBytesValue

```
public byte[] getBytesValue()
                      throws java.sql.SQLException
```

XML データを含んだバイト値を XMLType から取得します。

23

パッケージ oracle.xdb.spi

この章では、パッケージ oracle.xdb.spi について説明します。oracle.xdb.spi に含まれているクラスは、サービス・プロバイダ・ドライバを実装します。これらのドライバは、JNDI インタフェースと JDBC インタフェースへの共通アクセスをアプリケーションに提供します。

この章は、次の項で構成されています。

パッケージ oracle.xdb.spi の説明

oracle.xdb.spi に含まれているクラスは、JNDI API と JNDI のサービス・プロバイダ・ドライバを実装します。これらのドライバは、ファイル・システムと標準の Web プロトコルへのアクセスをアプリケーションに提供します。

- JNDI（Java Naming and Directory Interface）は、Java プログラムを DNS、LDAP、NDS などのネーミング・サービスおよびディレトリ・サービスに接続する API です。
- JNDI SPI（Service Provider Interface）は、ディレトリ・ドライバを書き込むための API です。
- JDBC（Java DataBase Connectivity）は、Java アプリケーションが SQL 言語を使用してデータベースにアクセスできるようにするプログラム・インタフェースです。JDBC は、マイクロソフトの ODBC の Java 対応インタフェースです。

アプリケーションは JNDI API に、ディレクトリ・ドライバは JNDI SPI（Service Provider Interface）に記述されています。パッケージ oracle.xdb.spi には、パブリック標準に対する Oracle 固有の拡張機能が含まれています。

このリリースでは、JNDI SPI の Oracle XML DB 実装によって、`javax.naming` インタフェースがサポートされています。ディレクトリ、属性およびイベントの各操作は、まだサポートされていません。

> **関連項目：** これらのインタフェースを使用したアプリケーション開発の詳細は、『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』を参照してください。

パッケージ oracle.xdb.spi クラスの概要

oracle.xdb.spi のクラスは、Oracle XML DB の核となる JNDI インタフェースと JNDI SPI インタフェースを実装します。

表 23-1 パッケージ oracle.xdb.spi のクラス

クラス	説明
XDBContext クラス	`javax.naming.context` インタフェースを実装します。
XDBContextFactory クラス	`javax.naming.spi.InitialContextFactory` を実装します。
XDBNameParser クラス	`javax.naming.NameParser` を実装します。
XDBNamingEnumeration クラス	`javax.naming.NamingEnumeration` を実装します。
XDBResource クラス	Oracle XML DB JNDIService Provider Interface（SPI）の核となる機能を実装します。
XDBResourceContext クラス	`javax.naming.context` を実装します。

XDBContext クラス

XDBContext の説明

このクラスは、Oracle XML DB の Java ネーミングおよびコンテキスト・インタフェースを実装し、javax.naming.context を拡張します。現行の実装はフェデレーションをサポートしないため、他の名前空間の存在は不明です。

XDBContext の構文

```
public class XDBContext

oracle.xdb.spi.XDBContext
```

メソッド

XDBContext()

説明

クラス XDBContext のコンストラクタ。次の表で、各オプションについて説明します。

構文	パラメータ
public XDBContext(java.util.Hashtable env)	環境をリソースに取得します。
public XDBContext(java.util.Hashtable env, java.lang.String path)	環境とパスをリソースに取得します。

パラメータ

env ― コンテキストのプロパティを説明する環境

path ― コンテキストの初期パス

XDBContextFactory クラス

XDBContextFactory の説明

このクラスは、javax.naming.context を実装します。

XDBContextFactory の構文

```
public class XDBContextFactory

oracle.xdb.spi.XDBContextFactory
```

コンストラクタ

XDBContextFactory()

説明

クラス XDBContextFactory のコンストラクタ

構文

```
public  XDBContextFactory()
```

XDBNameParser クラス

説明

javax.naming.NameParser を実装します。

構文

```
public class XDBNameParser
```

oracle.xdb.spi.XDBNameParser

XDBNamingEnumeration クラス

説明

javax.naming.NamingEnumeration を実装します。

構文

```
public class XDBNamingEnumeration
```

oracle.xdb.spi.XDBNamingEnumeration

XDBResource クラス

説明

このクラスは、Oracle XML DB JNDIService Provider Interface（SPI）の核となる機能を実装します。現行の実装はフェデレーションをサポートしないため、他の名前空間の存在は不明です。

構文

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

java.lang.Object
  |
  +--oracle.xdb.spi.XDBResource
```

メソッド

表 23-2 XDBResource のメソッド

メソッド	説明
XDBResource(Hashtable)	環境とパスをリソースに取得します。
XDBResource(Hashtable, String)	環境とパスを 1 つの文字列としてリソースに取得します。
getAuthor()	リソースの作成者を戻します。
getComment()	リソースの DAV コメントを戻します。
getContent()	リソースのコンテンツを戻します。
getContentType()	リソースのコンテンツ型を戻します。
getCreateDate()	リソースの作成日付を戻します。
getDisplayName()	リソースの表示名を戻します。
getLanguage()	リソースの言語を戻します。
getLastModDate()	リソースの最終変更日付を戻します。
getOwnerId()	リソースの所有者 ID を戻します。
setACL(String)	リソースに ACL を設定します。
setAuthor(String)	リソースの作成者を設定します。

表 23-2 XDBResource のメソッド（続き）

メソッド	説明
setComment(String)	リソースの DAV コメントを設定します。
setContent(Object)	リソースのコンテンツを設定します。
setContentType(String)	リソースのコンテンツ型を設定します。
setCreateDate(Date)	リソースの作成日付を設定します。
setDisplayName(String)	リソースの表示名を設定します。
setInheritedACL(String)	リソースに ACL を設定します。
setLanguage(String)	リソースの言語を設定します。
setLastModDate(Date)	リソースの最終変更日付を設定します。
setOwnerId(long)	リソースの所有者 ID を設定します。

XDBResource(Hashtable)

説明

環境とパスをリソースに取得します。

構文

```
public  XDBResource(java.util.Hashtable env)
```

パラメータ

env ― 渡された環境

XDBResource(Hashtable, String)

説明

環境とパスを 1 つの文字列としてリソースに取得します。

構文

```
public  XDBResource(java.util.Hashtable env, java.lang.String path)
```

getAuthor()

説明

リソースの作成者を戻します。

構文

```
public java.lang.String getAuthor()
```

getComment()

説明

リソースの DAV コメントを戻します（DAV は Web の分散オーサリングとバージョニングで使用されます）。

構文

```
public java.lang.String getComment()
```

getContent()

説明

リソースのコンテンツを戻します。

構文

```
public java.lang.Object getContent()
```

getContentType()

説明

リソースのコンテンツ型を戻します。

構文

```
public java.lang.String getContentType()
```

getCreateDate()

説明

リソースの作成日付を戻します。

構文

```
public java.util.Date getCreateDate()
```

getDisplayName()

説明

リソースの表示名を戻します。

構文

```
public java.lang.String getDisplayName()
```

getLanguage()

説明

リソースの言語を戻します。

構文

```
public java.lang.String getLanguage()
```

getLastModDate()

説明

リソースの最終変更日付を戻します。

構文

```
public java.util.Date getLastModDate()
```

getOwnerId()

説明

リソースの所有者 ID を戻します。

構文

```
public long getOwnerId()
```

setACL(String)

説明
リソースに ACL を設定します。

構文
```
public void setACL(java.lang.String aclpath)
```

パラメータ
aclpath − ACL リソースへのパス

setAuthor(String)

説明
リソースの作成者を設定します。

構文
```
public void setAuthor(java.lang.String authname)
```

パラメータ
authname − リソースの作成者

setComment(String)

説明
リソースの DAV コメントを設定します（Web の分散オーサリングとバージョニングで使用されます)。

構文
```
public void setComment(java.lang.String davcom)
```

パラメータ
davcom − リソースの DAV コメント

setContent(Object)

説明

リソースのコンテンツを設定します。

構文

```
public void setContent(java.lang.Object xmlobj)
```

パラメータ

xmlobj ― リソースのコンテンツ

setContentType(String)

説明

リソースのコンテンツ型を設定します。

構文

```
public void setContentType(java.lang.String conttype)
```

パラメータ

conttype ― リソースのコンテンツ型

setCreateDate(Date)

説明

リソースの作成日付を設定します。

構文

```
public void setCreateDate(java.util.Date credate)
```

パラメータ

credate ― リソースの作成日付

setDisplayName(String)

説明

リソースの表示名を設定します。

構文

```
public void setDisplayName(java.lang.String dname)
```

パラメータ

dname ― リソースの表示名

setInheritedACL(String)

説明

リソースに ACL を設定します。この ACL は指定のリソースからコピーされます。

構文

```
public void setInheritedACL(java.lang.String aclpath)
```

パラメータ

aclpath ― 設定する ACL へのパス

setLanguage(String)

説明

リソースの言語を設定します。

構文

```
public void setLanguage(java.lang.String lang)
```

パラメータ

lang ― リソースの言語

setLastModDate(Date)

説明

リソースの最終変更日付を設定します。

構文

```
public void setLastModDate(java.util.Date d)
```

パラメータ

d ― リソースの最終変更日付

setOwnerId(long)

説明

リソースの所有者 ID を設定します。

構文

```
public void setOwnerId(long ownerid)
```

パラメータ

ownerid ― リソースの所有者 ID

XDBResourceContext クラス

XDBResourceContext の説明

このクラスは、JNDI SPI の核となる Oracle XML DB 機能を提供します。現行の実装はフェデレーションをサポートしないため、他の名前空間の存在は不明です。このクラスは、`javax.naming.context` を実装します。

XDBResourceContext の構文

```
public class XDBResourceContext extends oracle.xdb.spi.XDBBaseContext

oracle.xdb.spi.XDBBaseContext
  |
  +--oracle.xdb.spi.XDBResourceContext
```

メソッド

表 23-3 XDBResourceContext のメソッド

メソッド	説明
XDBResourceContext(Hashtable)	環境を実装します。
getAuthor()	リソースの作成者を戻します。
getContentType()	リソースのコンテンツ型を戻します。
getCreateDate()	リソースの作成日付を戻します。
getDavComment()	リソースの DAV コメントを戻します。
getDisplayName()	リソースの表示名を戻します。
getEnvironment()	このコンテキストの環境のコピーを戻します。
getLanguage()	リソースの言語を戻します。
getLastModDate()	リソースの最終変更日付を戻します。
getOwnerId()	リソースの所有者 ID を戻します。
setAuthor(String)	リソースの作成者を設定します。
setContentType(String)	リソースのコンテンツ型を設定します。
setCreateDate(Date)	リソースの作成日付を設定します。
setDavComment(String)	リソースの DAV コメントを設定します。

表 23-3 XDBResourceContext のメソッド（続き）

メソッド	説明
setDisplayName(String)	リソースの表示名を設定します。
setLanguage(String)	リソースの言語を設定します。
setLastModDate(Date)	リソースの最終変更日付を設定します。
setOwnerId(long)	リソースの所有者 ID を設定します。

XDBResourceContext(Hashtable)

説明

環境を実装します。

構文

```
public   XDBResourceContext(java.util.Hashtable env)
```

パラメータ

env ― コンテキストの環境

getAuthor()

説明

リソースの作成者を戻します。

構文

```
public java.lang.String getAuthor()
```

getContentType()

説明

リソースのコンテンツ型を戻します。

構文

```
public java.lang.String getContentType()
```

getCreateDate()

説明
リソースの作成日付を戻します。

構文
```
public java.util.Date getCreateDate()
```

getDavComment()

説明
リソースの DAV コメントを戻します。

構文
```
public java.lang.String getDavComment()
```

getDisplayName()

説明
リソースの表示名を戻します。

構文
```
public java.lang.String getDisplayName()
```

getEnvironment()

説明
このコンテキストの環境プロパティを取り出します。取り出した結果は環境の新規コピーです。戻されたオブジェクトに対する変更は、コンテキストに影響を与えません。

構文
```
public java.util.Hashtable getEnvironment()
```

オーバーライド
XDBBaseContext クラスの XDBBaseContext.getEnvironment()

戻り値
このコンテキストの環境のコピー

getLanguage()

説明

リソースの言語を戻します。

構文

```
public java.lang.String getLanguage()
```

getLastModDate()

説明

リソースの最終変更日付を戻します。

構文

```
public java.util.Date getLastModDate()
```

getOwnerId()

説明

リソースの所有者 ID を戻します。

構文

```
public long getOwnerId()
```

setAuthor(String)

説明

リソースの作成者を設定します。

構文

```
public void setAuthor(java.lang.String authname)
```

パラメータ

authname ― リソースの作成者

setContentType(String)

説明
リソースのコンテンツ型を設定します。

構文
```
public void setContentType(java.lang.String conttype)
```

パラメータ
conttype ― リソースのコンテンツ型

setCreateDate(Date)

説明
リソースの作成日付を設定します。

構文
```
public void setCreateDate(java.util.Date credate)
```

パラメータ
credate ― リソースの作成日付

setDavComment(String)

説明
リソースの DAV コメントを設定します。

構文
```
public void setDavComment(java.lang.String davcom)
```

パラメータ
davcom ― リソースの DAV コメント

setDisplayName(String)

説明

リソースの表示名を設定します。

構文

```
public void setDisplayName(java.lang.String dname)
```

パラメータ

dname ― リソースの表示名

setLanguage(String)

説明

リソースの言語を設定します。

構文

```
public void setLanguage(java.lang.String lang)
```

パラメータ

lang ― リソースの言語

setLastModDate(Date)

説明

リソースの最終変更日付を設定します。

構文

```
public void setLastModDate(java.util.Date d)
```

パラメータ

d ― リソースの最終変更日付

setOwnerId(long)

説明

リソースの所有者 ID を設定します。

構文

```
public void setOwnerId(long ownerid)
```

パラメータ

`ownerid` ― リソースの所有者 ID

索引

記号

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

て

は

め

も