Oracle8*i*

XML リファレンス・ガイド Vol.2

リリース 8.1

2000 年 11 月

部品番号 : J02338-01

ORACLE®

Oracle8*i* XML リファレンス・ガイド Vol.2, リリース 8.1

部品番号 : J02338-01

原本名：Oracle8*i* XML Reference, Volume2, Release 3 (8.1.7)

原本部品番号：A86228-01

原本著者：Chitra Sharma

原本協力者：Shelley Higgins、Stefan Kiritzov、Visar Nimani、Mark Scardina

Copyright © 2000, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されております。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

* オラクル社とは、Oracle Corporation（米国オラクル）または日本オラクル株式会社（日本オラクル）を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

目次

はじめに v

このマニュアルの内容 vi

第 II 部　XDK for PL/SQL パッケージ

5　XML Parser for PL/SQL

PL/SQL Parser API 5-2

型およびファンクション 5-2

パーサー・インタフェース型の説明 5-4

ファンクション・プロトタイプ 5-4

拡張可能スタイルシート言語（XSL）Package Processor API 5-14

型およびファンクション 5-14

ファンクション・プロトタイプ 5-16

W3C 文書オブジェクト・モデル（DOM）API 5-23

型 5-24

メソッド・プロトタイプ 5-51

Named Node Map メソッド 5-60

Node List メソッド 5-61

Document Type メソッド 5-69

Element メソッド 5-74

org.w3c.dom.Attr インタフェース 5-94

抽象メソッド（一覧）..... 5-94

抽象メソッド（詳細）..... 5-95

org.w3c.dom.CDATASection インタフェース 5-96

org.w3c.dom.CharacterData インタフェース 5-96
抽象メソッド（一覧）..... 5-96
抽象メソッド（詳細）..... 5-97
org.w3c.dom.Comment インタフェース 5-100
org.w3c.dom.Document インタフェース 5-100
抽象メソッド（一覧）..... 5-101
抽象メソッド（詳細）..... 5-102
org.w3c.dom.DocumentFragment インタフェース 5-106
org.w3c.dom.DocumentType インタフェース 5-106
抽象メソッド（一覧）..... 5-107
抽象メソッド（詳細）..... 5-107
org.w3c.dom.DOMException クラス 5-108
org.w3c.dom.DOMImplementation インタフェース 5-109
抽象メソッド（一覧）..... 5-109
抽象メソッド（詳細）..... 5-110
org.w3c.dom.Element インタフェース 5-110
抽象メソッド（一覧）..... 5-111
抽象メソッド（詳細）..... 5-112
org.w3c.dom.Entity インタフェース 5-115
抽象メソッド（一覧）..... 5-116
抽象メソッド（詳細）..... 5-116
org.w3c.dom.EntityReference インタフェース 5-117
org.w3c.dom.NamedNodeMap インタフェース 5-117
抽象メソッド（一覧）..... 5-117
抽象メソッド（詳細）..... 5-118
org.w3c.dom.Node インタフェース 5-120
抽象メソッド（一覧）..... 5-120
抽象メソッド（詳細）..... 5-123
org.w3c.dom.NodeList インタフェース 5-128
抽象メソッド（一覧）..... 5-128
抽象メソッド（詳細）..... 5-128
org.w3c.dom.Notation インタフェース 5-129
抽象メソッド（一覧）..... 5-129
抽象メソッド（詳細）..... 5-129
org.w3c.dom.ProcessingInstruction インタフェース 5-130

抽象メソッド（一覧）..... 5-130
抽象メソッド（詳細）..... 5-130
org.w3c.dom.Text インタフェース 5-131
抽象メソッド（一覧）..... 5-131
抽象メソッド（詳細）..... 5-131

第 III 部　XML-SQL Utility（XSU）パッケージ

6　Oracle XSU Java API

oracle.xml.sql.dml.OracleXMLSave 6-2
oracle.xml.sql.query.OracleXMLQuery 6-8
oracle.xml.sql.OracleXMLSQLException 6-18
oracle.xml.sql.OracleXMLSQLNoRowsException 6-21

7　Oracle XSU PL/SQL API

DBMS_XMLQuery 7-2
型 7-2
定数 7-2
ファンクションおよびプロシージャ（一覧）..... 7-3
ファンクションおよびプロシージャ（詳細）..... 7-6
DBMS_XMLSave 7-16
型 7-16
定数 7-16
ファンクションおよびプロシージャ（一覧）..... 7-16
ファンクションおよびプロシージャ（詳細）..... 7-19

索引

はじめに

この章の内容は、次のとおりです。

- このマニュアルの内容

このマニュアルの内容

このマニュアルでは、Oracle8*i* で提供される XML について説明します。第 I 部では XDK for Java パッケージ、第 II 部では XDK for PL/SQL パッケージについて、例を使用して説明します。第 III 部では、XSU for Java および XSU for PL/SQL について説明します。

詳細は、次のマニュアルを参照してください。

- 『Oracle8*i* アプリケーション開発者ガイド 基礎編』
- 『Oracle8*i* アプリケーション開発者ガイド - XML』

第 II 部

XDK for PL/SQL パッケージ

第 II 部に含まれる章は、次のとおりです。

- 第 5 章「XML Parser for PL/SQL」

5

XML Parser for PL/SQL

この章では、拡張可能マークアップ言語（XML）Parser for PL/SQL について説明します。内容は次のとおりです。

- PL/SQL Parser API
- 拡張可能スタイルシート言語（XSL）Package Processor API
- W3C 文書オブジェクト・モデル（DOM）API

PL/SQL Parser API

XML は、XML ドキュメントというデータ・オブジェクトのクラスを記述します。XML は、XML ドキュメントを処理するコンピュータ・プログラムの動作を部分的に記述します。XML はアプリケーション・プロファイルであり、標準汎用マークアップ言語（Standard Generalized Markup Language: SGML）の機能を一部制限したものです。構成的には、XML ドキュメントは SGML ドキュメントに準拠します。

XML ドキュメントは、エンティティという記憶単位で構成されます。エンティティには、解析対象データまたは解析対象外データのいずれかが格納されます。解析対象データは文字で構成され、文字データやマークアップを形成します。マークアップは、ドキュメントの記憶レイアウトおよび論理構造の記述をエンコーディングします。XML は、記憶レイアウトおよび論理構造に制約を適用するためのメカニズムを提供します。

XML ドキュメントの読込み、およびその内容と構造へのアクセスには、XML プロセッサというソフトウェア・モジュールが使用されます。XML プロセッサは、別のモジュール（アプリケーション）にかわって作業を行っていることを想定しています。この XML プロセッサ（またはパーサー）の PL/SQL 実装は、W3C の XML 仕様（改訂 REC-xml-19980210）に準拠しており、XML データを読み込むために XML プロセッサに必要な動作、およびアプリケーションに提供する必要がある情報を含んでいます。

この PL/SQL XML パーサーのデフォルトの動作を次に示します。

- DOM API がアクセス可能な 1 つの解析ツリーが構築されます。
- パーサーは、DTD が検出されると検証を行い、検出されないと検証を行いません。
- エラー・ログを指定しないとエラーは記録されませんが、解析が正常に実行されなかった場合は、アプリケーション・エラーが発生します。

このマニュアルで説明する型およびメソッドは、PL/SQL パッケージ xmlparser で利用可能です。

型およびファンクション

parse(VARCHAR2)

与えられた URL/ ファイルに格納された XML を解析し、構築済の DOM ドキュメントを戻します。

newParser

新しいパーサー・インスタンスを戻します。

parse(Parser, VARCHAR2)

与えられた URL/ ファイルに格納された XML を解析します。

parseBuffer(Parser, VARCHAR2)

与えられたバッファに格納された XML を解析します。

parseClob(Parser, CLOB)

与えられた CLOB に格納された XML を解析します。

parseDTD(Parser, VARCHAR2, VARCHAR2)

与えられた URL/ ファイルに格納された XML を解析します。

parseDTDBuffer(Parser, VARCHAR2, VARCHAR2)

与えられたバッファに格納された XML を解析します。

parseDTDClob(Parser, CLOB, VARCHAR2)

与えられた CLOB に格納された XML を解析します。

setBaseDir(Parser, VARCHAR2)

関連 URL を変換するために使用するベース・ディレクトリを設定します。

showWarnings(Parser, BOOLEAN)

警告を表示または非表示にします。

setErrorLog(Parser, VARCHAR2)

指定したファイルにエラーが送信されるように設定します。

setPreserveWhitespace(Parser, BOOLEAN)

空白保持モードを設定します。

setValidationMode(Parser, BOOLEAN)

検証モードを設定します。

getValidationMode(Parser)

検証モードを取得します。

setDoctype(Parser, DOMDocumentType)

DTD を設定します。

getDoctype(Parser)

DTD パーサーを取得します。

getDocument(Parser)

DOM ドキュメントを取得します。

freeParser(Parser)

パーサー・オブジェクトを解放します。

パーサー・インタフェース型の説明

TYPE Parser IS RECORD (ID VARCHAR2(5));

ファンクション・プロトタイプ

parse

用途

与えられた URL/ ファイルに格納された XML を解析し、構築済の DOM ドキュメントを戻します。

構文

```
FUNCTION parse(url VARCHAR2) RETURN DOMDocument;
```

パラメータ

`url` (IN) - 解析する URL/ ファイルの完全パス

戻り値

なし

コメント

パーサーのデフォルトの動作が受け入れられ、URL/ ファイルのみを解析する必要がある場合に使用します。

解析が正常に実行されなかった場合、アプリケーション・エラーが発生します。

newParser

用途

新しいパーサー・インスタンスを戻します。

構文

```
FUNCTION newParser RETURN Parser;
```

パラメータ

なし

戻り値

新しいパーサー・インスタンス

コメント

パーサーのデフォルトの動作が変更される前で、他の解析メソッドを使用する必要がある場合にコールします。

parse

用途

与えられた URL/ ファイルに格納された XML を解析します。

構文

```
PROCEDURE parse(p Parser, url VARCHAR2);
```

パラメータ

p	(IN)	- パーサー・インスタンス
url	(IN)	- 解析する URL/ ファイルの完全パス

戻り値

なし

コメント

パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。

解析が正常に実行されなかった場合、アプリケーション・エラーが発生します。

parseBuffer

用途

与えられたバッファに格納された XML を解析します。

構文

```
PROCEDURE parseBuffer(p Parser, doc VARCHAR2);
```

パラメータ

p (IN) - パーサー・インスタンス

doc (IN) - 解析する XML ドキュメント・バッファ

戻り値

なし

コメント

パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。

解析が正常に実行されなかった場合、アプリケーション・エラーが発生します。

parseClob

用途

与えられた CLOB に格納された XML を解析します。

構文

```
PROCEDURE parseClob(p Parser, doc CLOB);
```

パラメータ

p (IN) - パーサー・インスタンス

doc (IN) - 解析する XML ドキュメント CLOB

戻り値

なし

コメント

パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。

解析が正常に実行されなかった場合、アプリケーション・エラーが発生します。

parseDTD

用途

与えられた URL/ ファイルに格納された DTD を解析します。

構文

```
PROCEDURE parseDTD(p Parser, url VARCHAR2, root VARCHAR2);
```

パラメータ

p	(IN)	- パーサー・インスタンス
url	(IN)	- 解析する URL/ ファイルの完全パス
root	(IN)	- ルート要素の名前

戻り値

なし

コメント

パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。

解析が正常に実行されなかった場合、アプリケーション・エラーが発生します。

parseDTDBuffer

用途

与えられたバッファに格納された DTD を解析します。

構文

```
PROCEDURE parseDTDBuffer(p Parser, dtd VARCHAR2, root VARCHAR2);
```

パラメータ

`p` (IN) - パーサー・インスタンス

`dtd` (IN) - 解析する DTD バッファ

`root` (IN) - ルート要素の名前

戻り値

なし

コメント

パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。

解析が正常に実行されなかった場合、アプリケーション・エラーが発生します。

parseDTDClob

用途

与えられた CLOB に格納された DTD を解析します。

構文

```
PROCEDURE parseDTDClob(p Parser, dtd CLOB, root VARCHAR2);
```

パラメータ

`p` (IN) - パーサー・インスタンス

`dtd` (IN) - 解析する DTD CLOB

`root` (IN) - ルート要素の名前

戻り値

なし

コメント

パーサーのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。

解析が正常に実行されなかった場合、アプリケーション・エラーが発生します。

setBaseDir

用途

関連 URL を変換するために使用するベース・ディレクトリを設定します。

構文

```
PROCEDURE setBaseDir(p Parser, dir VARCHAR2);
```

パラメータ

p　　(IN) - パーサー・インスタンス

dir　　(IN) - ベース・ディレクトリとして使用するディレクトリ

戻り値

なし

コメント

解析が正常に実行されなかった場合、アプリケーション・エラーが発生します。

showWarnings

用途

警告を表示または非表示にします。

構文

```
PROCEDURE showWarnings(p Parser, yes BOOLEAN);
```

パラメータ

p　　(IN) - パーサー・インスタンス

yes　　(IN) - 設定モード : TRUE（警告を表示する）または FALSE（警告を表示しない）

戻り値

なし

setErrorLog

用途

指定したファイルにエラーが送信されるように設定します。

構文

```
PROCEDURE setErrorLog(p Parser, fileName VARCHAR2);
```

パラメータ

p (IN) - パーサー・インスタンス

fileName (IN) - エラー・ログとして使用するファイルの完全パス

戻り値

なし

setPreserveWhitespace

用途

空白保持モードを設定します。

構文

```
PROCEDURE setPreserveWhitespace(p Parser, yes BOOLEAN);
```

パラメータ

p (IN) - パーサー・インスタンス

yes (IN) - 設定モード : TRUE（保持する）または FALSE（保持しない）

戻り値

なし

setValidationMode

用途

検証モードを設定します。

構文

```
PROCEDURE setValidationMode(p Parser, yes BOOLEAN);
```

パラメータ

p　　(IN) - パーサー・インスタンス

yes　　(IN) - 設定モード : TRUE（検証する）または FALSE（検証しない）

戻り値

なし

getValidationMode

用途

検証モードを取得します。

構文

```
FUNCTION getValidationMode(p Parser) RETURN BOOLEAN;
```

パラメータ

p　　(IN) - パーサー・インスタンス

戻り値

検証モード : TRUE（検証する）または FALSE（検証しない）

setDoctype

用途

パーサーが検証に使用する DTD を設定します。

構文

```
PROCEDURE setDoctype(p Parser, dtd DOMDocumentType);
```

パラメータ

p　　(IN) - パーサー・インスタンス

dtd　　(IN) - 設定する DTD

戻り値

なし

getDoctype

用途

DTD を取得します。このファンクションのコールは、DTD の解析後のみ有効です。

構文

```
FUNCTION getDoctype(p Parser) RETURN DOMDocumentType;
```

パラメータ

p　　(IN) - パーサー・インスタンス

戻り値

解析済 DTD

getDocument

用途

パーサーが構築した DOM ドキュメントを取得します。このファンクションのコールは、ドキュメントの解析後のみ有効です。

構文

```
FUNCTION getDocument(p Parser) RETURN DOMDocument;
```

パラメータ

p　　(IN) - パーサー・インスタンス

戻り値

DOM ツリーのルート

freeParser

用途

パーサー・オブジェクトを解放します。

構文

```
PROCEDURE freeParser(p Parser)
```

パラメータ

p　　(IN) - パーサー・インスタンス

拡張可能スタイルシート言語（XSL）Package Processor API

拡張可能スタイルシート言語変換（Extensible Stylesheet Language Transformation: XSLT）は、ソース・ツリーを結果ツリーに変換するための規則を記述しています。XSLT に記述されている変換をスタイルシートといいます。指定した変換は、スタイルシートに定義されたテンプレートを使用してパターンを対応付けることによって実行されます。テンプレートは、結果ツリーの一部を作成するためにインスタンス化されます。この XSL プロセッサの PL/SQL 実装は、W3C の XSLT ワーキング・ドラフト（改訂 WD-xslt-19990813）に準拠しており、XSLT スタイルシートを読み込むために XML プロセッサに必要な動作、および実行する必要がある変換を含んでいます。

この PL/SQL XML パーサーのデフォルトの動作を次に示します。

- DOM API がアクセス可能な 1 つの結果ツリーが構築されます。
- エラー・ログを指定しないとエラーは記録されませんが、解析が正常に実行されなかった場合は、アプリケーション・エラーが発生します。

このマニュアルで説明する型およびメソッドは、PL/SQL パッケージの xslprocessor で利用可能です。

型およびファンクション

newProcessor

新しいプロセッサ・インスタンスを戻します。

processXSL(Processor, Stylesheet, DOMDocument)

与えられた DOM ドキュメントおよびスタイルシートを使用して、入力された XML ドキュメントを変換します。

processXSL(Processor, Stylesheet, DOMDocumentFragment)

与えられた DOM ドキュメント・フラグメントおよびスタイルシートを使用して、入力された XML ドキュメントを変換します。

showWarnings(Processor, BOOLEAN)

警告を表示または非表示にします。

setErrorLog(Processor, VARCHAR2)

指定したファイルにエラーが送信されるように設定します。

newStylesheet(VARCHAR2, VARCHAR2)

与えられた入力 URL および参照 URL を使用して、新しいスタイルシートを作成します。

transformNode(DOMNode, Stylesheet)

与えられたスタイルシートを使用して、DOM ツリーのノードを変換します。

selectNodes(DOMNode, VARCHAR2)

DOM ツリーから、与えられたパターンに一致するノードを選択します。

selectSingleNodes(DOMNode, VARCHAR2)

ツリーから、与えられたパターンに一致する最初のノードを選択します。

valueOf(DOMNode, VARCHAR2)

ツリーから、与えられたパターンに一致する最初のノードの値を取得します。

setParam(Stylesheet, VARCHAR2, VARCHAR2)

スタイルシートのトップレベル・パラメータを設定します。

removeParam(Stylesheet, VARCHAR2)

スタイルシートのトップレベル・パラメータを削除します。

resetParams(Stylesheet)

スタイルシートのトップレベル・パラメータをリセットします。

freeStylesheet(Stylesheet)

スタイルシート・オブジェクトを解放します。

freeProcessor(Processor)

プロセッサ・オブジェクトを解放します。

ファンクション・プロトタイプ

newProcessor

用途

新しいプロセッサ・インスタンスを戻します。

構文

```
FUNCTION newProcessor RETURN Processor;
```

パラメータ

なし

戻り値

新しいプロセッサ・インスタンス

コメント

プロセッサのデフォルトの動作が変更される前で、他のプロセッサ・メソッドを使用する必要がある場合にコールします。

processXSL

用途

与えられた DOM ドキュメントおよびスタイルシートを使用して、入力された XML ドキュメントを変換します。

構文

```
PROCEDURE processXSL(p Processor, ss Stylesheet, xmldoc DOMDocument);
```

パラメータ

p	(IN)	- プロセッサ・インスタンス
ss	(IN)	- スタイルシート・インスタンス
xmldoc	(IN)	- 変換する XML ドキュメント

戻り値

なし

コメント

プロセッサのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。

処理が正常に実行されなかった場合、アプリケーション・エラーが発生します。

processXSL

用途

与えられた DOM ドキュメント・フラグメントおよびスタイルシートを使用して、入力された XML ドキュメントを変換します。

構文

```
PROCEDURE processXSL(p Processor, ss Stylesheet, xmldoc DOMDocumentFragment);
```

パラメータ

p	(IN)	- プロセッサ・インスタンス
ss	(IN)	- スタイルシート・インスタンス
xmldoc	(IN)	- 変換する XML ドキュメント・フラグメント

戻り値

なし

コメント

プロセッサのデフォルトの動作に対する変更は、このプロシージャをコールする前に有効にする必要があります。

処理が正常に実行されなかった場合、アプリケーション・エラーが発生します。

showWarnings

用途

警告を表示または非表示にします。

構文

```
PROCEDURE showWarnings(p Processor, yes BOOLEAN);
```

パラメータ

p　(IN) - プロセッサ・インスタンス

yes　(IN) - 設定モード : TRUE（警告を表示する）または FALSE（警告を表示しない）

戻り値

なし

setErrorLog

用途

指定したファイルにエラーが送信されるように設定します。

構文

```
PROCEDURE setErrorLog(p Processor, fileName VARCHAR2);
```

パラメータ

p　(IN) - プロセッサ・インスタンス

fileName　(IN) - エラー・ログとして使用するファイルの完全パス

戻り値

なし

newStylesheet

用途

与えられた入力 URL および参照 URL を使用して、新しいスタイルシートを作成します。

構文

```
FUNCTION newStylesheet(inp VARCHAR2, ref VARCHAR2) RETURN Stylesheet;
```

パラメータ

inp　(IN) - プロセッサ・インスタンス

ref　(IN) - 参照 URL

戻り値

新しいスタイルシート・インスタンス

transformNode

用途

与えられたスタイルシートを使用して、DOM ツリーのノードを変換します。

構文

```
FUNCTION transformNode(n DOMNode, ss Stylesheet) RETURN DOMDocumentFragment;
```

パラメータ

n	(IN)	- 変換する DOM ノード
ss	(IN)	- 使用するスタイルシート

戻り値

ドキュメント・フラグメント

selectNodes

用途

DOM ツリーから、与えられたパターンに一致するノードを選択します。

構文

```
FUNCTION selectNodes(n DOMNode, pattern VARCHAR2) RETURN DOMNodeList;
```

パラメータ

n	(IN)	- 変換する DOM ノード
pattern	(IN)	- 使用するパターン

戻り値

一致するノード・リスト

selectSingleNodes

用途

ツリーから、与えられたパターンに一致する最初のノードを選択します。

構文

```
FUNCTION selectSingleNodes(n DOMNode, pattern VARCHAR2) RETURN DOMNode;
```

パラメータ

n　　(IN) - 変換する DOM ノード

pattern　(IN) - 使用するパターン

戻り値

一致するノード

valueOf

用途

ツリーから、与えられたパターンに一致する最初のノードの値を取得します。

構文

```
FUNCTION valueOf(n DOMNode, pattern VARCHAR2) RETURN VARCHAR2;
```

パラメータ

n　　(IN) - 変換する DOM ノード

pattern　(IN) - 使用するパターン

戻り値

一致するノードの値

setParam

用途

スタイルシートのトップレベル・パラメータを設定します。

構文

```
PROCEDURE setParam(ss Stylesheet, name VARCHAR2, value VARCHAR2)
```

パラメータ

ss (IN) - スタイルシート

name (IN) - パラメータの名前

value (IN) - パラメータの値

removeParam

用途

スタイルシートのトップレベル・パラメータを削除します。

構文

```
PROCEDURE removeParam(ss Stylesheet, name VARCHAR2)
```

パラメータ

ss (IN) - スタイルシート

name (IN) - パラメータの名前

resetParams

用途

スタイルシートのトップレベル・パラメータをリセットします。

構文

```
PROCEDURE resetParams(ss Stylesheet)
```

パラメータ

ss (IN) - スタイルシート

freeStylesheet

用途

スタイルシート・オブジェクトを解放します。

構文

```
PROCEDURE freestylesheet(ss Stylesheet)
```

パラメータ

ss　　(IN) - スタイルシート

freeProcessor

用途

プロセッサ・オブジェクトを解放します。

構文

```
PROCEDURE freeProccessor(p Processor)
```

パラメータ

p　　(IN) - プロセッサ

W3C 文書オブジェクト・モデル（DOM）API

文書オブジェクト・モデル（Document Object Model: DOM）は、HTML および XML ドキュメント用のアプリケーション・プログラム・インタフェース（API）です。DOM は、ドキュメントの論理構造、ドキュメントへのアクセス方法および操作方法を定義します。DOM 仕様では、「ドキュメント」という用語は、幅広い意味で使用されます。XML は、多様なシステムに格納できる様々な情報を表す手段としてますます需要が高まっていますが、情報の多くは、従来、ドキュメントとしてではなくデータとして認識されてきました。これに対し、XML はデータをドキュメントとして表し、DOM を使用して、このデータを管理します。

プログラマは、DOM を使用してドキュメントを構築し、その構造をナビゲートしたり、要素および内容を追加、変更および削除することができます。HTML または XML ドキュメントのすべての内容は、DOM を使用してアクセス、変更、削除または追加できます。ただし、いくつかの例外もあります。特に、XML の内部および外部サブセットに対する DOM インタフェースは、まだ仕様が確立されていません。

W3C による DOM 仕様の重要な目的の 1 つは、様々な環境およびアプリケーションで使用できる標準プログラミング・インタフェースを提供することです。DOM は、すべてのプログラミング言語で使用できるように設計されています。DOM 標準はオブジェクト指向であるため、PL/SQL に適用するには、次の変更が必要となります。

- Node、Element などの様々な DOM インタフェースには、それぞれ同等の PL/SQL 型 DOMNode や DOMElement があります。
- WRONG_DOCUMENT_ERR、HIERARCHY_REQUEST_ERR などの様々な DOMException コードには、同様の名前が付いた PL/SQL 例外があります。
- ELEMENT_NODE、ATTRIBUTE_NODE などの様々な DOM ノード型コードには、同様の名前が付いた PL/SQL 定数があります。
- DOM 型で定義済のメソッドは、DOM 型をパラメータとして取るファンクションまたはプロシージャになります。たとえば、appendChild を DOM ノード n で実行するには、次の PL/SQL ファンクションを実行します。

ファンクション

```
appendChild(n DOMNode, newChild IN DOMNode)
```

戻り値

DOMNode

また、setAttribute を DOM 要素 elem で実行するには、次の PL/SQL プロシージャを実行します。

プロシージャ

```
setAttribute(elem DOMElement, name IN VARCHAR2,
                              value IN VARCHAR2)
```

DOM は、継承階層を定義します。たとえば、ドキュメント、要素および属性は、ノードのサブタイプとして定義されます。このため、Node インタフェースで定義済のメソッドは、ドキュメント、要素および属性でも使用可能です。PL/SQL の場合、直接このような継承を行うことはできないため、異なる DOM 型で makeNode ファンクションをコールし、これらの DOM 型を DOMNode に変換する必要があります。この変換後に、DOMNode を指定する適切なファンクションまたはプロシージャをコールすると、これらの型を操作できるようになります。その後、型固有の機能が必要になった場合は、makeXXX ファンクションを使用して、DOMNode を元の型に戻すこともできます。ここの場合、DOMXXX は、希望の DOM 型を表します。

この PL/SQL DOM インタフェースの実装は、改訂 REC-DOM-Level-1-19981001 の DOM 標準に準拠します。このマニュアルで説明する型およびメソッドは、PL/SQL パッケージ xmldom で利用可能です。

型

DOM ノード型

ELEMENT_NODE

ATTRIBUTE_NODE

TEXT_NODE

CDATA_SECTION_NODE

ENTITY_REFERENCE_NODE

ENTITY_NODE

PROCESSING_INSTRUCTION_NODE

COMMENT_NODE

DOCUMENT_NODE

DOCUMENT_TYPE_NODE

DOCUMENT_FRAGMENT_NODE

NOTATION_NODE

DOM 例外型

INDEX_SIZE_ERR

DOMSTRING_SIZE_ERR

HIERARCHY_REQUEST_ERR

WRONG_DOCUMENT_ERR

INVALID_CHARACTER_ERR

NO_DATA_ALLOWED_ERR

NO_MODIFICATION_ALLOWED_ERR

NOT_FOUND_ERR

NOT_SUPPORTED_ERR

INUSE_ATTRIBUTE_ERR

DOM インタフェース型

DOMNode

DOMNamedNodeMap

DOMNodeList

DOMAttr

DOMCDataSection

DOMCharacterData

DOMComment

DOMDocumentFragment

DOMElement

DOMEntity

DOMEntityReference

DOMNotation

DOMProcessingInstruction

DOMText

DOMImplementation

DOMDocumentType

DOMDocument

メソッド

Node メソッド

ファンクション

`isNull(n DOMNode)`

戻り値

BOOLEAN

ファンクション

`makeAttr(n DOMNode)`

戻り値

DOMAttr

ファンクション

`makeCDataSection(n DOMNode)`

戻り値

DOMCDataSection

ファンクション

`makeCharacterData(n DOMNode)`

戻り値

DOMCharacterData

ファンクション

`makeComment(n DOMNode)`

戻り値

DOMComment

ファンクション

`makeDocumentFragment(n DOMNode)`

戻り値

DOMDocumentFragment

ファンクション

`makeDocumentType(n DOMNode)`

戻り値

DOMDocumentType

ファンクション

`makeElement(n DOMNode)`

戻り値

DOMElement

ファンクション

`makeEntity(n DOMNode)`

戻り値

DOMEntity

ファンクション

`makeEntityReference(n DOMNode)`

戻り値

DOMEntityReference

ファンクション

`makeNotation(n DOMNode)`

戻り値

DOMNotation

ファンクション

makeProcessingInstruction(n DOMNode)

戻り値

DOMProcessingInstruction

ファンクション

makeText(n DOMNode)

戻り値

DOMText

ファンクション

makeDocument(n DOMNode)

戻り値

DOMDocument

プロシージャ

writeToFile(n DOMNode, fileName VARCHAR2)

プロシージャ

writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2)

プロシージャ

writeToClob(n DOMNode, cl IN OUT CLOB)

プロシージャ

writeToFile(n DOMNode, fileName VARCHAR2, charset VARCHAR2)

プロシージャ

writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2, charset VARCHAR2)

プロシージャ

writeToClob(n DOMNode, cl IN OUT CLOB, charset VARCHAR2)

ファンクション

`getNodeName(n DOMNode)`

戻り値

VARCHAR2

ファンクション

`getNodeValue(n DOMNode)`

戻り値

VARCHAR2

プロシージャ

`setNodeValue(n DOMNode, nodeValue IN VARCHAR2)`

ファンクション

`getNodeType(n DOMNode)`

戻り値

NUMBER

ファンクション

`getParentNode(n DOMNode)`

戻り値

DOMNode

ファンクション

`getChildNodes(n DOMNode)`

戻り値

DOMNodeList

ファンクション

`getFirstChild(n DOMNode)`

戻り値

DOMNode

ファンクション

`getLastChild(n DOMNode)`

戻り値

DOMNode

ファンクション

`getPreviousSibling(n DOMNode)`

戻り値

DOMNode

ファンクション

`getNextSibling(n DOMNode)`

戻り値

DOMNode

ファンクション

`getAttributes(n DOMNode)`

戻り値

DOMNamedNodeMap

ファンクション

`getOwnerDocument(n DOMNode)`

戻り値

DOMDocument

ファンクション

`insertBefore(n DOMNode, newChild IN DOMNode, refChild IN DOMNode)`

戻り値

DOMNode

ファンクション

`replaceChild(n DOMNode, newChild IN DOMNode, oldChild IN DOMNode)`

戻り値

DOMNode

ファンクション

`removeChild(n DOMNode, oldChild IN DOMNode)`

戻り値

DOMNode

ファンクション

`appendChild(n DOMNode, newChild IN DOMNode)`

戻り値

DOMNode

ファンクション

`hasChildNodes(n DOMNode)`

戻り値

BOOLEAN

ファンクション

`cloneNode(n DOMNode, deep boolean)`

戻り値

DOMNode

Named Node Map メソッド

ファンクション

`isNull(nnm DOMNamedNodeMap)`

戻り値

BOOLEAN

ファンクション

`getNamedItem(nnm DOMNamedNodeMap, name IN VARCHAR2)`

戻り値

DOMNode

ファンクション

`setNamedItem(nnm DOMNamedNodeMap, arg IN DOMNode)`

戻り値

DOMNode

ファンクション

`removeNamedItem(nnm DOMNamedNodeMap, name IN VARCHAR2)`

戻り値

DOMNode

ファンクション

`item(nnm DOMNamedNodeMap, idx IN NUMBER)`

戻り値

DOMNode

ファンクション

`getLength(nnm DOMNamedNodeMap)`

戻り値

NUMBER

Node List メソッド

ファンクション

`isNull(nl DOMNodeList)`

戻り値

BOOLEAN

ファンクション

`item(nl DOMNodeList, idx IN NUMBER)`

戻り値

DOMNode

ファンクション

`getLength(nl DOMNodeList)`

戻り値

NUMBER

Attr メソッド

ファンクション

`isNull(a DOMAttr)`

戻り値

BOOLEAN

ファンクション

`makeNode(a DOMAttr)`

戻り値

DOMNode

ファンクション

getQualifiedName(a DOMAttr)

戻り値

VARCHAR2

ファンクション

getNamespace(a DOMAttr)

戻り値

VARCHAR2

ファンクション

getLocalName(a DOMAttr)

戻り値

VARCHAR2

ファンクション

getExpandedName(a DOMAttr)

戻り値

VARCHAR2

ファンクション

getName(a DOMAttr)

戻り値

VARCHAR2

ファンクション

getSpecified(a DOMAttr)

戻り値

BOOLEAN

ファンクション

`getValue(a DOMAttr)`

戻り値

VARCHAR2

プロシージャ

`setValue(a DOMAttr, value IN VARCHAR2)`

C Data Section メソッド

ファンクション

`isNull(cds DOMCDataSection)`

戻り値

BOOLEAN

ファンクション

`makeNode(cds DOMCDataSection)`

戻り値

DOMNode

Character Data メソッド

ファンクション

`isNull(cd DOMCharacterData)`

戻り値

BOOLEAN

ファンクション

`makeNode(cd DOMCharacterData)`

戻り値

DOMNode

ファンクション

getData(cd DOMCharacterData)

戻り値

VARCHAR2

プロシージャ

setData(cd DOMCharacterData, data IN VARCHAR2)

ファンクション

getLength(cd DOMCharacterData)

戻り値

NUMBER

ファンクション

substringData(cd DOMCharacterData, offset IN NUMBER, cnt IN NUMBER)

戻り値

VARCHAR2

プロシージャ

appendData(cd DOMCharacterData, arg IN VARCHAR2)

プロシージャ

insertData(cd DOMCharacterData, offset IN NUMBER, arg IN VARCHAR2)

プロシージャ

deleteData(cd DOMCharacterData, offset IN NUMBER, cnt IN NUMBER)

プロシージャ

replaceData(cd DOMCharacterData, offset IN NUMBER, cnt IN NUMBER, arg IN VARCHAR2)

Comment メソッド

ファンクション

`isNull(com DOMComment)`

戻り値

BOOLEAN

ファンクション

`makeNode(com DOMComment)`

戻り値

DOMNode

DOM Implementation メソッド

ファンクション

`isNull(di DOMImplementation)`

戻り値

BOOLEAN

ファンクション

`hasFeature(di DOMImplementation, feature IN VARCHAR2, version IN VARCHAR2)`

戻り値

BOOLEAN

Document Fragment メソッド

ファンクション

`isNull(df DOMDocumentFragment)`

戻り値

BOOLEAN

ファンクション

makeNode(df DOMDocumentFragment)

戻り値

DOMNode

Document Type メソッド

ファンクション

isNull(dt DOMDocumentType)

戻り値

BOOLEAN

ファンクション

makeNode(dt DOMDocumentType)

戻り値

DOMNode

ファンクション

findEntity(dt DOMDocumentType, name VARCHAR2, par BOOLEAN)

戻り値

DOMEntity

ファンクション

findNotation(dt DOMDocumentType, name VARCHAR2)

戻り値

DOMNotation

ファンクション

getPublicId(dt DOMDocumentType)

戻り値

VARCHAR2

ファンクション

getSystemId(dt DOMDocumentType)

戻り値

VARCHAR2

プロシージャ

writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2)

プロシージャ

writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2)

プロシージャ

writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB)

プロシージャ

writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2, charset VARCHAR2)

プロシージャ

writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2, charset VARCHAR2)

プロシージャ

writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB, charset VARCHAR2)

ファンクション

getName(dt DOMDocumentType)

戻り値

VARCHAR2

ファンクション

getEntities(dt DOMDocumentType)

戻り値

DOMNamedNodeMap

ファンクション

`getNotations(dt DOMDocumentType)`

戻り値

DOMNamedNodeMap

Element メソッド

ファンクション

`isNull(elem DOMElement)`

戻り値

BOOLEAN

ファンクション

`makeNode(elem DOMElement)`

戻り値

DOMNode

ファンクション

`getQualifiedName(elem DOMElement)`

戻り値

VARCHAR2

ファンクション

`getNamespace(elem DOMElement)`

戻り値

VARCHAR2

ファンクション

`getLocalName(elem DOMElement)`

戻り値

VARCHAR2

ファンクション

`getExpandedName(elem DOMElement)`

戻り値

VARCHAR2

ファンクション

`getChildrenByTagName(elem DOMElement, name IN VARCHAR2)`

戻り値

DOMNodeList

ファンクション

`getElementsByTagName(elem DOMElement, name IN VARCHAR2, ns VARCHAR2)`

戻り値

DOMNodeList

ファンクション

`getChildrenByTagName(elem DOMElement, name IN VARCHAR2, ns VARCHAR2)`

戻り値

DOMNodeList

ファンクション

`resolveNamespacePrefix(elem DOMElement, prefix VARCHAR2)`

戻り値

VARCHAR2

ファンクション

`getTagName(elem DOMElement)`

戻り値

VARCHAR2

ファンクション

```
getAttribute(elem DOMElement, name IN VARCHAR2)
```

戻り値

VARCHAR2

プロシージャ

```
setAttribute(elem DOMElement, name IN VARCHAR2, value IN VARCHAR2)
```

プロシージャ

```
removeAttribute(elem DOMElement, name IN VARCHAR2)
```

ファンクション

```
getAttributeNode(elem DOMElement, name IN VARCHAR2)
```

戻り値

DOMAttr

ファンクション

```
setAttributeNode(elem DOMElement, newAttr IN DOMAttr)
```

戻り値

DOMAttr

ファンクション

```
removeAttributeNode(elem DOMElement, oldAttr IN DOMAttr)
```

戻り値

DOMAttr

ファンクション

```
getElementsByTagName(elem DOMElement, name IN VARCHAR2)
```

戻り値

DOMNodeList

プロシージャ

`normalize(elem DOMElement)`

Entity メソッド

ファンクション

`isNull(ent DOMEntity)`

戻り値

BOOLEAN

ファンクション

`makeNode(ent DOMEntity)`

戻り値

DOMNode

ファンクション

`getPublicId(ent DOMEntity)`

戻り値

VARCHAR2

ファンクション

`getSystemId(ent DOMEntity)`

戻り値

VARCHAR2

ファンクション

`getNotationName(ent DOMEntity)`

戻り値

VARCHAR2

Entity Reference メソッド

ファンクション

isNull(eref DOMEntityReference)

戻り値

BOOLEAN

ファンクション

makeNode(eref DOMEntityReference)

戻り値

DOMNode

Notation メソッド

ファンクション

isNull(n DOMNotation)

戻り値

BOOLEAN

ファンクション

makeNode(n DOMNotation)

戻り値

DOMNode

ファンクション

getPublicId(n DOMNotation)

戻り値

VARCHAR2

ファンクション

`getSystemId(n DOMNotation)`

戻り値

VARCHAR2

Processing Instruction メソッド

ファンクション

`isNull(pi DOMProcessingInstruction)`

戻り値

BOOLEAN

ファンクション

`makeNode(pi DOMProcessingInstruction)`

戻り値

DOMNode

ファンクション

`getData(pi DOMProcessingInstruction)`

戻り値

VARCHAR2

ファンクション

`getTarget(pi DOMProcessingInstruction)`

戻り値

VARCHAR2

プロシージャ

`setData(pi DOMProcessingInstruction, data IN VARCHAR2)`

Text メソッド

ファンクション

`isNull(t DOMText)`

戻り値

BOOLEAN

ファンクション

`makeNode(t DOMText)`

戻り値

DOMNode

ファンクション

`splitText(t DOMText, offset IN NUMBER)`

戻り値

DOMText

Document メソッド

ファンクション

`isNull(doc DOMDocument)`

戻り値

BOOLEAN

ファンクション

`makeNode(doc DOMDocument)`

戻り値

DOMNode

ファンクション

newDOMDocument

戻り値

DOMDocument

ファンクション

getVersion(doc DOMDocument)

戻り値

VARCHAR2

プロシージャ

setVersion(doc DOMDocument, version VARCHAR2)

ファンクション

getCharset(doc DOMDocument)

戻り値

VARCHAR2

プロシージャ

setCharset(doc DOMDocument, charset VARCHAR2)

ファンクション

getStandalone(doc DOMDocument)

戻り値

VARCHAR2

プロシージャ

setStandalone(doc DOMDocument, value VARCHAR2)

プロシージャ

writeToFile(doc DOMDocument, fileName VARCHAR2)

プロシージャ

writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2)

プロシージャ

writeToClob(doc DOMDocument, cl IN OUT CLOB)

プロシージャ

writeToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2)

プロシージャ

writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2)

プロシージャ

writeToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2)

プロシージャ

writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2)

プロシージャ

writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2)

プロシージャ

writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB)

プロシージャ

writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2)

プロシージャ

writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2)

プロシージャ

writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2)

ファンクション

getDoctype(doc DOMDocument)

戻り値

DOMDocumentType

ファンクション

`getImplementation(doc DOMDocument)`

戻り値

DOMImplementation

ファンクション

`getDocumentElement(doc DOMDocument)`

戻り値

DOMElement

ファンクション

`createElement(doc DOMDocument, tagName IN VARCHAR2)`

戻り値

DOMElement

ファンクション

`createDocumentFragment(doc DOMDocument)`

戻り値

DOMDocumentFragment

ファンクション

`createTextNode(doc DOMDocument, data IN VARCHAR2)`

戻り値

DOMText

ファンクション

`createComment(doc DOMDocument, data IN VARCHAR2)`

戻り値

DOMComment

ファンクション

```
createCDATASection(doc DOMDocument, data IN VARCHAR2)
```

戻り値

DOMCDATASection

ファンクション

```
createProcessingInstruction(doc DOMDocument, target IN VARCHAR2, data IN VARCHAR2)
```

戻り値

DOMProcessingInstruction

ファンクション

```
createAttribute(doc DOMDocument, name IN VARCHAR2)
```

戻り値

DOMAttr

ファンクション

```
createEntityReference(doc DOMDocument, name IN VARCHAR2)
```

戻り値

DOMEntityReference

ファンクション

```
getElementsByTagName(doc DOMDocument, tagname IN VARCHAR2)
```

戻り値

DOMNodeList

メソッド・プロトタイプ

Node メソッド

ファンクション

`isNull(n DOMNode)`

戻り値

BOOLEAN

指定した DOM ノードが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM ノードが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(n DOMNode) RETURN BOOLEAN;
```

パラメータ

`n` (IN) - チェックする DOM ノード

ファンクション

`makeAttr(n DOMNode)`

戻り値

DOMAttr

用途

与えられた DOM ノードを DOM 属性にキャストします。

構文

```
FUNCTION makeAttr(n DOMNode) RETURN DOMAttr;
```

パラメータ

`n` (IN) - キャストする DOM ノード

ファンクション

makeCDataSection(n DOMNode)

戻り値

DOMCDataSection

用途

与えられた DOM ノードを DOMCDATA セクションにキャストします。

構文

```
FUNCTION makeCDataSection(n DOMNode) RETURN DOMCDataSection;
```

パラメータ

n　　(IN) - キャストする DOM ノード

ファンクション

makeCharacterData(n DOMNode)

戻り値

DOMCharacterData

用途

与えられた DOM ノードを DOM 文字データにキャストします。

構文

```
FUNCTION makeCharacterData(n DOMNode) RETURN DOMCharacterData;
```

パラメータ

n　　(IN) - キャストする DOM ノード

ファンクション

makeComment(n DOMNode)

戻り値

DOMComment

用途

与えられた DOM ノードを DOM コメントにキャストします。

構文

```
FUNCTION makeComment(n DOMNode) RETURN DOMComment;
```

パラメータ

n　　(IN) - キャストする DOM ノード

ファンクション

makeDocumentFragment(n DOMNode)

戻り値

DOMDocumentFragment

用途

与えられた DOM ノードを DOM ドキュメント・フラグメントにキャストします。

構文

```
FUNCTION makeDocumentFragment(n DOMNode) RETURN DOMDocumentFragment;
```

パラメータ

n　　(IN) - キャストする DOM ノード

ファンクション

makeDocumentType(n DOMNode)

戻り値

DOMDocumentType

用途

与えられた DOM ノードを DOM ドキュメント・タイプにキャストします。

構文

```
FUNCTION makeDocumentType(n DOMNode) RETURN DOMDocumentType;
```

パラメータ

n　　(IN) - キャストする DOM ノード

ファンクション

makeElement(n DOMNode)

戻り値

DOMElement

用途

与えられた DOM ノードを DOM 要素にキャストします。

構文

```
FUNCTION makeElement(n DOMNode) RETURN DOMElement;
```

パラメータ

n　　(IN) - キャストする DOM ノード

ファンクション

makeEntity(n DOMNode)

戻り値

DOMEntity

用途

与えられた DOM ノードを DOM エンティティにキャストします。

構文

```
FUNCTION makeEntity(n DOMNode) RETURN DOMEntity;
```

パラメータ

n　　(IN) - キャストする DOM ノード

ファンクション

makeEntityReference(n DOMNode)

戻り値

DOMEntityReference

用途

与えられた DOM ノードを DOM エンティティ・リファレンスにキャストします。

構文

```
FUNCTION makeEntityReference(n DOMNode) RETURN DOMEntityReference;
```

パラメータ

n　　(IN) - キャストする DOM ノード

ファンクション

makeNotation(n DOMNode)

戻り値

DOMNotation

用途

与えられた DOM ノードを DOM 記法にキャストします。

構文

```
FUNCTION makeNotation(n DOMNode) RETURN DOMNotation;
```

パラメータ

n　　(IN) - キャストする DOM ノード

ファンクション

makeProcessingInstruction(n DOMNode)

戻り値

DOMProcessingInstruction

用途

与えられた DOM ノードを DOM 処理命令にキャストします。

構文

```
FUNCTION makeProcessingInstruction(n DOMNode) RETURN DOMProcessingInstruction;
```

パラメータ

n　　(IN) - キャストする DOM ノード

ファンクション

makeText(n DOMNode)

戻り値

DOMText

用途

与えられた DOM ノードを DOM テキストにキャストします。

構文

```
FUNCTION makeText(n DOMNode) RETURN DOMText;
```

パラメータ

n　　(IN) - キャストする DOM ノード

ファンクション

makeDocument(n DOMNode)

戻り値

DOMDocument

用途

与えられた DOM ノードを DOM ドキュメントにキャストします。

構文

```
FUNCTION makeDocument(n DOMNode) RETURN DOMDocument;
```

パラメータ

n　　(IN) - キャストする DOM ノード

プロシージャ

writeToFile(n DOMNode, fileName VARCHAR2)

用途

データベース・キャラクタ・セットを使用して、指定したファイルに XML ノードを書き込みます。

構文

```
PROCEDURE writeToFile(n DOMNode, fileName VARCHAR2);
```

パラメータ

n (IN) - DOM ノード

fileName (IN) - 書込み先のファイル

プロシージャ

writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2)

用途

データベース・キャラクタ・セットを使用して、指定したバッファに XML ノードを書き込みます。

構文

```
PROCEDURE writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2);
```

パラメータ

n (IN) - DOM ノード

buffer (OUT) - 書込み先のバッファ

プロシージャ

writeToClob(n DOMNode, cl IN OUT CLOB)

用途

データベース・キャラクタ・セットを使用して、指定した CLOB に XML ノードを書き込みます。

構文

```
PROCEDURE writeToClob(n DOMNode, cl IN OUT CLOB);
```

パラメータ

n (IN) - DOM ノード

cl (OUT) - 書込み先の CLOB

プロシージャ

writeToFile(n DOMNode, fileName VARCHAR2, charset VARCHAR2)

用途

与えられたキャラクタ・セットを使用して、指定したファイルに XML ノードを書き込みます。

構文

```
PROCEDURE writeToFile(n DOMNode, fileName VARCHAR2, charset VARCHAR2);
```

パラメータ

n (IN) - DOM ノード

fileName (IN) - 書込み先のファイル

charset (IN) - キャラクタ・セット

プロシージャ

writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2, charset VARCHAR2)

用途

与えられたキャラクタ・セットを使用して、指定したバッファに XML ノードを書き込みます。

構文

```
PROCEDURE writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2, charset VARCHAR2);
```

パラメータ

n (IN) - DOM ノード

buffer (OUT) - 書込み先のバッファ

charset (IN) - キャラクタ・セット

プロシージャ

`writeToClob(n DOMNode, cl IN OUT CLOB, charset VARCHAR2)`

用途

与えられたキャラクタ・セットを使用して、指定した CLOB に XML ノードを書き込みます。

構文

```
PROCEDURE writeToClob(n DOMNode, cl IN OUT CLOB, charset VARCHAR2);
```

パラメータ

`n` (IN) - DOM ノード

`cl` (OUT) - 書込み先の CLOB

`charset` (IN) - キャラクタ・セット

Named Node Map メソッド

ファンクション

`isNull(nnm DOMNamedNodeMap)`

戻り値

BOOLEAN

与えられた DOM NamedNodeMap が NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM NamedNodeMap が NULL かどうかをチェックします。

構文

```
FUNCTION isNull(nnm DOMNamedNodeMap) RETURN BOOLEAN;
```

パラメータ

`nnm` (IN) - チェックする DOMNamedNodeMap

Node List メソッド

ファンクション

isNull(nl DOMNodeList)

戻り値

BOOLEAN

与えられた DOM ノード・リストが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM ノード・リストが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(nl DOMNodeList) RETURN BOOLEAN;
```

パラメータ

nl (IN) - チェックする DOM ノード・リスト

Attr メソッド

ファンクション

isNull(a DOMAttr)

戻り値

BOOLEAN

与えられた DOM 属性が NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM 属性が NULL かどうかをチェックします。

構文

```
FUNCTION isNull(a DOMAttr) RETURN BOOLEAN;
```

パラメータ

a (IN) - チェックする DOM 属性

ファンクション

makeNode(a DOMAttr)

戻り値

DOMNode

用途

与えられた DOM 属性を DOM ノードにキャストします。

構文

```
FUNCTION makeNode(a DOMAttr) RETURN DOMNode;
```

パラメータ

a　　(IN) - キャストする DOM ノード

ファンクション

getQualifiedName(a DOMAttr)

戻り値

VARCHAR2

修飾名

用途

DOM 属性の修飾名を戻します。

構文

```
FUNCTION getQualifiedName(a DOMAttr) RETURN VARCHAR2;
```

パラメータ

a　　(IN) - DOM 属性

ファンクション

getNamespace(a DOMAttr)

戻り値

VARCHAR2

名前空間

用途

DOM 属性の名前空間を戻します。

構文

```
FUNCTION getNamespace(a DOMAttr) RETURN VARCHAR2;
```

パラメータ

a　　(IN) - DOM 属性

ファンクション

getLocalName(a DOMAttr)

戻り値

VARCHAR2

ローカル名

用途

DOM 属性のローカル名を戻します。

構文

```
FUNCTION getLocalName(a DOMAttr) RETURN VARCHAR2;
```

パラメータ

a　　(IN) - DOM 属性

ファンクション

getExpandedName(a DOMAttr)

戻り値

VARCHAR2

拡張名

用途

DOM 属性の拡張名を戻します。

構文

```
FUNCTION getExpandedName(a DOMAttr) RETURN VARCHAR2;
```

パラメータ

a　　(IN) - DOM 属性

C Data Section メソッド

ファンクション

isNull(cds DOMCDataSection)

戻り値

BOOLEAN

与えられた DOM CDATA セクションが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM CDATA セクションが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(cds DOMCDataSection) RETURN BOOLEAN;
```

パラメータ

cds　　(IN) - チェックする DOM CDATA セクション

ファンクション

makeNode(cds DOMCDataSection)

戻り値

DOMNode

用途

与えられた DOM CDATA セクションを DOM ノードにキャストします。

構文

```
FUNCTION makeNode(cds DOMCDataSection) RETURN DOMNode;
```

パラメータ

cds　　(IN) - キャストする DOM CDATA セクション

Character Data メソッド

ファンクション

isNull(cd DOMCharacterData)

戻り値

BOOLEAN

与えられた DOM 文字データが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM 文字データが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(cd DOMCharacterData) RETURN BOOLEAN;
```

パラメータ

cd　　(IN) - チェックする DOM 文字データ

ファンクション

makeNode(cd DOMCharacterData)

戻り値

DOMNode

用途

与えられた DOM 文字データを DOM ノードにキャストします。

構文

```
FUNCTION makeNode(cd DOMCharacterData) RETURN DOMNode;
```

パラメータ

cd　　(IN) - キャストする DOM 文字データ

Comment メソッド

ファンクション

isNull(com DOMComment)

戻り値

BOOLEAN

与えられた DOM コメントが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM コメントが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(com DOMComment) RETURN BOOLEAN;
```

パラメータ

com　　(IN) - チェックする DOM コメント

ファンクション

makeNode(com DOMComment)

戻り値

DOMNode

用途

与えられた DOM コメントを DOM ノードにキャストします。

構文

```
FUNCTION makeNode(com DOMComment) RETURN DOMNode;
```

パラメータ

com　　(IN) - キャストする DOM コメント

DOM Implementation メソッド

ファンクション

isNull(di DOMImplementation)

戻り値

BOOLEAN

与えられた DOM インプリメンテーションが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM インプリメンテーションが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(di DOMImplementation) RETURN BOOLEAN;
```

パラメータ

di　　(IN) - チェックする DOM インプリメンテーション

Document Fragment メソッド

ファンクション

`isNull(df DOMDocumentFragment)`

戻り値

BOOLEAN

与えられた DOM ドキュメント・フラグメントが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM ドキュメント・フラグメントが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(df DOMDocumentFragment) RETURN BOOLEAN;
```

パラメータ

`df` (IN) - チェックする DOM ドキュメント・フラグメント

ファンクション

`makeNode(df DOMDocumentFragment)`

戻り値

DOMNode

用途

与えられた DOM ドキュメント・フラグメントを DOMNode にキャストします。

構文

```
FUNCTION makeNode(df DOMDocumentFragment) RETURN DOMNode;
```

パラメータ

`df` (IN) - キャストする DOM ドキュメント・フラグメント

Document Type メソッド

ファンクション

`isNull(dt DOMDocumentType)`

戻り値

BOOLEAN

与えられた DOM ドキュメント・タイプが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM ドキュメント・タイプが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(dt DOMDocumentType) RETURN BOOLEAN;
```

パラメータ

`dt` (IN) - チェックする DOM ドキュメント・タイプ

ファンクション

`makeNode(dt DOMDocumentType)`

戻り値

DOMNode

用途

与えられた DOM ドキュメント・タイプを DOM ノードにキャストします。

構文

```
FUNCTION makeNode(dt DOMDocumentType) RETURN DOMNode;
```

パラメータ

`dt` (IN) - キャストする DOM ドキュメント・タイプ

ファンクション

`findEntity(dt DOMDocumentType, name VARCHAR2, par BOOLEAN)`

戻り値

DOMEntity（検索された場合）

用途

与えられた DTD のエンティティを検索します。

構文

```
FUNCTION findEntity(dt DOMDocumentType, name VARCHAR2, par BOOLEAN) RETURN
DOMEntity;
```

パラメータ

dt	(IN)	- DTD
name	(IN)	- 検索するエンティティ
par	(IN)	- TRUE - パラメータ・エンティティ、FALSE - 標準エンティティ

ファンクション

`findNotation(dt DOMDocumentType, name VARCHAR2)`

戻り値

DOMNotation（検索された場合）

用途

与えられた DTD の記法を検索します。

構文

```
FUNCTION findNotation(dt DOMDocumentType, name VARCHAR2) RETURN DOMNotation;
```

パラメータ

dt	(IN)	- DTD
name	(IN)	- 検索する記法

ファンクション

`getPublicId(dt DOMDocumentType)`

戻り値

VARCHAR2

公開識別子

用途

与えられた DTD の公開識別子を戻します。

構文

```
FUNCTION getPublicId(dt DOMDocumentType) RETURN VARCHAR2;
```

パラメータ

`dt` `(IN)` - DTD

ファンクション

`getSystemId(dt DOMDocumentType)`

戻り値

VARCHAR2

システム識別子

用途

与えられた DTD のシステム識別子を戻します。

構文

```
FUNCTION getSystemId(dt DOMDocumentType) RETURN VARCHAR2;
```

パラメータ

`dt` `(IN)` - DTD

プロシージャ

```
writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2)
```

用途

データベース・キャラクタ・セットを使用して、指定したファイルに DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2);
```

パラメータ

dt (IN) - DOM ドキュメント・タイプ

fileName (IN) - 書込み先のファイル

プロシージャ

```
writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2)
```

用途

データベース・キャラクタ・セットを使用して、指定したバッファに DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2);
```

パラメータ

dt (IN) - DOM ドキュメント・タイプ

buffer (OUT) - 書込み先のバッファ

プロシージャ

```
writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB)
```

用途

データベース・キャラクタ・セットを使用して、指定した CLOB に DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB);
```

パラメータ

dt (IN) - DOM ドキュメント・タイプ

cl (OUT) - 書込み先の CLOB

プロシージャ

```
writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2, charset VARCHAR2)
```

用途

与えられたキャラクタ・セットを使用して、指定したファイルに DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2, charset
VARCHAR2);
```

パラメータ

dt (IN) - DOM ドキュメント・タイプ

fileName (IN) - 書込み先のファイル

charset (IN) - キャラクタ・セット

プロシージャ

```
writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2, charset VARCHAR2)
```

用途

与えられたキャラクタ・セットを使用して、指定したバッファに DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2,
charset VARCHAR2);
```

パラメータ

dt (IN) - DOM ドキュメント・タイプ

buffer (OUT) - 書込み先のバッファ

charset (IN) - キャラクタ・セット

プロシージャ

```
writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB, charset VARCHAR2)
```

用途

与えられたキャラクタ・セットを使用して、指定した CLOB に DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB, charset
VARCHAR2);
```

パラメータ

dt (IN) - DOM ドキュメント・タイプ

cl (OUT) - 書込み先の CLOB

charset (IN) - キャラクタ・セット

Element メソッド

ファンクション

```
isNull(elem DOMElement)
```

戻り値

BOOLEAN

与えられた DOM 要素が NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM 要素が NULL かどうかをチェックします。

構文

```
FUNCTION isNull(elem DOMElement) RETURN BOOLEAN;
```

パラメータ

elem (IN) - チェックする DOM 要素

ファンクション

makeNode(elem DOMElement)

戻り値

DOMNode

用途

与えられた DOM 要素を DOM ノードにキャストします。

構文

```
FUNCTION makeNode(elem DOMElement) RETURN DOMNode;
```

パラメータ

elem (IN) - キャストする DOM 要素

ファンクション

getQualifiedName(elem DOMElement)

戻り値

VARCHAR2

修飾名

用途

DOM 要素の修飾名を戻します。

構文

```
FUNCTION getQualifiedName(elem DOMElement) RETURN VARCHAR2;
```

パラメータ

elem (IN) - DOM 要素

ファンクション

getNamespace(elem DOMElement)

戻り値

VARCHAR2

名前空間

用途

DOM 要素の名前空間を戻します。

構文

```
FUNCTION getNamespace(elem DOMElement) RETURN VARCHAR2;
```

パラメータ

elem　　(IN) - DOM 要素

ファンクション

getLocalName(elem DOMElement)

戻り値

VARCHAR2

ローカル名

用途

DOM 要素のローカル名を戻します。

構文

```
FUNCTION getLocalName(elem DOMElement) RETURN VARCHAR2;
```

パラメータ

elem　　(IN) - DOM 要素

ファンクション

`getExpandedName(elem DOMElement)`

戻り値

VARCHAR2

拡張名

用途

DOM 要素の拡張名を戻します。

構文

```
FUNCTION getExpandedName(elem DOMElement) RETURN VARCHAR2;
```

パラメータ

`elem` (IN) - DOM 要素

ファンクション

`getChildrenByTagName(elem DOMElement, name varchar2)`

戻り値

DOMNodeList

与えられたタグ名を持つ子

用途

与えられたタグ名を持つ DOM 要素の子を戻します。

構文

```
FUNCTION getChildrenByTagName(elem DOMElement, name varchar2) RETURN DOMNodeList;
```

パラメータ

`elem` (IN) - DOM 要素

`name` (IN) - タグ名（* はすべてのタグに一致する）

ファンクション

getElementsByTagName(elem DOMElement, name varchar2, ns VARCHAR2)

戻り値

DOMNodeList

与えられたタグ名および名前空間を持つ要素

用途

与えられたタグ名および名前空間を持つ DOM 要素の要素の子を戻します。

構文

```
FUNCTION getElementsByTagName(elem DOMElement, name varchar2, ns VARCHAR2 ) RETURN
DOMNodeList;
```

パラメータ

elem (IN) - DOM 要素

name (IN) - タグ名（* はすべてのタグに一致する）

ns (IN) - 名前空間

ファンクション

getChildrenByTagName(elem DOMElement, name varchar2, ns VARCHAR2)

戻り値

DOMNodeList

与えられたタグ名および名前空間を持つ子

用途

与えられたタグ名および名前空間を持つ DOM 要素の子を戻します。

構文

```
FUNCTION getChildrenByTagName(elem DOMElement, name varchar2, ns VARCHAR2 ) RETURN
DOMNodeList;
```

パラメータ

elem (IN) - DOM 要素

name (IN) - タグ名（* はすべてのタグに一致する）

ns (IN) - 名前空間

ファンクション

`resolveNamespacePrefix(elem DOMElement, prefix VARCHAR2)`

戻り値

VARCHAR2

解決済名前空間

用途

与えられた名前空間の接頭辞を変換します。

構文

```
FUNCTION resolveNamespacePrefix(elem DOMElement, prefix VARCHAR2) RETURN VARCHAR2;
```

パラメータ

`elem` (IN) - DOM 要素

`prefix` (IN) - 名前空間の接頭辞

Entity メソッド

ファンクション

`isNull(ent DOMEntity)`

戻り値

BOOLEAN

与えられた DOM エンティティが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM エンティティが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(ent DOMEntity) RETURN BOOLEAN;
```

パラメータ

`ent` (IN) - チェックする DOM エンティティ

ファンクション

makeNode(ent DOMEntity)

戻り値

DOMNode

用途

与えられた DOM エンティティを DOM ノードにキャストします。

構文

```
FUNCTION makeNode(ent DOMEntity) RETURN DOMNode;
```

パラメータ

ent　　(IN) - キャストする DOM エンティティ

Entity Reference メソッド

ファンクション

isNull(eref DOMEntityReference)

戻り値

BOOLEAN

与えられた DOM エンティティ・リファレンスが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM エンティティ・リファレンスが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(eref DOMEntityReference) RETURN BOOLEAN;
```

パラメータ

eref　　(IN) - チェックする DOM エンティティ・リファレンス

ファンクション

makeNode(eref DOMEntityReference)

戻り値

DOMNode

用途

与えられた DOM エンティティ・リファレンスを DOM ノードにキャストします。

構文

```
FUNCTION makeNode(eref DOMEntityReference) RETURN DOMNode;
```

パラメータ

eref　　(IN) - キャストする DOM エンティティ・リファレンス

Notation メソッド

ファンクション

isNull(n DOMNotation)

戻り値

BOOLEAN

与えられた DOM 記法が NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM 記法が NULL かどうかをチェックします。

構文

```
FUNCTION isNull(n DOMNotation) RETURN BOOLEAN;
```

パラメータ

n　　(IN) - チェックする DOM 記法

ファンクション

makeNode(n DOMNotation)

戻り値

DOMNode

用途

与えられた DOM 記法を DOM ノードにキャストします。

構文

```
FUNCTION makeNode(n DOMNotation) RETURN DOMNode;
```

パラメータ

n　　(IN) - キャストする DOM 記法

Processing Instruction メソッド

ファンクション

isNull(pi DOMProcessingInstruction)

戻り値

BOOLEAN

与えられた DOM 処理命令が NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM 処理命令が NULL かどうかをチェックします。

構文

```
FUNCTION isNull(pi DOMProcessingInstruction) RETURN BOOLEAN;
```

パラメータ

pi　　(IN) - チェックする DOM 処理命令

ファンクション

makeNode(pi DOMProcessingInstruction)

戻り値

DOMNode

用途

与えられた DOM 処理命令を DOM ノードにキャストします。

構文

```
FUNCTION makeNode(pi DOMProcessingInstruction) RETURN DOMNode;
```

パラメータ

pi　　(IN) - キャストする DOM 処理命令

Text メソッド

ファンクション

isNull(t DOMText)

戻り値

BOOLEAN

与えられた DOM テキストが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM テキストが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(t DOMText) RETURN BOOLEAN;
```

パラメータ

t　　(IN) - チェックする DOM テキスト

ファンクション

makeNode(t DOMText)

戻り値

DOMNode

用途

与えられた DOM テキストを DOM ノードにキャストします。

構文

```
FUNCTION makeNode(t DOMText) RETURN DOMNode;
```

パラメータ

t　　(IN) - キャストする DOM テキスト

Document メソッド

ファンクション

isNull(doc DOMDocument)

戻り値

BOOLEAN

与えられた DOM ドキュメントが NULL かどうかを戻します。TRUE の場合は NULL、FALSE の場合は NOT NULL です。

用途

与えられた DOM ドキュメントが NULL かどうかをチェックします。

構文

```
FUNCTION isNull(doc DOMDocument) RETURN BOOLEAN;
```

パラメータ

doc　　(IN) - チェックする DOM ドキュメント

ファンクション

makeNode(doc DOMDocument)

戻り値

DOMNode

用途

与えられた DOM ドキュメントを DOM ノードにキャストします。

構文

```
FUNCTION makeNode(doc DOMDocument) RETURN DOMNode;
```

パラメータ

doc　　(IN) - キャストする DOM ドキュメント

ファンクション

newDOMDocument

戻り値

DOMDocument

新しい DOM ドキュメント・インスタンス

用途

新しい DOM ドキュメント・インスタンスを戻します。

構文

```
FUNCTION newDOMDocument RETURN DOMDocument;
```

パラメータ

なし

プロシージャ

freeDocument(doc DOMDocument)

用途

ドキュメント・オブジェクトを解放します。

構文

```
PROCEDURE freeDocument(doc DOMDocument)
```

パラメータ

doc　　(IN) - DOM ドキュメント

ファンクション

getVersion(doc DOMDocument)

戻り値

VARCHAR2

バージョン情報

用途

XML ドキュメントのバージョン情報を取得します。

構文

```
FUNCTION getVersion(doc DOMDocument) RETURN VARCHAR2;
```

パラメータ

doc　　(IN) - DOM ドキュメント

プロシージャ

setVersion(doc DOMDocument, version VARCHAR2)

用途

XML ドキュメントのバージョン情報を設定します。

構文

```
PROCEDURE setVersion(doc DOMDocument, version VARCHAR2);
```

パラメータ

doc　(IN) - DOM ドキュメント

version　(IN) - バージョン情報

ファンクション

getCharset(doc DOMDocument)

戻り値

VARCHAR2

XML ドキュメントのキャラクタ・セット

用途

XML ドキュメントのキャラクタ・セットを取得します。

構文

```
FUNCTION getCharset(doc DOMDocument) RETURN VARCHAR2;
```

パラメータ

doc　(IN) - DOM ドキュメント

プロシージャ

setCharset(doc DOMDocument, charset VARCHAR2)

用途

XML ドキュメントのキャラクタ・セットを設定します。

構文

```
PROCEDURE setCharset(doc DOMDocument, charset VARCHAR2);
```

パラメータ

doc　(IN) - DOM ドキュメント

charset　(IN) - キャラクタ・セット

ファンクション

getStandalone(doc DOMDocument)

戻り値

VARCHAR2

スタンドアロン情報

用途

XML ドキュメントのスタンドアロン情報を取得します。

構文

```
FUNCTION getStandalone(doc DOMDocument) RETURN VARCHAR2;
```

パラメータ

doc (IN) - DOM ドキュメント

プロシージャ

setStandalone(doc DOMDocument, value VARCHAR2)

用途

XML ドキュメントのスタンドアロン情報を設定します。

構文

```
PROCEDURE setStandalone(doc DOMDocument, value VARCHAR2);
```

パラメータ

doc (IN) - DOM ドキュメント

value (IN) - スタンドアロン情報

プロシージャ

writeToFile(doc DOMDocument, fileName VARCHAR2)

用途

データベース・キャラクタ・セットを使用して、指定したファイルに XML ドキュメントを書き込みます。

構文

```
PROCEDURE writeToFile(doc DOMDocument, fileName VARCHAR2);
```

パラメータ

doc (IN) - DOM ドキュメント

fileName (IN) - 書込み先のファイル

プロシージャ

```
writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2)
```

用途

データベース・キャラクタ・セットを使用して、指定したバッファに XML ドキュメントを書き込みます。

構文

```
PROCEDURE writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2);
```

パラメータ

doc (IN) - DOM ドキュメント

buffer (OUT) - 書込み先のバッファ

プロシージャ

```
writeToClob(doc DOMDocument, cl IN OUT CLOB)
```

用途

データベース・キャラクタ・セットを使用して、指定した CLOB に XML ドキュメントを書き込みます。

構文

```
PROCEDURE writeToClob(doc DOMDocument, cl IN OUT CLOB);
```

パラメータ

doc (IN) - DOM ドキュメント

cl (OUT) - 書込み先の CLOB

プロシージャ

writeToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2)

用途

与えられたキャラクタ・セットを使用して、指定したファイルに XML ドキュメントを書き込みます。

構文

```
PROCEDURE writeToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2);
```

パラメータ

doc (IN) - DOM ドキュメント

fileName (IN) - 書込み先のファイル

charset (IN) - キャラクタ・セット

プロシージャ

writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2)

用途

与えられたキャラクタ・セットを使用して、指定したバッファに XML ドキュメントを書き込みます。

構文

```
PROCEDURE writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2);
```

パラメータ

doc (IN) - DOM ドキュメント

buffer (OUT) - 書込み先のバッファ

charset (IN) - キャラクタ・セット

プロシージャ

writeToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2)

用途

与えられたキャラクタ・セットを使用して、指定した CLOB に XML ドキュメントを書き込みます。

構文

```
PROCEDURE writeToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2);
```

パラメータ

doc (IN) - DOM ドキュメント

cl (OUT) - 書込み先の CLOB

charset (IN) - キャラクタ・セット

プロシージャ

```
writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2)
```

用途

データベース・キャラクタ・セットを使用して、指定したファイルに外部 DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2);
```

パラメータ

doc (IN) - DOM ドキュメント

fileName (IN) - 書込み先のファイル

プロシージャ

```
writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2)
```

用途

データベース・キャラクタ・セットを使用して、指定したバッファに外部 DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2);
```

パラメータ

doc (IN) - DOM ドキュメント

buffer (OUT) - 書込み先のバッファ

プロシージャ

```
writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB)
```

用途

データベース・キャラクタ・セットを使用して、指定した CLOB に外部 DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB);
```

パラメータ

doc　(IN) - DOM ドキュメント

cl　(OUT) - 書込み先の CLOB

プロシージャ

```
writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2)
```

用途

与えられたキャラクタ・セットを使用して、指定したファイルに外部 DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2, charset
VARCHAR2);
```

パラメータ

doc　(IN) - DOM ドキュメント

fileName　(IN) - 書込み先のファイル

charset　(IN) - キャラクタ・セット

プロシージャ

```
writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2)
```

用途

与えられたキャラクタ・セットを使用して、指定したバッファに外部 DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset
VARCHAR2);
```

パラメータ

doc (IN) - DOM ドキュメント

buffer (OUT) - 書込み先のバッファ

charset (IN) - キャラクタ・セット

プロシージャ

```
writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2)
```

用途

与えられたキャラクタ・セットを使用して、指定した CLOB に外部 DTD を書き込みます。

構文

```
PROCEDURE writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2);
```

パラメータ

doc (IN) - DOM ドキュメント

cl (OUT) - 書込み先の CLOB

charset (IN) - キャラクタ・セット

org.w3c.dom.Attr インタフェース

public interface Attr extends Node

Attr インタフェースは、要素オブジェクトの属性を表します。通常、属性の許容値は、文書型定義で定義されます。

属性オブジェクトはノード・インタフェースを継承しますが、実際にはこのオブジェクトが記述する要素の子ノードではないため、DOM では、このオブジェクトをドキュメント・ツリーの一部とは考えません。このため、ノードの属性 parentNode、previousSibling および nextSibling は、属性オブジェクトに対し、NULL 値を持ちます。DOM は、属性は要素のプロパティであると考えられています（対応する要素とは別のものと考えられているわけではありません）。このことから、与えられた型のすべての要素に対応するデフォルトの属性として、このような機能を実装すると効果的です。また、属性ノードは、ドキュメント・フラグメントの直下の子になることはできません。ただし、このノードをドキュメント・フラグメントに含まれる要素ノードに対応付けることはできます。つまり、属性ノードには、Node インタフェースを継承する他のオブジェクトとの共通部分もありますが、両者は別のものであることを認識する必要があります。

属性の有効値は、次のように決定します。この属性に任意の値が明示的に割り当てられている場合は、その値が属性の有効値です。また、この属性に対する宣言があり、その宣言にデフォルト値が含まれている場合は、そのデフォルト値が属性の有効値です。それ以外の場合は、明示的に追加されない限り、構造モデルのこの要素に属性は存在しません。また、属性インスタンスの nodeValue 属性を使用して、文字列の属性値を取得することもできます。XML では、属性値がエンティティ参照を含むことができる場合に、属性ノードの子ノードは、エンティティ参照が拡張されない表現を提供できます。これらの子ノードは、テキスト・ノードまたはエンティティ・リファレンス・ノードのどちらかになります。属性の型が不明なため、トークン化された属性値はありません。

抽象メソッド（一覧）

getName

この属性の名前を戻します。

getSpecified

元のドキュメントでこの属性に値が明示的に与えられている場合は TRUE を、それ以外の場合は FALSE を戻します。

getValue

検索時に、属性の値が文字列として戻されます。

setValue

適切な値を入力します。

抽象メソッド（詳細）

getName

```
public abstract String getName()
```

この属性の名前を戻します。

getSpecified

```
public abstract boolean getSpecified()
```

元のドキュメントでこの属性に値が明示的に与えられている場合は TRUE を、それ以外の場合は FALSE を戻します。ユーザーではなく、実装がこの属性を操作することに注意してください。ユーザーが属性の値を変更した場合（結果的に、デフォルト値と同じ値になる場合でも）、specified フラグが自動的に TRUE になります。属性を DTD のデフォルト値として再指定するには、その属性を削除する必要があります。削除後、実装によって、FALSE に設定された specified およびデフォルト値（存在する場合）で新しい属性が使用可能になります。

属性にドキュメントの値が割り当てられている場合、specified は TRUE で、値は割り当てられた値になります。属性にドキュメントの値が割り当てられておらず、DTD のデフォルト値を持つ場合、specified は FALSE で、値は DTD のデフォルト値になります。属性にドキュメントの値が割り当てられておらず、DTD に #IMPLIED が指定されている場合、属性はドキュメントの構造モデルでは使用されません。

getValue

```
public abstract String getValue()
```

検索時に、属性の値が文字列として戻されます。文字および一般的なエンティティ参照は、それらの値に置き換えられます。

これによって、設定時に、解析対象外の文字列の内容を持つテキスト・ノードが作成されます。

setValue

```
public abstract void setValue(String value)
```

org.w3c.dom.CDATASection インタフェース

public interface **CDATASection** extends Text

CDATA セクションは、マークアップであると判断される可能性がある文字を含むテキストのブロックをエスケープするために使用されます。CDATA セクションで認識されるデリミタは、このセクションを終了する "]]>" 文字列のみです。CDATA セクションを、ネストすることはできません。このセクションは、主に、すべてのデリミタをエスケープしなくても、XML の断片部分などのマテリアルを含めるために使用します。

テキスト・ノードの DOMString 属性は、CDATA セクションに含まれるテキストを保持します。これには、CDATA セクション外でエスケープする必要がある文字が含まれる場合があり、シリアライズ化のために選択されたキャラクタ・コード（キャラクタ・セット）によっては、CDATA セクションの一部として書き出すことができない文字があることに注意してください。

CDATASection インタフェースは、Text インタフェースから CharacterData インタフェースを継承します。隣接する CDATA セクション・ノードは、Element.normalize() メソッドを使用してもマージされません。

org.w3c.dom.CharacterData インタフェース

public interface **CharacterData** extends Node

CharacterData インタフェースは、DOM の文字データにアクセスするための属性およびメソッドを使用して、ノードを拡張します。このセットは、これらの属性およびメソッドを使用する各オブジェクトではなく、ここで定義されます。テキストおよびその他のノードは文字データ・ノードからインタフェースを継承しますが、DOM オブジェクトは、このノードに直接対応していません。このインタフェースのすべてのオフセットは、0（ゼロ）から開始します。

抽象メソッド（一覧）

appendData(String)

文字列をノードの文字データの最後に追加します。

DeleteData(int, int)

一定範囲の文字をノードから削除します。

getData()

このインタフェースを実装するノードの文字データを取得します。

getLength()

data および次の substringData メソッドを介して使用可能な文字数を取得します。

insertData(int, String)

指定した文字オフセットに文字列を挿入します。

replaceData(int, int, String)

指定した文字オフセットから始まる文字を、指定した文字列と置き換えます。

setData(String)

substringData(int, int)

一定範囲のデータをノードから抽出します。

抽象メソッド（詳細）

appendData

```
public abstract void appendData(String arg) throws DOMException
```

文字列をノードの文字データの最後に追加します。文字列が正常に追加されると、連続した data と指定した DOM 文字列へのアクセスが可能になります。

パラメータ

`arg` - 追加する DOM 文字列

例外の発生

DOMException

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

deleteData

```
public abstract void deleteData(int offset,
                                int count) throws DOMException
```

一定の範囲の文字をノードから削除します。文字が正常に削除された場合、その変更が data および length に反映されます。

パラメータ

`offset` - 文字削除の開始点とするオフセット

`count` - 削除する文字数。offset と count の合計が length を超える場合は、offset からデータの終わりまでのすべての文字が削除されます。

例外の発生

DOMException

INDEX_SIZE_ERR: 指定した offset が負数であるか、data 内の文字数より多い場合、または指定した count が負数である場合に発生します。
NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。

getData

```
public abstract String getData() throws DOMException
```

このインタフェースを実装するノードの文字データを取得します。DOM インプリメンテーションでは、文字データ・ノードに格納できるデータ量を制限することはできません。ただし、実装制限によって、ノードのデータ全体を単一の DOM 文字列に格納できない場合があります。このような場合、substringData をコールすると、データを適切なピース・サイズで取得できます。

例外の発生

DOMException

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

例外の発生

DOMException

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

getLength

```
public abstract int getLength()
```

data および substringData メソッドを介して使用可能な文字数を取得します。0（ゼロ）の値を持つ（文字データ・ノードが空である）場合もあります。

insertData

```
public abstract void insertData(int offset,
                                String arg) throws DOMException
```

文字列を指定した文字オフセットに挿入します。

パラメータ

`offset` - 挿入点とする文字オフセット

`arg` - 挿入する DOM 文字列

例外の発生

DOMException

INDEX_SIZE_ERR: 指定した offset が負数であるか、data 内の文字数より多い場合に発生します。
NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。

replaceData

```
public abstract void replaceData(int offset,
                                 int count,
                                 String arg) throws DOMException
```

指定した文字オフセットから開まる文字を、指定した文字列で置換します。

パラメータ

`offset` - 置換の開始点とするオフセット

`count` - 置換する文字数。offset と count の合計が length を超える場合は、データの終わりまでのすべての文字が置換されます（これは、同じ範囲で remove メソッドをコールしてから、append メソッドをコールした場合と同様です）。

`arg` - 指定した範囲の文字と置き換える DOM 文字列

例外の発生

DOMException

INDEX_SIZE_ERR: 指定した offset が負数であるか、data 内の文字数より多い場合、または指定した count が負数である場合に発生します。
NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に呼び出されます。

setData

```
public abstract void setData(String data) throws DOMException
```

substringData

```
public abstract String substringData(int offset,
                                     int count) throws DOMException
```

一定の範囲のデータをノードから抽出します。

パラメータ

`offset` - 抽出する部分文字列の開始オフセット

`count` - 抽出する文字数

戻り値

指定した部分文字列。offset と count の合計が length を超える場合は、データの終わりまでのすべての文字が戻されます。

例外の発生

DOMException

INDEX_SIZE_ERR: 指定した offset が負数であるか、data 内の文字数より多い場合、または指定した count が負数である場合に発生します。
DOMSTRING_SIZE_ERR: 指定した範囲のテキストが、DOM 文字列より多い場合に発生します。

org.w3c.dom.Comment インタフェース

public interface **Comment** extends Node

このインタフェースは、コメントの内容（`<!--` から `-->` までのすべての文字）を表します。HTML ツールには完全な SGML コメント構造を実装するものもありますが、このインタフェースは、XML（実際には HTML）におけるコメントの定義であることに注意してください。

org.w3c.dom.Document インタフェース

public interface **Document** extends Node

Document インタフェースは、HTML または XML ドキュメント全体を表します。概念上、このインタフェースはドキュメント・ツリーのルートであり、ドキュメントのデータへのプライマリ・アクセスを提供します。

要素、テキスト・ノード、コメント、処理命令などは、ドキュメントのコンテキスト外では存在できないため、Document インタフェースには、これらのオブジェクトを作成するために必要なファクトリ・メソッドも含まれます。作成されたノード・オブジェクトは、この属性によって、オブジェクトが作成されたドキュメントのコンテキスト内で、オブジェクトとそのドキュメントが対応付けられます。ownerDocument 属性を持ちます。

抽象メソッド（一覧）

createAttribute(String)

与えられた名前の属性を作成します。

createCDATASection(String)

指定した文字列を値に持つ CDATA セクション・ノードを作成します。

createComment(String)

指定した文字列を値に持つコメント・ノードを作成します。

createDocumentFragment()

空のドキュメント・フラグメント・オブジェクトを作成します。

createElement(String)

指定した型の要素を作成します。

createEntityReference(String)

エンティティ・リファレンス・オブジェクトを作成します。

createProcessingInstruction(String, String)

指定した名前およびデータ文字列を値に持つ処理命令ノードを作成します。

createTextNode(String)

指定した文字列を値に持つテキスト・ノードを作成します。

getDocType()

このドキュメントに対応する文書型宣言（「DocumentType」を参照）を取得します。

getDocumentElement()

ドキュメントのルート要素である子ノードへの直接アクセスを可能にする有効な属性です。

getElementsByTagName(String)

与えられたタグ名を持つすべての要素のノード・リストを、ドキュメント・ツリーの前順走査で一致する順に戻します。

getImplementation()

このドキュメントを処理する DOM インプリメンテーション・オブジェクトを取得します。

抽象メソッド（詳細）

createAttribute

`public abstract  createAttribute(String name) throws DOMException`
与えられた名前の属性を作成します。この属性インスタンスは、setAttribute メソッドを使用して、要素に対して設定できます。

パラメータ

`name` - 属性名

戻り値

新しい属性オブジェクト

例外の発生

DOMException

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

createCDATASection

`public abstract  createCDATASection(String data) throws DOMException`
指定した文字列を値に持つ CDATA セクション・ノードを作成します。

パラメータ

`data` - CDATA セクションの内容のデータ

戻り値

新しい CDATA セクション・オブジェクト

例外の発生

DOMException

NOT_SUPPORTED_ERR: このドキュメントが HTML ドキュメントである場合に発生します。

createComment

```
public abstract  createComment(String data)
```

指定した文字列を値に持つコメント・ノードを作成します。

パラメータ

`data` - ノードのデータ

戻り値

新しいコメント・オブジェクト

createDocumentFragment

```
public abstract  createDocumentFragment()
```

空のドキュメント・フラグメント・オブジェクトを作成します。

戻り値

新しいドキュメント・フラグメント

createElement

```
public abstract  createElement(String tagName) throws DOMException
```

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

パラメータ

`tagName` - インスタンス化する要素型の名前。XML では、このパラメータの大文字 / 小文字が区別されます。HTML では、tagName パラメータは、大文字 / 小文字を区別せずに指定できますが、DOM インプリメンテーションによって大文字の正規形にマップする必要があります。

戻り値

新しい要素オブジェクト

例外の発生

DOMException

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

createEntityReference

```
public abstract  createEntityReference(String name) throws DOMException
```

エンティティ・リファレンス・オブジェクトを作成します。

パラメータ

name - 参照するエンティティ名

戻り値

新しいエンティティ・リファレンス・オブジェクト

例外の発生

INVALID_CHARACTER_ERR: 指定した名前が無効な文字を含む場合に発生します。
NOT_SUPPORTED_ERR: このドキュメントが HTML ドキュメントである場合に発生します。

createProcessingInstruction

```
public abstract  createProcessingInstruction(String target,String data) throws
DOMException
```

指定した名前およびデータ文字列を値に持つ処理命令ノードを作成します。

パラメータ

target - 処理命令のターゲット部分

data - ノードのデータ

戻り値

新しい処理命令オブジェクト

例外の発生

DOMException

INVALID_CHARACTER_ERR: 無効な文字を指定した場合に発生します。
NOT_SUPPORTED_ERR: このドキュメントが HTML ドキュメントである場合に発生します。

createTextNode

`public abstract  createTextNode(String data)`
指定した文字列を値に持つテキスト・ノードを作成します。

パラメータ

`data` - ノードのデータ

戻り値

新しいテキスト・オブジェクト

getDoctype

`public abstract  getDoctype()`
このドキュメントに対応する文書型宣言（「DocumentType」を参照）を取得します。HTML ドキュメントおよび文書型宣言を持たない XML ドキュメントの場合は、NULL を戻します。DOM レベル 1 では、文書型宣言の編集はサポートされないため、docType は変更できません。

getDocumentElement

`public abstract  getDocumentElement()`
これは、ドキュメントのルート要素である子ノードへの直接アクセスを可能にする有効な属性です。HTML ドキュメントの場合、これは「HTML」という tagName を持つ要素です。

getElementsByTagName

`public abstract  getElementsByTagName(String tagname)`
与えられたタグ名を持つすべての要素のノード・リストを、ドキュメント・ツリーの前順走査で一致する順に戻します。

パラメータ

`tagname` - 一致するタグ名。特殊値「*」を指定すると、すべてのタグに一致します。

戻り値

一致するすべての要素を含む新しいノード・リスト・オブジェクト

getImplementation

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

org.w3c.dom.DocumentFragment インタフェース

public interface **DocumentFragment** extends Node

ドキュメント・フラグメントは、軽量または最小のドキュメント・オブジェクトです。ドキュメント・ツリーの一部を抽出したり、ドキュメント・フラグメントを新しく作成することはよくあります。断片部分を移動してドキュメントを再配置する場合を考えてみます。このような断片部分を保持できるオブジェクトを持つことが最適であり、このためにはノードを使用することが非常に合理的です。実際に、ドキュメント・オブジェクトがこの役割を果たすことは可能ですが、基になる実装によっては、ドキュメント・オブジェクトは重量級のオブジェクトになる場合があります。ここで必要なのは、非常に軽量なオブジェクトです。ドキュメント・フラグメントは、それに該当するオブジェクトです。

また、ノードを別の Node の子として挿入するなどの様々な操作では、ドキュメント・フラグメント・オブジェクトを引数として使用します。その場合、ドキュメント・フラグメントのすべての子ノードは、このノードの子リストに移動されます。

ドキュメント・フラグメント・ノードの子は、ドキュメントの構造を定義するサブツリーの最上位を表す 0（ゼロ）以上のノードです。ドキュメント・フラグメント・ノードは、整形式の XML ドキュメントである必要はありません（ただし、このノードは、解析対象の整形式の XML エンティティの規則に従う必要があり、複数の最上位ノードを持つことができます）。たとえば、ドキュメント・フラグメントが 1 つの子のみを持ち、その子がテキスト・ノードである場合があります。このような構造モデルは、HTML ドキュメントも整形式の XML ドキュメントも表しません。

ドキュメント・フラグメントがドキュメント（または、実際には、子を持つことができる他のノード）に挿入された場合は、ドキュメント・フラグメント自身ではなくドキュメント・フラグメントの子がノードに挿入されます。このため、ドキュメント・フラグメントは、兄弟ノードを作成する場合に非常に有効です。ドキュメント・フラグメントは、insertBefore() や appendChild() など、Node インタフェースの標準メソッドを使用できるように、これらのノードの親として機能します。

org.w3c.dom.DocumentType インタフェース

public interface **DocumentType** extends Node

各ドキュメントは、NULL またはドキュメント・タイプ・オブジェクトを値に持つ doctype 属性を持ちます。DOM レベル 1 Core の DocumentType インタフェースによって、ドキュメント用に定義されたエンティティのリストに対するインタフェースは提供されますが、その他にはほとんど効果がありません。これは、DTD を表すための名前空間の効果および様々な XML スキームがこの書込み時点では明確に理解されないためです。

DOM レベル 1 は、ドキュメント・タイプ・ノードの編集をサポートしていません。

抽象メソッド（一覧）

getEntities()

DTD で宣言されている外部および内部の汎用エンティティを含む NamedNodeMap を取得します。

getName()

DTD の名前（DOCTYPE キーワードの直後の名前）を取得します。

getNotations()

DTD で宣言されている表記法を含む NamedNodeMap を取得します。

抽象メソッド（詳細）

getEntities

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

getName

`public abstract String getName()`
DTD の名前（`DOCTYPE` キーワードのすぐ後の名前）を取得します。

getNotations

`public abstract  getNotations()`
DTD で宣言されている表記法を含む NamedNodeMap を取得します。複製は廃棄されます。このマップのすべてのノードは、Notation インタフェースも実装します。
DOM レベル 1 では、表記法の編集がサポートされないため、表記法は変更できません。

org.w3c.dom.DOMException クラス

```
java.lang.Object
   |
   +---java.lang.Throwable
           |
           +---java.lang.Exception
                   |
                   +---java.lang.RuntimeException
                           |
                           +---org.w3c.dom.DOMException
```

public abstract class **DOMException** extends RuntimeException

DOM 操作では、論理的な理由、データの消失、実装の不安定さによって操作を実行できないという例外的な状況でのみ例外が発生します。通常、DOM メソッドでは、ノード・リスト使用時の out-of-bound エラーなど、通常の処理状況での特定のエラー値が戻されます。

実装によっては、別の状況で別の例外が発生する場合があります。たとえば、NULL 引数が渡された場合に、実装依存の例外が発生する場合があります。

一部の言語およびオブジェクト・システムには、例外の概念をサポートしないものもあります。このようなシステムでは、システム固有のエラー・レポート・メカニズムを使用してエラーの状況が示されます。たとえば、特定のバインドでは、メソッドによって、対応するメソッドの説明で表示されるエラー・コードと同様のコードが戻される場合があります。

INDEX_SIZE_ERR

public static final short INDEX_SIZE_ERR

DOMSTRING_SIZE_ERR

public static final short DOMSTRING_SIZE_ERR

HIERARCHY_REQUEST_ERR

public static final short HIERARCHY_REQUEST_ERR

WRONG_DOCUMENT_ERR

public static final short WRONG_DOCUMENT_ERR

INVALID_CHARACTER_ERR

public static final short INVALID_CHARACTER_ERR

NO_DATA_ALLOWED_ERR

public static final short NO_DATA_ALLOWED_ERR

NO_MODIFICATION_ALLOWED_ERR

public static final short NO_MODIFICATION_ALLOWED_ERR

NOT_FOUND_ERR

public static final short NOT_FOUND_ERR

NOT_SUPPORTED_ERR

public static final short NOT_SUPPORTED_ERR

INUSE_ATTRIBUTE_ERR

public static final short INUSE_ATTRIBUTE_ERR

DOMException

```
public DOMException(short code,
                    String message)
```

org.w3c.dom.DOMImplementation インタフェース

public interface **DOMImplementation**

DOMImplementation インタフェースによって、DOM の特定のインスタンスに依存しない操作を実行するために、多数のメソッドが提供されます。

DOM レベル 1 では、ドキュメント・インスタンスの作成方法は指定されていません。したがって、ドキュメントの作成は、実装固有の操作です。DOM 仕様の将来のレベルでは、ドキュメントを直接作成するためのメソッドが提供される予定です。

抽象メソッド（一覧）

hasFeature(String, String)

DOM インプリメンテーションが固有の機能を実装するかどうかをテストします。

抽象メソッド（詳細）

hasFeature

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

DOM インプリメンテーションが固有の機能を実装するかどうかをテストします。

パラメータ

`feature` - 確認する機能のパッケージ名。レベル 1 の場合、正当な値は「HTML」および「XML」です（大文字 / 小文字は区別されません）。

`version` - テストするパッケージ名のバージョン番号です。レベル 1 の場合は文字列「1.0」です。バージョンを指定していない場合、その機能のすべてのバージョンがサポートされ、TRUE が戻されます。

戻り値

機能が、指定したバージョンに実装されている場合は TRUE、それ以外の場合は FALSE を戻します。

org.w3c.dom.Element インタフェース

public interface **Element** extends Node

ドキュメントで使用されるオブジェクト（テキスト以外）の大部分は、要素ノードです。<elementExample id="demo"> <subelement1/> <subelement2><subsubelement/></subelement2> </elementExample> という XML ドキュメントがあるとします。

DOM を使用して表すと、最上位ノードは elementExample の要素ノードです。このノードには、2 つの子である要素ノードがあり、1 つは subelement1、もう 1 つは subelement2 です。subelement1 は、子ノードを含みません。

要素は、それに対応する属性を持ちます。Element インタフェースは Node を継承するため、汎用 Node インタフェース・メソッド getAttributes を使用して、要素のすべての属性セットを取得できます。Element インタフェースには、名前で属性オブジェクトまたは属性値を取得するためのメソッドがあります。XML では、属性値にエンティティ参照が含まれる場合に、属性オブジェクトを取り出して、属性値を表す複雑なサブツリーを調べる必要があります。これに対し、HTML では、すべての属性は単純な文字列を値に持つため、属性値に直接アクセスするメソッドを使用しても、問題なく効果的に処理できます。

抽象メソッド（一覧）

getAttribute(String)

名前で属性値を取得します。

getAttributeNode(String)

名前で属性ノードを取得します。

getElementsByTagName(String)

与えられたタグ名を持つすべての子要素のノード・リストを、要素ツリーの前順走査で一致する順に戻します。

getTagName()

要素名を取得します。

normalize()

要素の下位サブツリー全体にあるすべてのテキスト・ノードを正規形に変換します。この正規形では、マークアップ（タグ、コメント、処理命令、CDATA セクション、エンティティ参照など）のみがテキスト・ノードを分割し、隣接するテキスト・ノードがない状態になります。

removeAttribute(String)

指定する属性名の属性を削除します。

removeAttributeNode(Attr)

指定する属性ノードの属性を削除します。

setAttribute(String, String)

新しい属性を追加します。

setAttributeNode(Attr)

新しい属性を追加します。

抽象メソッド（詳細）

getAttribute

```
public abstract String getAttribute(String name)
```
名前で属性値を取得します。

パラメータ

name - 取得する属性の名前

戻り値

文字列としての属性値を戻します。その属性が指定した値またはデフォルト値を持たない場合は空の文字列を戻します。

getAttributeNode

```
public abstract Attr getAttributeNode(String name)
```
名前で属性ノードを取得します。

パラメータ

name - 取得する属性の名前

戻り値

指定した属性名を持つ属性ノード、または該当する属性が存在しない場合は NULL

getElementsByTagName

```
public abstract  getElementsByTagName(String name)
```
与えられたタグ名を持つすべての子要素のノード・リストを、要素ツリーの前順走査で一致する順に戻します。

パラメータ

name - 一致するタグ名。特殊値「*」を指定すると、すべてのタグに一致します。

戻り値

一致する要素ノードのリスト

getTagName

`public abstract String getTagName()`
要素名を取得します。たとえば、<elementExample id="demo"> ... </elementExample> の場合、要素名は値「elementExample」を持ちます。XML では、DOM のすべての操作と同様、大文字 / 小文字が区別されることに注意してください。HTML の DOM は、ソースである HTML ドキュメントでの大文字 / 小文字にかかわらず、HTML 要素の要素名を大文字の正規形で戻します。

normalize

`public abstract void normalize()`
要素の下位サブツリーの全体にあるすべてのテキスト・ノードを正規形に変換します。この正規形では、マークアップ（タグ、コメント、処理命令、CDATA セクション、エンティティ参照など）のみがテキスト・ノードを分割し、隣接するテキスト・ノードがない状態になります。これを使用して、ドキュメントの DOM 表示が、保存および再ロードされた場合と同様であることを確認できます。また、特定のドキュメント・ツリー構造に依存する操作（X ポインタ検索など）を使用する場合に有効です。

removeAttribute

`public abstract void removeAttribute(String name) throws DOMException`
指定する属性名の属性を削除します。削除した属性がデフォルト値を持つ場合は、すぐに置換されます。

パラメータ

`name` - 削除する属性の名前

例外の発生

DOMException

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

removeAttributeNode

`public abstract Attr removeAttributeNode(Attr oldAttr) throws DOMException`
指定する属性ノードの属性を削除します。

パラメータ

`oldAttr` - 属性リストから削除する属性ノード。削除した属性がデフォルト値を持つ場合は、すぐに置換されます。

戻り値

削除した属性ノード

例外の発生

DOMException

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

setAttribute

```
public abstract void setAttribute(String name,
                                  String value) throws DOMException
```

新しい属性を追加します。指定する属性名を持つ属性がすでに要素に存在する場合は、その値が、該当する値パラメータの値に変更されます。この値は、単純な文字列であり、設定されている場合は解析されません。このため、すべてのマークアップ（エンティティ参照として認識される構文など）はリテラル・テキストとして処理され、作成時に実装によって適切にエスケープされる必要があります。エンティティ参照を含む属性値を割り当てるには、1 つの属性ノードに加えて、任意のテキスト・ノードおよびエンティティ・リファレンス・ノードを作成し、適切なサブツリーを構築して、setAttributeNode によって、属性値として割り当てる必要があります。

パラメータ

`name` - 作成または変更する属性の名前

`value` - 文字列形式で設定する値

例外の発生

DOMException

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

setAttributeNode

```
public abstract  Attr setAttributeNode(Attr newAttr) throws
```

新しい属性を追加します。指定する属性名を持つ属性がすでに要素に存在する場合は、新しい属性によって置換されます。

パラメータ

`newAttr` - 属性リストに追加する属性ノード

戻り値

同じ属性名を持つ既存の属性が newAttr 属性によって置換されると、前に存在した属性ノードが戻されます。それ以外の場合は NULL が戻されます。

例外の発生

DOMException

WRONG_DOCUMENT_ERR: newAttr が、要素を作成したドキュメントとは異なるドキュメントから作成された場合に発生します。
NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
INUSE_ATTRIBUTE_ERR: newAttr が、すでに他の要素オブジェクトの属性である場合に発生します。DOM ユーザーは、属性ノードを明示的にクローニングして、それを他の要素で再度使用する必要があります。

org.w3c.dom.Entity インタフェース

public interface **Entity** extends Node

このインタフェースは、XML ドキュメントの解析対象または解析対象外エンティティを表します。エンティティ宣言ではなく、エンティティ自体をモデル化します。エンティティ宣言のモデル化は、DOM 仕様将来のレベルで対応される予定です。

ノードから継承された nodeName 属性には、エンティティ名が含まれます。

XML プロセッサは、構造モデルが DOM に渡される前に、エンティティを完全に拡張する場合があります。この場合、ドキュメント・ツリーにはエンティティ・リファレンス・ノードが存在しません。

XML では、非検証 XML プロセッサに対し、外部サブセットで作成されたエンティティ宣言、または外部パラメータ・エンティティで宣言されたエンティティ宣言の読込みおよび処理命令は行いません。これは、外部サブセットで宣言された解析対象エンティティを、アプリケーションのいくつかのクラスで拡張する必要があること、およびエンティティの置換値を使用できない場合があることを表します。置換値が使用可能である場合、対応するエンティティ・ノードの子リストに、その置換テキストの構造が示されます。使用不可の場合、子リストは空になります。

エンティティの子の変換（置換値）は、遅延評価される場合があります。この場合、ユーザーによる操作（エンティティ・ノードでの childNodes メソッドのコールなど）によって、この遅延評価がトリガーされると想定します。

DOM レベル 1 では、エンティティ・ノードの編集はサポートされません。エンティティの内容を変更する場合は、構造モデルで、すべての関連エンティティ・リファレンス・ノードをエンティティの内容のクローンに置き換え、かわりに、これらのクローンに必要な変更を加える必要があります。エンティティ・ノードのすべての子は、読取り専用です。

エンティティ・ノードは親を持ちません。

抽象メソッド（一覧）

getNotationName()

解析対象外エンティティの場合、そのエンティティの表記法の名前を取得します。

getPublicId()

エンティティに対応する公開識別子（指定されている場合）を取得します。

getSystemId()

エンティティに対応するシステム識別子（指定されている場合）を取得します。

抽象メソッド（詳細）

getNotationName

```
public abstract String getNotationName()
```

解析対象外エンティティの場合、そのエンティティの表記法の名前を取得します。解析対象エンティティの場合は、NULL になります。

getPublicId

```
public abstract String getPublicId()
```

エンティティに対応する公開識別子（指定されている場合）を取得します。公開識別子が指定されていない場合、NULL になります。

getSystemId

```
public abstract String getSystemId()
```

エンティティに対応するシステム識別子（指定されている場合）を取得します。システム識別子が指定されていない場合、NULL になります。

org.w3c.dom.EntityReference インタフェース

public interface **EntityReference** extends Node

エンティティ参照がソース・ドキュメント内にある場合、またはユーザーがエンティティ参照を挿入する場合、エンティティ・リファレンス・オブジェクトを構造モデルに挿入できます。文字参照および事前定義済エンティティへの参照は、文字がエンティティ参照ではなく、それに相当する Unicode で表されるように、HTML または XML プロセッサによって拡張されることに注意してください。また、XML プロセッサは、エンティティ・リファレンス・オブジェクトの挿入時ではなく、構造モデルの構築中に、エンティティへの参照を完全に拡張できます。XML プロセッサがこのオブジェクトを挿入すると、与えられたエンティティ・リファレンス・ノードに対し、参照エンティティを表すエンティティ・ノードが存在しない場合があります。ただし、参照エンティティを表すエンティティが存在する場合、エンティティ・リファレンス・ノードの子リストは、エンティティ・ノードの子リストと同じになります。エンティティ・ノードの場合と同様、エンティティ・リファレンスのすべての子は読取り専用です。

エンティティ・リファレンスの子の変換（参照エンティティの置換値）は、遅延評価される場合があります。この場合、ユーザーによる操作（エンティティ・リファレンス・ノードでの childNodes メソッドのコールなど）によってこの遅延評価がトリガーされると想定します。

org.w3c.dom.NamedNodeMap インタフェース

public interface **NamedNodeMap**

NamedNodeMap インタフェースを実装するオブジェクトは、名前によってアクセス可能なノードのコレクションを表すために使用されます。NamedNodeMap はノード・リストを継承せず、NamedNodeMap の順序は維持されないことに注意してください。
NamedNodeMap を実装するオブジェクトに含まれるオブジェクトにも順序索引によってアクセス可能ですが、これは単に NamedNodeMap の内容を列挙する場合に有効であり、DOM がこれらのノードの順序を指定するわけではありません。

抽象メソッド（一覧）

getLength() getLength()

マップ内のノード数を取得します。

getNamedItem(String)

名前で指定したノードを取得します。

item(int)

マップ内の index 番目の項目を戻します。

removeNamedItem(String)

名前で指定したノードを削除します。

setNamedItem(Node)

ノードを、その nodeName 属性を使用して追加します。

抽象メソッド（詳細）

getLength

`public abstract int getLength()`
マップ内のノード数を取得します。有効な子ノード索引の範囲は、0 ～ length-1（両方の数値を含む）です。

getNamedItem

`public abstract  getNamedItem(String name)`
名前で指定したノードを取得します。

パラメータ

`name` - 取得するノードの名前

戻り値

指定した名前を持つ任意の型のノード、指定した名前がマップ内のノードを識別しなかった場合は NULL

item

`public abstract Node item(int index)`
マップ内の index 番目の項目を戻します。index の値がマップ内のノードの数以上である場合は、NULL を戻します。

パラメータ

`index` - マップ内の索引

戻り値

NamedNodeMap 内で index 番目の位置にあるノード、またはそれが有効な索引でない場合は NULL

removeNamedItem

```
public abstract Node removeNamedItem(String name) throws DOMException
```

名前で指定したノードを削除します。削除したノードがデフォルト値を持つ属性である場合は、すぐに置換されます。

パラメータ

`name` - 削除するノードの名前

戻り値

マップから削除されたノードを戻します。該当する名前を持つノードが存在しない場合は NULL を戻します。

例外の発生

DOMException

NOT_FOUND_ERR: マップ内に name という名前のノードが存在しない場合に発生します。

setNamedItem

```
public abstract Node setNamedItem(Node arg) throws DOMException
```

ノードを、その nodeName 属性を使用して追加します。
nodeName 属性を使用して、格納するノードの名前を導出する場合、名前が重複する特定の型（値が特殊文字列の型）を持つ複数のノードを格納することはできません。ノードに別名を付けるより、ノードを追加することをお薦めします。

パラメータ

`arg` - 名前付きノード・マップに格納するノード。格納したノードには、ノードの nodeName 属性の値を使用してアクセスできます。その名前を持つノードがすでにマップに存在する場合は、新しいノードによって置換されます。

戻り値

同じ値を持つ既存のノードが新しいノードによって置換された場合は、前に存在したノードを戻します。それ以外の場合は NULL を戻します。

例外の発生

DOMException

WRONG_DOCUMENT_ERR: arg が、NamedNodeMap を作成したドキュメントとは異なるドキュメントから作成された場合に発生します。
NO_MODIFICATION_ALLOWED_ERR: この NamedNodeMap が読取り専用である場合に発生します。
INUSE_ATTRIBUTE_ERR: arg が、すでに他の要素オブジェクトの属性である属性である場合に発生します。DOM ユーザーは、属性ノードを明示的にクローニングして、それを他の要素で再度使用する必要があります。

org.w3c.dom.Node インタフェース

public interface **Node**

Node インタフェースは、DOM 全体のプライマリ・データ型です。このインタフェースは、ドキュメント・ツリーの単一ノードを表します。Node インタフェースを実装するすべてのオブジェクトは、子を処理するためのメソッドを公開しますが、Node インタフェースを実装するすべてのオブジェクトが子を持つことができるわけではありません。たとえば、テキスト・ノードは子を持つことができず、このようなノードに子を追加すると、DOMException が発生します。

特定の導出インタフェースにキャストしなくてもノード情報を取得できるメカニズムとして nodeName、nodeValue および attributes 属性が含まれます。特定の nodeType の属性（要素の nodeValue、コメントの attributes など）が明確にマッピングされていない場合は、NULL が戻されます。特殊インタフェースには、関連情報を取得および設定するためにより有効なメカニズムが他にも含まれる場合があることに注意してください。

抽象メソッド（一覧）

appendChild(Node)

このノードの子リストの最後に newChild ノードを追加します。

cloneNode(boolean)

このノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。

getAttributes()

このノードが要素の場合は、その属性を含む NamedNodeMap を、それ以外の場合は NULL を取得します。

getChildNodes()

このノードのすべての子ノードを含むノード・リストを取得します。

getFirstChild()

このノードの最初の子を取得します。

getLastChild()

このノードの最後の子を取得します。

getNextSibling()

このノードの直後のノードを取得します。

getNodeName()

このノードの名前を、ノードの型に応じて取得します。前述の表を参照してください。

getNodeType()

基礎となるオブジェクトの型を表すコードを、前述の定義に従って取得します。

getNodeValue()

このノードの値を、ノードの型に応じて取得します。前述の表を参照してください。

getOwnerDocument()

このノードに対応するドキュメント・オブジェクトを取得します。

getParentNode()

このノードの親を取得します。

getPreviousSibling()

このノードの直前のノードを取得します。

hasChildNodes()

ノードに子ノードがあるかどうかを判別します。

insertBefore(Node, Node)

newChild ノードを既存の子ノード refChild の前に挿入します。

removeChild(Node)

oldChild が示す子ノードを子リストから削除し、それを戻します。

replaceChild(Node, Node)

子リスト内の子ノード oldChild を newChild で置換し、oldChild ノードを戻します。

setNodeValue(String)

ELEMENT_NODE

```
public static final short ELEMENT_NODE
```

ATTRIBUTE_NODE

```
public static final short ATTRIBUTE_NODE
```

TEXT_NODE

```
public static final short TEXT_NODE
```

CDATA_SECTION_NODE

```
public static final short CDATA_SECTION_NODE
```

ENTITY_REFERENCE_NODE

```
public static final short ENTITY_REFERENCE_NODE
```

ENTITY_NODE

```
public static final short ENTITY_NODE
```

PROCESSING_INSTRUCTION_NODE

```
public static final short PROCESSING_INSTRUCTION_NODE
```

COMMENT_NODE

```
public static final short COMMENT_NODE
```

DOCUMENT_NODE

```
public static final short DOCUMENT_NODE
```

DOCUMENT_TYPE_NODE

```
public static final short DOCUMENT_TYPE_NODE
```

DOCUMENT_FRAGMENT_NODE

```
public static final short DOCUMENT_FRAGMENT_NODE
```

NOTATION_NODE

```
public static final short NOTATION_NODE
```

抽象メソッド（詳細）

appendChild

```
public abstract  appendChild( newChild) throws DOMException
```

このノードの子リストの最後に newChild ノードを追加します。newChild がツリー内にすでに存在する場合は、最初に削除されます。

パラメータ

newChild - 追加するノード。これがドキュメント・フラグメント・オブジェクトである場合は、ドキュメント・フラグメントの内容全体が、このノードの子リストに移動されます。

戻り値

追加されたノード

例外の発生

DOMException

HIERARCHY_REQUEST_ERR: このノードが newChild ノード型の子を許可しない型である場合、または追加するノードがこのノードの祖先クラスの 1 つである場合に発生します。
WRONG_DOCUMENT_ERR: newChild が、このノードを作成したドキュメントとは異なるドキュメントから作成された場合に発生します。
NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。

cloneNode

```
public abstract Node cloneNode(boolean deep)
```

このノードの複製を戻し、ノードの汎用コピー・コンストラクタとして機能します。複製ノードは親を持ちません（parentNode は NULL を戻します）。
要素をクローニングすると、XML プロセッサによって生成されたデフォルト値の属性を含む、すべての属性およびその値がコピーされます。ただし、このメソッドでは、ノードに含まれるテキストは、子であるテキスト・ノードに含まれているため、ディープ・クローンの場合以外はコピーされません。他の型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、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ます。FALSE の場合、ノードのみが（ノードが要素の場合はその属性も）クローニングされます。

戻り値

複製ノード

getAttributes

```
public abstract NamedNodeMap getAttributes()
```

このノードが要素の場合はその属性を含む NamedNodeMap を、それ以外の場合は NULL を取得します。

getChildNodes

```
public abstract  NodeList getChildNodes()
```

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

getFirstChild

```
public abstract Node getFirstChild()
```

このノードの最初の子を取得します。該当するノードがない場合は、NULL を戻します。

getLastChild

```
public abstract Node getLastChild()
```

このノードの最後の子を取得します。該当するノードがない場合は、NULL を戻します。

getNextSibling

```
public abstract Node getNextSibling()
```

このノードの直後のノードを取得します。該当するノードがない場合は、NULL を戻します。

getNodeName

```
public abstract String getNodeName()
```

このノードの名前を、ノードの型に応じて取得します。

getNodeType

```
public abstract short getNodeType()
```

基礎となるオブジェクトの型を表すコードを取得します。

getNodeValue

```
public abstract String getNodeValue() throws
```

このノードの値を、ノードの型に応じて取得します。

例外の発生

DOMException

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

例外の発生

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

getOwnerDocument

```
public abstract Document getOwnerDocument()
```

このノードに対応するドキュメント・オブジェクトを取得します。これは、新しいノードを作成するために使用するドキュメント・オブジェクトでもあります。このノードがドキュメントである場合は、NULL を戻します。

getParentNode

```
public abstract Node getParentNode()
```

このノードの親を取得します。ドキュメント、ドキュメント・フラグメントおよび属性以外のすべてのノードは、親を持ちます。ただし、ノードが作成された直後でツリーに追加されない場合、またはツリーから削除された場合は、NULL になります。

getPreviousSibling

```
public abstract Node getPreviousSibling()
```

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

hasChildNodes

```
public abstract boolean hasChildNodes()
```
ノードに子ノードがあるかどうかを判別します。

戻り値

ノードが子を持つ場合は TRUE、子を持たない場合は FALSE が戻されます。

insertBefore

```
public abstract Node insertBefore(Node newChild,
                                  Node refChild) throws DOMException
```
newChild ノードを既存の子ノード refChild の前に挿入します。refChild が NULL の場合は、子リストの最後に newChild を挿入します。
newChild がドキュメント・フラグメント・オブジェクトである場合は、そのすべての子が同じ順序で refChild の前に挿入されます。newChild がツリー内にすでに存在する場合は、最初に削除されます。

パラメータ

`newChild` - 挿入するノード

`refChild` - 参照ノード（このノードの前に新しいノードが挿入されます）

戻り値

挿入中のノード

例外の発生

DOMException

HIERARCHY_REQUEST_ERR: このノードが newChild ノード型の子を許可しない型である場合、または挿入するノードがこのノードの祖先クラスの 1 つである場合に発生します。
WRONG_DOCUMENT_ERR: newChild が、このノードを作成したドキュメントとは異なるドキュメントから作成された場合に発生します。
NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
NOT_FOUND_ERR: refChild がこのノードの子でない場合に発生します。

removeChild

```
public abstract Node removeChild(Node oldChild) throws DOMException
```
oldChild が示す子ノードを子リストから削除し、それを戻します。

パラメータ

`oldChild` - 削除中のノード

戻り値

削除されたノード

例外の発生

DOMException

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

replaceChild

```
public abstract Node replaceChild(Node newChild,
                    Node oldChild) throws DOMException
```

子リスト内の子ノード oldChild を newChild と置換し、oldChild ノードを戻します。newChild がツリー内にすでに存在する場合は、最初に削除されます。

パラメータ

`newChild` - 子リストに挿入する新しいノード

`oldChild` - リスト内で置換されるノード

戻り値

置換されたノード

例外の発生

DOMException

HIERARCHY_REQUEST_ERR: このノードが newChild ノード型の子を許可しない型である場合、または挿入するノードがこのノードの祖先クラスの 1 つである場合に発生します。
WRONG_DOCUMENT_ERR: newChild が、このノードを作成したドキュメントとは異なるドキュメントから作成された場合に発生します。
NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。
NOT_FOUND_ERR: oldChild がこのノードの子でない場合に発生します。

setNodeValue

```
public abstract void setNodeValue(String nodeValue) throws DOMException
```

org.w3c.dom.NodeList インタフェース

public interface **NodeList**

ノード・リスト・インタフェースは、順序付けられたノード・コレクションを、このコレクションの実装方法を定義または制約せずに抽象化します。

ノード・リスト内の項目には、0 から始まる統合索引を介してアクセスできます。

抽象メソッド（一覧）

getLenth()

リスト内のノード数を取得します。

item(int)

コレクション内の index 番目の項目を戻します。

抽象メソッド（詳細）

getLength

`public abstract int getLength()`
リスト内のノード数を取得します。有効な子ノード索引の範囲は、0 ～ length-1（両方の数値を含む）です。

item

`public abstract Node item(int index)`
コレクション内の index 番目の項目を戻します。index がリスト内のノード数以上である場合は、NULL を戻します。

パラメータ

`index` - コレクションに対する索引

戻り値

ノード・リスト内で index 番目の位置にあるノード、またはそれが有効な索引でない場合は NULL を戻します。

org.w3c.dom.Notation インタフェース

public interface **Notation** extends Node

このインタフェースは、DTD で宣言されている記法を表します。記法は、解析対象外エンティティ形式を名前で宣言したり（XML1.0 仕様のセクション 4.7 を参照）、処理命令ターゲットの形式宣言に使用されます（同セクション 2.6 を参照）。Node から継承された nodeName 属性には、記法の宣言名が設定されます。

DOM レベル 1 では、記法ノードの編集がサポートされないため、このノードは読取り専用です。

記法ノードは親を持ちません。

抽象メソッド（一覧）

getPublicId()

この記法の公開識別子を取得します。

getSystemId()

この記法のシステム識別子を取得します。

抽象メソッド（詳細）

getPublicId

`public abstract String getPublicId()`
この記法の公開識別子を取得します。公開識別子が指定されていない場合は、NULL になります。

getSystemId

`public abstract String getSystemId()`
この記法のシステム識別子を取得します。システム識別子が指定されていない場合は、NULL になります。

org.w3c.dom.ProcessingInstruction インタフェース

public interface **ProcessingInstruction** extends Node

ProcessingInstruction インタフェースは、処理命令を表します。このインタフェースは、プロセッサ固有の情報をドキュメントのテキスト内に保持する手段として XML で使用されます。

抽象メソッド（一覧）

getData()

この処理命令の内容を取得します。

getTarget()

この処理命令のターゲットを取得します。

setData(String)

抽象メソッド（詳細）

getData

`public abstract String getData()`
この処理命令の内容を取得します。これは、ターゲットの後の最初の非空白文字から ?> の直前の文字までです。

例外の発生

DOMException

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

getTarget

`public abstract String getTarget()`
この処理命令のターゲットを取得します。XML は、これを、処理命令を開始するマークアップの後の最初のトークンとして定義します。

setData

`public abstract void setData(String data) throws`

org.w3c.dom.Text インタフェース

public interface **Text** extends Character Data

Text インタフェースは、要素または属性のテキスト内容（XML では「文字データ」という）を表します。要素の内容にマークアップがない場合、テキストは、その要素の唯一の子である、Text インタフェースを実装するシングル・オブジェクトに含まれます。マークアップがある場合、テキストは、要素の子リストを構成する要素およびテキスト・ノードのリストに解析されます。

ドキュメントが DOM によって使用可能になった時点では、テキストの各ブロックには 1 つのテキスト・ノードしかありません。ユーザーは、マークアップを使用しなくても与えられた要素の内容を表す隣接したテキスト・ノードを作成できます。ただし、XML または HTML では、これらのノードの区切りを表す方法がないため、通常、これらのノードが、DOM 編集セッション間で存続することはありません。要素の normalize() メソッドは、このような隣接するテキスト・オブジェクトを、テキストの各ブロック用の単一ノードにマージします。このマージは、特定のドキュメント構造に依存する操作（XPointer によるナビゲーションなど）を実行する前に行うことをお薦めします。

抽象メソッド（一覧）

splitText(int)

このテキスト・ノードを、指定したオフセットで 2 つのテキスト・ノードに分割し、その両方を兄弟としてツリー内に保持します。

抽象メソッド（詳細）

splitText

```
public abstract Text splitText(int offset) throws DOMException
```

テキスト・ノードを、指定したオフセットで 2 つのテキスト・ノードに分割し、その両方を兄弟としてツリー内に保持します。1 つ目のノードには、オフセット点までのすべての内容が含まれます。新しく挿入される 2 つ目のノードには、オフセット点およびそれから後のすべての内容が含まれます。

パラメータ

`offset` - 分割点となるオフセット（0 から開始）

戻り値

新しいテキスト・ノード

例外の発生

DOMException

INDEX_SIZE_ERR: 指定したオフセットが負数である場合、または data 内の文字数より多い場合に発生します。
NO_MODIFICATION_ALLOWED_ERR: このノードが読取り専用である場合に発生します。

第 III 部

XML-SQL Utility（XSU）パッケージ

第 III 部に含まれる章は、次のとおりです。

- 第 6 章「Oracle XSU Java API」
- 第 7 章「Oracle XSU PL/SQL API」

6

Oracle XSU Java API

この章では、Oracle XSU Java API について説明します。

- oracle.xml.sql.dml.OracleXMLSave
- oracle.xml.sql.query.OracleXMLQuery
- oracle.xml.sql.OracleXMLSQLException
- oracle.xml.sql.OracleXMLSQLNoRowsException

oracle.xml.sql.dml.OracleXMLSave

構文

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

java.lang.Object
  |
  +--oracle.xml.sql.dml.OracleXMLSave
```

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正規のマッピングをサポートします。挿入、更新および削除がサポートされます。ユーザーはまず、これらの DML 操作を行う必要がある表名を指定して、クラスを作成します。その後、この表に対して挿入 / 更新 / 削除ができるようになります。次に例を示します。

```
OracleXMLSave sav = new OracleXMLSave(conn, "emp");
// 挿入処理
sav.insertXML(xmlDoc); -or-
// 更新処理
String[]tempArr = new String[2];
tempArr[0] = "EMPNO";
// empno を更新のキーとして設定します。
sav.setUpdateKeyColumnList(tempArray);
sav.updateXML(xmlDoc); -or-
sav.deleteXML(xmlDoc);
sav.close();
```

このクラスでは、更新または削除のためのキー列を識別したり、更新される列を制限するために有効な多くの機能が提供されています。

メンバーの概要

フィールド

DATE_FORMAT	setDateFormat(String) で使用される日付書式です。
DEFAULT_BATCH_SIZE	挿入時のデフォルトのバッチ・サイズは 17 です。

コンストラクタ

OracleXMLSave(Connection, String)	OracleXMLSave ユーティリティのパブリック・コンストラクタです。

メンバーの概要

メソッド

close()	このオブジェクトに対応するすべてのコンテキストのクローズまたは割当て解除を行います。
deleteXML(Document)	XML ドキュメントのレコードに対応するデータベース・オブジェクトから、レコードを削除します。
deleteXML(InputStream)	XML ドキュメントのレコードに対応するデータベース・オブジェクトから、レコードを削除します。
deleteXML(URL)	XML ドキュメントのレコードに対応するデータベース・オブジェクトから、レコードを削除します。
insertXML(Document)	指定した表に XML ドキュメントを挿入します。処理された行の数を戻します。
insertXML(InputStream)	指定した表に XML ドキュメントを挿入します。処理された行の数を戻します。
insertXML(String)	指定した表に XML ドキュメントを挿入します。処理された行の数を戻します。
insertXML(URL)	指定した表に XML ドキュメントを挿入します。処理された行の数を戻します。
setBatchSize(int)	DML 操作中に使用されるバッチ・サイズを変更します。
setCommitBatch(int)	コミットのバッチ・サイズを設定します。
setDateFormat(String)	日付タグの書式を設定します。
setIgnoreCase(boolean)	XSU がタグ名と表 / ビューの列名を比較する場合に、タグ名の大 / 小文字を区別しないように指定します。
setKeyColumnList(String[])	更新または削除中に、データベース表内の特定の行を識別するために使用する、列のリストを設定します。
setRowTag(String)	XML ドキュメントで使用するタグを指定し、各行の値に対応する XML 要素を囲みます。
setUpdateColumnList(String[])	更新する列の値を設定します。
updateXML(Document)	XML ドキュメントを与えられた表を更新します。
updateXML(InputStream)	XML ドキュメントを与えられた表を更新します。
updateXML(String)	XML ドキュメントを与えられた表を更新します。
updateXML(URL)	XML ドキュメントを与えられた表を更新します。

継承されたメンバーの概要

java.lang.Object クラスから継承されたメソッド

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

フィールド

DATE_FORMAT

```
public static final java.lang.String DATE_FORMAT
```

DEFAULT_BATCH_SIZE

```
public static int DEFAULT_BATCH_SIZE
```

挿入時のデフォルトのバッチ・サイズは 17 です。

コンストラクタ

OracleXMLSave(Connection, String)

```
public  OracleXMLSave(java.sql.Connection oconn, java.lang.String tabName)
```

OracleXMLSave ユーティリティのパブリック・コンストラクタです。

メソッド

close()

```
public void close()
```

deleteXML(Document)

```
public int deleteXML(org.w3c.dom.Document doc)
```

XML ドキュメントのレコードに対応するデータベース・オブジェクトから、レコードを削除します。

deleteXML(InputStream)

```
public int deleteXML(java.io.InputStream xmlStream)
```

XML ドキュメントのレコードに対応するデータベース・オブジェクトから、レコードを削除します。

deleteXML(String)

```
public int deleteXML(java.lang.String xmlDoc)
```

XML ドキュメントのレコードに対応するデータベース・オブジェクトから、レコードを削除します。

deleteXML(URL)

```
public int deleteXML(java.net.URL url)
```

XML ドキュメントのレコードに対応するデータベース・オブジェクトから、レコードを削除します。

insertXML(Document)

```
public int insertXML(org.w3c.dom.Document doc)
```

指定した表に XML ドキュメントを挿入します。処理された行の数を戻します。

insertXML(InputStream)

```
public int insertXML(java.io.InputStream xmlStream)
```

指定した表に XML ドキュメントを挿入します。処理された行の数を戻します。

insertXML(String)

```
public int insertXML(java.lang.String xmlDoc)
```

指定した表に XML ドキュメントを挿入します。処理された行の数を戻します。

insertXML(URL)

```
public int insertXML(java.net.URL url)
```

指定した表に XML ドキュメントを挿入します。処理された行の数を戻します。

setBatchSize(int)

```
public void setBatchSize(int size)
```

DML 操作中に使用されるバッチ・サイズを変更します。挿入、更新または削除を実行する場合、別々の文ではなく、1 回の処理でそれらの操作が実行されるように、バッチ処理を行う方が適しています。欠点は、その操作を実行する前に、すべてのバインド値を保持するための処理メモリーがさらに多く必要になることです。バッチ処理を行う場合、バッチ処理中のみコミットが発生することに注意してください。そのため、バッチ内の 1 文が正常に実行されなかった場合、バッチ処理全体がロールバックされます。これを防止するには、バッチ・サイズを 1 に設定します。デフォルトのバッチ・サイズは、DEFAULT_BATCH_SIZE です。

setCommitBatch(int)

```
public void setCommitBatch(int size)
```

コミットのバッチ・サイズを設定します。コミットのバッチ・サイズとは、実行後にコミットする必要がある挿入の回数または記録を意味します。commitBatch が 1 未満またはセッションが自動コミット・モードの場合、XSU は明示的なコミットを行わないことに注意してください。コミットのデフォルトのバッチ・サイズは 0（ゼロ）です。

setDateFormat(String)

```
public void setDateFormat(java.lang.String dateformat)
```

日付タグの書式を設定します。デフォルトでは、OracleXMLSave は、日付書式を 'MM/dd/yyyy HH:mm:ss' であると想定しています。これは、OracleXMLQuery によって出力される書式です。ここに書式を設定して、日付書式を上書きできます。書式は、JDBC が認識できる有効な書式である必要があります。有効な日付書式のリストについては、java.text.SimpleDateFormat にある JDBC のドキュメントを参照してください。ここで行う変更はグローバルです。NULL 値または空の文字列を指定すると、デフォルトの書式にリセットされます。

setIgnoreCase(boolean)

```
public void setIgnoreCase(boolean ignore)
```

タグ名と表 / ビューの列名を比較する場合に、タグ名の大 / 小文字を区別しないように指定します。デフォルトでは、タグ名の大 / 小文字が、対応する列名と同じである必要があります。setIgnoreCase(true) をコールすると、大 / 小文字を区別しないように変更されます。insertXML を最初にコールした後は、多くのメタデータがキャッシュされることに注意してください。setIgnoreCase の状態が変更された場合、すべてのメタデータを再取得する必要があります。

setKeyColumnList(String[])

```
public void setKeyColumnList(java.lang.String[] keyColNames)
```

更新または削除中に、データベース表内の特定の行を識別するために使用する、列のリストを設定します。このコールは、挿入の場合は無視されます。更新を実行する前に、キー列を設定する必要があります。IT は、削除用のオプションです。このキー列を設定すると、XML ドキュメント内のこれらのタグの値は、更新または削除するデータベース行を識別するために使用されます。

setRowTag(String)

```
public void setRowTag(java.lang.String rowTag)
```

各行の値を区別するために使用する、ROW タグの名前を設定します。この値を NULL に設定すると、ROW タグが存在せず、ドキュメントの最上位の要素が行自体に対応することを表します。

setUpdateColumnList(String[])

```
public void setUpdateColumnList(java.lang.String[] updColNames)
```

更新する列の値を設定します。これは、挿入および更新でのみ有効です。削除では無視されます。挿入の場合、デフォルトでは、表内のすべての列に値が挿入されます。更新の場合、デフォルトでは、XML ドキュメントの ROW 要素にあるタグに対応する列のみが更新されます。列を指定すると指定した列のみが、更新または挿入文で更新されます。他のすべてのタグは無視されます。

updateXML(Document)

```
public int updateXML(org.w3c.dom.Document doc)
```

この updateXML は、ドキュメントへのストリームを取ります。

updateXML(InputStream)

```
public int updateXML(java.io.InputStream xmlStream)
```

この updateXML は、ドキュメントへのストリームを取ります。

updateXML(String)

```
public int updateXML(java.lang.String xmlDoc)
```

この updateXML は、String のフォームにその XML ドキュメント自体を取ります。更新された行の数を戻します。

updateXML(URL)

```
public int updateXML(java.net.URL url)
```

データベース接続オブジェクトおよび更新する列名のリストを指定して、指定した URL から指定した表に XML ドキュメントを更新します。列名には、NULL を指定できます。UpdKeyNames は、where 句がチェックされるすべての列名をリストします。UpdColNames は、更新する列名をリストします。たとえば、次の従業員表 emp を考えます。

TABLE emp(empno NUMBER, ename VARCHAR2(20), hiredate DATE)

XML ドキュメントを使用して表を更新し、タグ値 EMPNO を使用して更新する行を取得して、雇用開始日の値のみを更新します。HIREDATE 値および EMPNO 値を含む XML ドキュメントに対し、EMPNO 文字列を含む updKeyColNames とともに updateXML ルーチンをコールできます。処理された XML の行要素数が戻されます。この数は、XML ドキュメントを介して選択された行が、表内の行を一意に識別するかどうかによって変更されたデータベースの行数と同等になる場合とならない場合があります。

oracle.xml.sql.query.OracleXMLQuery

構文

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

java.lang.Object
|
+--oracle.xml.sql.query.OracleXMLQuery
```

説明

OracleXMLQuery クラスは、SQL 問合せを指定して XML を生成します。単純な例を次に示します。

```
import java.sql.*;
import oracle.xml.sql.query.*;
import oracle.jdbc.driver.*;

public class sample
{
public static void main(String args[]) throws Exception
{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection conn =
DriverManager.getConnection("jdbc:oracle:oci8:scott/tiger@");
OracleXMLQuery qry =  new OracleXMLQuery(conn, "select * from emp");
System.out.println(qry.getXMLString());
conn.close();
}
}
```

メンバーの概要

フィールド

DTD	DTD の生成を指定します。
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	すべての行を結果に含めます。
NONE	DTD を生成しないことを指定します。
ROW_TAG	ROW 要素のデフォルトのタグ名を指定します。

メンバーの概要	
ROWIDATTR_TAG	ROW 要素のデフォルトのタグ名を指定します。
ROWSET_TAG	ドキュメントのデフォルトのタグ名を指定します。
SKIPROWS_ALL	結果内ですべての行をスキップします。
コンストラクタ	
OracleXMLQuery(Connection, ResultSet)	OracleXMLQueryObject のコンストラクタです。
OracleXMLQuery(Connection, String)	OracleXMLQueryObject のコンストラクタです。
OracleXMLQuery(OracleXMLDataSet)	OracleXMLQueryObject のコンストラクタです。
メソッド	
close()	すべてのオープン・リソースをクローズします。
getXMLDOM()	OracleXMLDocGenDOM クラスを使用して、XML ドキュメントの DOM 表現を戻し、DOM ツリーを生成します。
getXMLDOM(Node)	ルート・ノードを指定します。
getXMLDOM(Node, int)	XML ドキュメントを含むドキュメント・オブジェクトを戻します。
getXMLMetaData(int, boolean)	getXML コールによって生成された可能性がある XML ドキュメントの DTD を戻します。
getXMLString()	OracleXMLDocGenString クラスを使用して、XML ドキュメントの文字列表現を戻し、文字列を作成します。DTD は生成しません。
getXMLString(int)	OracleXMLDocGenString クラスを使用して、XML ドキュメントの文字列表現を戻し、文字列を作成します。
getXMLString(Node)	作成されたすべての子を、このルート・ノードに追加します。
getXMLString(Node, int)	XML ドキュメントを含む String オブジェクトを戻します。
keepObjectOpen(boolean)	ResultSet オブジェクトを取らないすべての getXML ファンクションのデフォルトの動作は、ResultSet オブジェクトおよび Statement オブジェクトをコールの終わりにクローズすることです。このメソッドを使用して、この動作を変更できます。これは、すべてのメタデータを保持するため、1 回に少数のレコードしか取得しない場合に有効です。
setCollIdAttrName(String)	
setDataHeader(Reader, String)	XML データ・ヘッダーを設定します。
setDateFormat(String)	XML ドキュメントに生成日付の書式を設定します。
setEncoding(String)	XML ドキュメントに処理命令のエンコーディングを設定します。

メンバーの概要	
setErrorTag(String)	エラーが発生した場合のエラー・タグ（デフォルトは ERROR）の名前を設定します。
setException(Exception)	ユーザーに例外の指定を許可し、XSU に対処させます。
setMaxRows(int)	skipRows の数の行がスキップされた後、問合せ結果から取り出す行の最大数を設定します。MAXROWS_NONE、MAXROWS_ALL または MAXROWS_DEFAULT のいずれかを使用できます。
setMetaHeader(Reader)	XML メタ・ヘッダーを設定します。
setRaiseException(boolean)	TRUE を指定してコールすると、NULL 行を戻すかわりに、「no data found」などの例外が発生します。
setRaiseNoRowsException(boolean)	生成された XML ドキュメントが空の場合に、OracleXMLNoRowsException が発生するかどうかを XSU に通知します。
setRowIdAttrName(String)	行の囲みタグの ID 属性名を設定します。
setRowIdAttrValue(String)	列値が行の囲みタグの ID 属性に割り当てられるスカラー列を指定します。
setRowsetTag(String)	XML データセットを囲むタグを設定します。
setRowTag(String)	データベースに対応する XML 要素を囲むタグを設定します。
setSkipRows(int)	スキップする行数を設定します。
setStylesheetHeader(String)	生成された XML ドキュメントにスタイルシート・ヘッダー（スタイルシート処理命令）を設定します。
setStylesheetHeader(String, String)	生成された XML ドキュメントにスタイルシート・ヘッダー（スタイルシート処理命令）を設定します。
setXSLT(Reader, String)	XML 生成中に適用するスタイルシートを設定します。
setXSLT(String, String)	XML 生成中に適用するスタイルシートを設定します。
useLowerCaseTagNames()	すべてのタグ名を小文字に設定します。
useNullAttributeIndicator(boolean)	NULL であることを示すために、XML 属性を使用するか、XML ドキュメント内の特定のエンティティを省略するかを指定します。
useUpperCaseTagNames()	すべてのタグ名を大文字に設定します。

継承されたメンバーの概要

java.lang.Object クラスから継承されたメソッド

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

フィールド

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
```
すべての行を結果に含めます。

NONE

```
public static final int NONE
```
DTD を生成しないことを指定します。

ROW_TAG

```
public static final 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。

SKIPROWS_ALL

```
public static final int SKIPROWS_ALL
```
結果内ですべての行をスキップします。これは、MAXROWS_NONE と同等です。

コンストラクタ

OracleXMLQuery(Connection, ResultSet)

```
public  OracleXMLQuery(java.sql.Connection conn, java.sql.ResultSet rset)
```

OracleXMLQuery(Connection, String)

```
public  OracleXMLQuery(java.sql.Connection conn, java.lang.String sqlString)
```

OracleXMLQuery(OracleXMLDataSet)

```
public  OracleXMLQuery(oracle.xml.sql.dataset.OracleXMLDataSet dset)
```

メソッド

close()

```
public void close()
```

OracleXML エンジンによって作成された、すべてのオープン・リソースをクローズします。ユーザーが指定したインスタンス結果セットはクローズしません。

getXMLDOM()

```
public oracle.xml.sql.query.Document getXMLDOM()
```

OracleXMLDocGenDOM クラスを使用して、XML ドキュメントの DOM 表現を戻し、DOM ツリーを生成します。

getXMLDOM(Node)

```
public oracle.xml.sql.query.Document getXMLDOM(oracle.xml.sql.query.Node root)
```

ルート・ノードを指定します。作成されたすべての子をこのルート・ノードに追加します。戻り値は、ドキュメント全体（ルート・ノードの所有者ドキュメント）を戻します。

getXMLDOM(Node, int)

```
public oracle.xml.sql.query.Document getXMLDOM(oracle.xml.sql.query.Node root, int
metaType)
```

XML ドキュメントを含むドキュメント・オブジェクトを戻します。このメソッドは、ノード・オブジェクトを取ります。ノード・オブジェクトは、NULL でない場合、XML ドキュメントの DOM ツリーのルート・ノードとして扱われます。もう 1 つのパラメータは、生成する XML メタデータがある場合、その型を指定するために使用されます。

getXMLMetaData(int, boolean)

```
public java.lang.String getXMLMetaData(int metaType, boolean withVersion)
```

getXML コールによって生成された可能性がある XML ドキュメントの DTD を戻します。metaType パラメータは、生成する XML メタデータの型を指定します。有効な値は DTD です。withVersion パラメータは、バージョン・ヘッダーを生成するかどうかを指定します。

getXMLMetaData(int, boolean, OracleXMLDocGen)

```
public void getXMLMetaData(int metaType, boolean withVersion,
oracle.xml.sql.docgen.OracleXMLDocGen doc)
```

getXMLString()

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

OracleXMLDocGenString クラスを使用して、XML ドキュメントの文字列表現を戻し、文字列を作成します。DTD は生成しません。

getXMLString(int)

```
public java.lang.String getXMLString(int metaType)
```

OracleXMLDocGenString クラスを使用して、XML ドキュメントの文字列表現を戻し、文字列を作成します。

getXMLString(Node)

```
public java.lang.String getXMLString(oracle.xml.sql.query.Node root)
```

作成されたすべての子を、このルート・ノードに追加します。新しい ROWTAG タグがこのルート・ノードに作成され、子が追加されます。

getXMLString(Node, int)

```
public java.lang.String getXMLString(oracle.xml.sql.query.Node root, int metaType)
```

XML ドキュメントを含む String オブジェクトを戻します。このメソッドは、ノード・オブジェクトを取ります。ノード・オブジェクトは、NULL でない場合、XML ドキュメントの DOM ツリーのルート・ノードとして扱われます。もう 1 つのパラメータは、生成する XML メタデータがある場合、その型を指定します。有効な値は、OracleXMLQuery.NONE および OracleXMLQuery.DTD です。ルートが NULL でない場合、DTD の作成を要求しても生成されないことに注意してください。

keepObjectOpen(boolean)

```
public void keepObjectOpen(boolean alive)
```

ResultSet オブジェクトを取らないすべての getXML ファンクションのデフォルトの動作は、ResultSet オブジェクトおよび Statement オブジェクトをコールの終わりにクローズすることです。getXML を繰り返しコールして次の行セットを取得する永続機能を使用する必要がある場合、このファンクションを TRUE に設定してコールすることで、このデフォルトの動作を停止する必要があります。OracleXMLQuery は、getXML のコール後、ResultSet および Statement オブジェクトをクローズしません。close() ファンクションをコールして、明示的にカーソル状態をクローズできます。

setCollIdAttrName(String)

```
public void setCollIdAttrName(java.lang.String collIdAttr)
```

コレクション要素の ID 属性名を設定します。NULL または空の文字列を指定すると、デフォルトの動作である collIDAttr の出力が制限されます。

setDataHeader(Reader, String)

```
public void setDataHeader(java.io.Reader header, java.lang.String docTag)
```

XML データ・ヘッダーを設定します。ヘッダーが設定されると、ヘッダーはこのオブジェクトが生成した各 XML ドキュメントのデータ部の先頭に挿入されます。docTag パラメータは、ヘッダーおよび生成された XML を追加した XML を囲むために使用するタグ名を指定します。ヘッダー・パラメータに NULL を指定してこのファンクションをコールすると、ヘッダーの設定が解除されます。

setDateFormat(String)

```
public void setDateFormat(java.lang.String mask)
```

XML ドキュメントに生成日付の書式を設定します。日付書式パターン（日付マスク）の構文は、java.text.SimpleDateFormat クラスの要件を満たす必要があります。マスクに NULL または空の文字列を設定すると、日付マスクの設定が解除されます。

パラメータ

`mask` - 日付マスク

setEncoding(String)

```
public void setEncoding(java.lang.String enc)
```

XML ドキュメントに処理命令のエンコーディングを設定します。エンコーディングとして NULL または空の文字列が指定された場合、デフォルトのキャラクタ・セットが処理命令のエンコーディングに指定されます。

パラメータ

`enc` - XML ドキュメントのキャラクタ・セットのエンコーディング

setErrorTag(String)

`public void setErrorTag(java.lang.String errorName)`
エラーが発生した場合のエラー・タグ（デフォルトは ERROR）の名前を設定します。エラーが発生した場合、DTD が生成されないことに注意してください。

setException(Exception)

`public void setException(java.lang.Exception e)`
ユーザーに例外の指定を許可し、XSU に対処させます。

setMaxRows(int)

`public void setMaxRows(int maxRows)`
skipRows の数の行がスキップされた後、問合せ結果から取り出す行の最大数を設定します。MAXROWS_NONE、MAXROWS_ALL または MAXROWS_DEFAULT のいずれかを使用できます。

setMetaHeader(Reader)

`public void setMetaHeader(java.io.Reader header)`
XML メタ・ヘッダーを設定します。ヘッダーが設定されると、このオブジェクトが生成した各 XML ドキュメントのメタデータ部（DTD）の先頭にヘッダーが挿入されます。このファンクションのヘッダー・パラメータに NULL を指定すると、ヘッダーの設定が解除されます。

setRaiseException(boolean)

`public void setRaiseException(boolean exp)`
TRUE を指定してコールすると、NULL 行を戻すかわりに、「no data found」などの例外が発生します。例外を捕捉して対処する必要があります。

setRaiseNoRowsException(boolean)

`public void setRaiseNoRowsException(boolean exp)`
TRUE を設定すると、デフォルトが FALSE である場合に、「no data found」などの例外が発生します。行が見つからない場合、空のドキュメント・インスタンスが取得されます。これを設定すると、実際の ResultSet が Data を生成した場合でも、OracleXMLSQLException が発生することに注意してください。ただし、maxRows および skipRows パラメータによって、行は選択されません。

setRowIdAttrName(String)

`public void setRowIdAttrName(java.lang.String rowIdTag)`
行区切りに使用される属性 ID の名前を設定します。NULL 値を指定すると、ID の出力を停止できます。

setRowIdAttrValue(String)

```
public void setRowIdAttrValue(java.lang.String colName)
```

ROW タグの ID 属性を、指定した列名の ID 属性値と同等にします。列はスカラー列である必要があります。NULL 値を指定すると、デフォルトの動作である行カウントに列を戻します。

setRowsetTag(String)

```
public void setRowsetTag(java.lang.String rowsetName)
```

結果の各レコード / 行に対するマークアップとして使用するタグ名を設定します。デフォルトのタグ名は、ROW です。NULL または空の文字列は、各行がカプセル化されたマークアップにならない（行セパレータが使用されない）ことを表します。この場合、問合せによって生成される結果セットは、1 列幅の多数のレコード、または多数の列幅の 1 つのレコードを含むことができます。

setRowTag(String)

```
public void setRowTag(java.lang.String rowTagName)
```

結果の各レコード / 行に対するマークアップとして使用するタグ名を設定します。デフォルトのタグ名は、ROW です。NULL または空の文字列は、各行がカプセル化されたマークアップにならない（行セパレータが使用されない）ことを表します。この場合、問合せによって生成される結果セットは、1 列幅の多数のレコード、または多数の列幅の 1 つのレコードを含むことができます。

setSkipRows(int)

```
public void setSkipRows(int skipRows)
```

結果を出力する前に、スキップする行数を設定します。SKIPROWS_NONE、SKIPROWS_ALL または SKIPROWS_DEFAULT のいずれかを使用できます。

setStylesheetHeader(String)

```
public void setStylesheetHeader(java.lang.String styleSheetURI)
```

setStylesheetHeader(String, String)

```
public void setStylesheetHeader(java.lang.String styleSheetURI, java.lang.String
styleSheetType)
```

setXSLT(Reader, String)

```
public void setXSLT()
```

生成された XML に適用するスタイルシートを登録します。スタイルシートがすでに登録されている場合は、新しいスタイルシートに置き換えられます。スタイルシートの登録を解除するには、stylesheet 引数に NULL または空の文字列を指定します。

setXSLT(String, String)

```
public void setXSLT()
```

生成された XML に適用するスタイルシートを登録します。スタイルシートがすでに登録されている場合は、新しいスタイルシートに置き換えられます。スタイルシートの登録を解除するには、stylesheet 引数に NULL または空の文字列を指定します。

useLowerCaseTagNames()

```
public void useLowerCaseTagNames()
```

すべてのタグ名を小文字に設定します。すべての希望のタグが設定された後に、これをコールする必要があります。

useNullAttributeIndicator(boolean)

```
public void useNullAttributeIndicator(boolean useNullAttr)
```

NULL であることを示すために、XML 属性を使用するか、XML ドキュメント内の特定のエンティティを省略するかを指定します。

useUpperCaseTagNames()

```
public void useUpperCaseTagNames()
```

すべてのタグ名を大文字に設定します。すべての希望のタグが設定された後に、これをコールする必要があります。

oracle.xml.sql.OracleXMLSQLException

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
```

既知のダイレクト・サブクラス

OracleXMLSQLNoRowsException

実装済インタフェース

java.io.Serializable

メンバーの概要

コンストラクタ

OracleXMLSQLException(Exception)

OracleXMLSQLException(Exception, String)

OracleXMLSQLException(String)

OracleXMLSQLException(String, Exception)

OracleXMLSQLException(String, Exception, String)

OracleXMLSQLException(String, int)

OracleXMLSQLException(String, int, String)

OracleXMLSQLException(String, String)

メンバーの概要	
メソッド	
getErrorCode()	
getParentException()	元の例外がある場合、それを戻します。ない場合は、NULL を戻します。
getXMLErrorString()	XML エラー文字列を、与えられたエラー・タグ名で出力します。
getXMLSQLErrorString()	エラー・メッセージにも SQL パラメータを出力します。
setErrorTag(String)	XML エラー・レポートを生成するために、getXMLErrorString および getXMLSQLErrorString で使用されるエラー・タグ名を設定します。

継承されたメンバーの概要
java.lang.Throwable クラスから継承されたメソッド
fillInStackTrace、getLocalizedMessage、getMessage、printStackTrace、printStackTrace、printStackTrace、toString
java.lang.Object クラスから継承されたメソッド
equals、getClass、hashCode、notify、notifyAll、wait、wait、wait

コンストラクタ

OracleXMLSQLException(Exception)

```
public  OracleXMLSQLException(java.lang.Exception e)
```

OracleXMLSQLException(Exception, String)

```
public  OracleXMLSQLException(java.lang.Exception e, java.lang.String errorTagName)
```

OracleXMLSQLException(String)

```
public  OracleXMLSQLException(java.lang.String message)
```

OracleXMLSQLException(String, Exception)

```
public  OracleXMLSQLException(java.lang.String message, java.lang.Exception e)
```

OracleXMLSQLException(String, Exception, String)

```
public  OracleXMLSQLException(java.lang.String message, java.lang.Exception e,
java.lang.String errorTagName)
```

OracleXMLSQLException(String, int)

```
public  OracleXMLSQLException(java.lang.String message, int errorCode)
```

OracleXMLSQLException(String, int, String)

```
public  OracleXMLSQLException(java.lang.String message, int errorCode,
java.lang.String errorTagName)
```

OracleXMLSQLException(String, String)

```
public  OracleXMLSQLException(java.lang.String message, java.lang.String
errorTagName)
```

メソッド

getErrorCode()

```
public int getErrorCode()
```

getParentException()

```
public java.lang.Exception getParentException()
```

元の例外がある場合、それを戻します。ない場合は、NULL を戻します。

getXMLErrorString()

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

XML エラー文字列を、与えられたエラー・タグ名で出力します。

getXMLSQLErrorString()

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

エラー・メッセージにも SQL パラメータを出力します。

setErrorTag(String)

```
public void setErrorTag(java.lang.String tagName)
```

XML エラー・レポートを生成するために、getXMLErrorString および getXMLSQLErrorString で使用されるエラー・タグ名を設定します。

oracle.xml.sql.OracleXMLSQLNoRowsException

構文

```
public class OracleXMLSQLNoRowsException extends OracleXMLSQLException

java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--java.lang.RuntimeException
|
+--OracleXMLSQLException
|
+--oracle.xml.sql.OracleXMLSQLNoRowsException
```

実装済インタフェース

java.io.Serializable

メンバーの概要

コンストラクタ

OracleXMLSQLNoRowsException()

OracleXMLSQLNoRowsException(String)

継承されたメンバーの概要

OracleXMLSQLException インタフェースから継承されたメソッド

getErrorCode()、getParentException()、getXMLErrorString()、getXMLSQLErrorString()、setErrorTag(String)

java.lang.Throwable クラスから継承されたメソッド

fillInStackTrace、getLocalizedMessage、getMessage、printStackTrace、printStackTrace、printStackTrace、toString

java.lang.Object クラスから継承されたメソッド

equals、getClass、hashCode、notify、notifyAll、wait、wait、wait

コンストラクタ

OracleXMLSQLNoRowsException()

```
public  OracleXMLSQLNoRowsException()
```

OracleXMLSQLNoRowsException(String)

```
public  OracleXMLSQLNoRowsException(java.lang.String errorTag)
```

7

Oracle XSU PL/SQL API

Oracle XSU PL/SQL API は、次の 2 つのパッケージで構成されます。

- DBMS_XMLQuery: DB_to_XML 型の機能を提供します。
- DBMS_XMLSave: XML_to_DB 型の機能を提供します。

DBMS_XMLQuery

型

ctxType

問合せコンテキスト・ハンドルの型です。DBMS_ XMLQuery.newContext() の戻り型です。

定数

DEFAULT_ROWSETTAG

結果セットから生成される XML を囲む要素のタグ名（ほとんどの場合、ルート・ノードのタグ名）で、ROWSET と表されます。

DEFAULT_ERRORTAG

発生したエラーを囲むデフォルトのタグで、ERROR と表されます。

DEFAULT_ROWIDATTR

データベース・レコードに対応する XML 要素のカーディナリティ属性のデフォルト名で、NUM と表されます。

DEFAULT_ROWTAG

データベース・レコードに対応する要素のデフォルトのタグ名で、ROW と表されます。

DEFAULT_DATE_FORMAT

デフォルトの日付マスクで、'MM/dd/yyyy HH:mm:ss' と表されます。

ALL_ROWS

すべての行を出力する必要があることを示します。

NONE

出力に XML メタデータ（DTD など）を含めないことを指定します。

DTD

DTD の生成を指定します。

LOWER_CASE

小文字のタグ名を使用します。

UPPER_CASE

大文字のタグ名を使用します。

ファンクションおよびプロシージャ（一覧）

プロシージャ

`closeContext(ctxType)`
指定した問合せコンテキストのクローズまたは割当てを行います。

ファンクション

`getDTD(ctxType, BOOLEAN := false)`

戻り値

CLOB

初期化に使用される SQL 問合せに基づいて、DTD を生成します。

プロシージャ

`getDTD(ctxType, CLOB, BOOLEAN := false)`
初期化に使用される SQL 問合せに基づいて、DTD を生成します。

プロシージャ

`getExceptionContent(ctxType, NUMBER, VARCHAR2)`

ファンクション

`getXML(ctxType, NUMBER := NONE)`

戻り値

CLOB

XML ドキュメントを生成します。

プロシージャ

`getXML(ctxType, CLOB, NUMBER := NONE)`
XML ドキュメントを生成します。

ファンクション

`newContext(VARCHAR2)`

戻り値

ctxType

問合せコンテキストを作成し、コンテキスト・ハンドルを戻します。

ファンクション

`newContext(CLOB)`

戻り値

ctxType

問合せコンテキストを作成し、コンテキスト・ハンドルを戻します。

プロシージャ

`propagateOriginalException(ctxType, BOOLEAN)`
例外が発生した場合、例外を OracleXMLSQLException でラップするかわりに、呼び出された実際の例外を発生させる必要があることを XSU に通知します。

プロシージャ

`setBindValue(ctxType, VARCHAR2, VARCHAR2)`
指定したバインド名の値を設定します。

プロシージャ

`setCollIdAttrName(ctxType, VARCHAR2`
コレクション要素の区切りタグの ID 属性名を設定します。

プロシージャ

`setDataHeader(ctxType, CLOB := null, VARCHAR2 := null)`
XML データ・ヘッダーを設定します。

プロシージャ

`setDateFormat(ctxType, VARCHAR2)`
XML ドキュメントに生成日付の書式を設定します。

プロシージャ

`setErrorTag(ctxType, VARCHAR2)`
XML エラー・ドキュメントを囲むためのタグを設定します。

プロシージャ

`setMaxRows (ctxType, NUMBER)`
XML に変換される行の最大数を設定します。

プロシージャ

`setMetaHeader(ctxType, CLOB := null)`
XML メタ・ヘッダーを設定します。

プロシージャ

`setRaiseException(ctxType, BOOLEAN)`
呼び出された例外を発生させるように XSU に通知します。

プロシージャ

`setRaiseNoRowsException(ctxType, BOOLEAN)`
生成された XML ドキュメントが空の場合に、OracleXMLNoRowsException が発生するかどうかを XSU に通知します。

プロシージャ

`setRowIdAttrName(ctxType, VARCHAR2)`
行の囲みタグの ID 属性名を設定します。

プロシージャ

`setRowIdAttrValue(ctxType, VARCHAR2)`
列値が行の囲みタグの ID 属性に割り当てられるスカラー列を指定します。

プロシージャ

`setRowsetTag(ctxType, VARCHAR2)`
XML データセットを囲むタグを設定します。

プロシージャ

`setRowTag(ctxType, VARCHAR2)`
データベースに対応する XML 要素を囲むタグを設定します。

プロシージャ

`setSkipRows(ctxType, NUMBER)`
スキップする行数を設定します。

プロシージャ

`setStylesheetHeader(ctxType, VARCHAR2, VARCHAR2 := 'text/xsl')`
スタイルシート・ヘッダーを設定します。

プロシージャ

`setTagCase(ctxType, NUMBER)`
生成された XML タグの大文字 / 小文字を指定します。

プロシージャ

`setXSLT(ctxType, VARCHAR2, VARCHAR2 := null)`
生成された XML に適用するスタイルシートを登録します。

プロシージャ

`setXSLT(ctxType, CLOB, VARCHAR2 := null)`
生成された XML に適用するスタイルシートを登録します。

プロシージャ

`useNullAttributeIndicator(ctxType, BOOLEAN)`
NULL であることを示すために、XML 属性を使用するか、XML ドキュメント内の特定のエンティティを省略するかを指定します。

ファンクションおよびプロシージャ（詳細）

newContext

ファンクション

`newContext(sqlQuery IN VARCHAR2)`

戻り値

ctxType

問合せコンテキストを作成し、コンテキスト・ハンドルを戻します。

パラメータ

`sqlQuery` - SQL 問合せ（XML への変換結果）

newContext

ファンクション

`newContext(sqlQuery IN CLOB)`

戻り値

ctxType

問合せコンテキストを作成し、コンテキスト・ハンドルを戻します。

パラメータ

`sqlQuery` - SQL 問合せ（XML への変換結果）

closeContext

プロシージャ

`closeContext(ctxHdl IN ctxType)`
特定の問合せコンテキストのクローズまたは割当て解除を行います。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

setRowsetTag

プロシージャ

`setRowsetTag(ctxHdl IN ctxType, tag IN VARCHAR2)`
XML データセットを囲むタグを設定します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`tag` - タグ名

setRowTag

プロシージャ

`setRowTag(ctxHdl IN ctxType, tag IN VARCHAR2)`
データベース・レコードに対応する XML 要素を囲むタグを設定します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`tag` - タグ名

setErrorTag

プロシージャ

`setErrorTag(ctxHdl IN ctxType, tag IN VARCHAR2)`
XML エラー・ドキュメントを囲むタグを設定します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`tag` - タグ名

setRowIdAttrName

プロシージャ

`setRowIdAttrName(ctxHdl IN ctxType, attrName IN VARCHAR2)`
行の囲みタグの ID 属性名を設定します。タグに NULL または空の文字列を渡すと、行 ID 属性は省略されます。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`attrName` - 属性名

setRowIdAttrValue

プロシージャ

`setRowIdAttrValue(ctxHdl IN ctxType, colName IN VARCHAR2)`
行の囲みタグの ID 属性に値が割り当てられるスカラー列を指定します。colName に NULL または空の文字列を渡すと、行 ID 属性には行カウント値（0, 1, 2 など）が割り当てられます。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`colName` - 行 ID 属性に割り当てられる値を持つ列

setCollIdAttrName

プロシージャ

`setCollIdAttrName(ctxHdl IN ctxType, attrName IN VARCHAR2)`
コレクション要素の区切りタグの ID 属性名を設定します。タグに NULL または空の文字列を渡すと、行 ID 属性は省略されます。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`attrName` - 属性名

useNullAttributeIndicator

プロシージャ

`useNullAttributeIndicator(ctxHdl IN ctxType, flag IN BOOLEAN)`
NULL であることを示すために、XML 属性を使用するか、XML ドキュメント内の特定のエンティティを省略するかを指定します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`flag` - NULL を示すために属性を使用するかどうか

setTagCase

プロシージャ

`setTagCase(ctxHdl IN ctxType, tCase IN NUMBER)`

生成された XML タグの大文字 / 小文字を指定します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`tCase` - タグの大文字 / 小文字（0 - 混在、1 - 小文字、2 - 大文字）

setDateFormat

プロシージャ

`setDateFormat(ctxHdl IN ctxType, mask IN VARCHAR2)`

XML ドキュメントに生成日付の書式を設定します。日付書式パターン（日付マスク）の構文は、java.text.SimpleDateFormat クラスの要件を満たす必要があります。マスクに NULL または空の文字列を設定すると、デフォルトのマスク DEFAULT_DATE_FORMAT が使用されます。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`mask` - 日付マスク

setMaxRows

プロシージャ

`setMaxRows (ctxHdl IN ctxType, rows IN NUMBER)`

XML に変換される行の最大数を設定します。デフォルトでは、最大数は設定されていません。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`rows` - 生成する行の最大数

setSkipRows

プロシージャ

`setSkipRows(ctxHdl IN ctxType, rows IN NUMBER)`
スキップする行数を設定します。デフォルトでは、0（ゼロ）行がスキップされます。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`rows` - スキップする行数

setStylesheetHeader

プロシージャ

`setStylesheetHeader(ctxHdl IN ctxType, uri IN VARCHAR2, type IN VARCHAR2 := 'text/xsl')`
生成された XML ドキュメントにスタイルシート・ヘッダー（スタイルシート処理命令）を設定します。URI 引数に NULL を渡すと、スタイルシート・ヘッダーおよびスタイルシート型の設定が解除されます。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`uri` - スタイルシートの URI

`type` - スタイルシートの型（デフォルトは 'text/xsl'）

setXSLT

プロシージャ

`setXSLT(ctxHdl IN ctxType, uri IN VARCHAR2, ref IN VARCHAR2 := null)`
生成された XML に適用するスタイルシートを登録します。スタイルシートがすでに登録されている場合は、新しいスタイルシートに置き換えられます。スタイルシートの登録を解除するには、URI 引数に NULL を指定します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`uri` - スタイルシートの URI

`ref` - インクルード、インポートおよび外部エンティティの URL

setXSLT

プロシージャ

`setXSLT(ctxHdl IN ctxType, stylesheet CLOB, ref IN VARCHAR2 := null)`
生成された XML に適用するスタイルシートを登録します。スタイルシートがすでに登録されている場合は、新しいスタイルシートに置き換えられます。スタイルシートの登録を解除するには、stylesheet 引数に空の文字列を指定します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`stylesheet` - スタイルシート

`ref` - インクルード、インポートおよび外部エンティティの URL

setBindValue

プロシージャ

`setBindValue(ctxHdl IN ctxType, bindName IN VARCHAR2, bindValue IN VARCHAR2)`
指定したバインド名の値を設定します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`bindName` - バインド名

`bindValue` - バインド値

setMetaHeader

プロシージャ

`setMetaHeader(ctxHdl IN ctxType, header IN CLOB := null)`
XML メタ・ヘッダーを設定します。ヘッダーが設定されると、このオブジェクトが生成した各 XML ドキュメントのメタデータ部（DTD または XMLSchema）の先頭にヘッダーが挿入されます。最後に指定されたメタ・ヘッダーが使用されることに注意してください。また、ヘッダーに NULL を指定すると、パラメータによってメタ・ヘッダーの設定が解除されます。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`header` - ヘッダー

setDataHeader

プロシージャ

`setDataHeader(ctxHdl IN ctxType, header IN CLOB := null, tag IN VARCHAR2 := null)`
XML データ・ヘッダーを設定します。データ・ヘッダーは、問合せで生成された XML エンティティ（行セット）の先頭に追加される XML エンティティです。2 つのエンティティが、docTag 引数を介して指定されたタグによって囲まれます。最後に指定されたデータ・ヘッダーが使用されることに注意してください。また、ヘッダーに NULL を指定すると、パラメータによってデータ・ヘッダーの設定が解除されます。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`header` - ヘッダー

`tag` - データ・ヘッダーおよび行セットを囲むタグ

setRaiseException

プロシージャ

`setRaiseException(ctxHdl IN ctxType, flag IN BOOLEAN)`
呼び出された例外を発生させるように XSU に通知します。これがコールされない場合、またはフラグ引数に FALSE が渡された場合、XSU は SQL 例外をキャッチし、例外のメッセージから XML ドキュメントを生成します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`flag` - 呼び出された例外を発生させるかどうか

setRaiseNoRowsException

プロシージャ

`setRaiseNoRowsException(ctxHdl IN ctxType, flag IN BOOLEAN)`
生成された XML ドキュメントが空の場合に、OracleXMLNoRowsException が発生するかどうかを XSU に通知します。デフォルトでは、例外は発生しません。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`flag` - データがない場合に、OracleXMLNoRowsException が発生するかどうか

propagateOriginalException

プロシージャ

`propagateOriginalException(ctxHdl IN ctxType, flag IN BOOLEAN)`
例外が発生した場合、例外を OracleXMLSQLException でラップするかわりに、呼び出された実際の例外を発生させる必要があることを XSU に通知します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`flag` - 元の例外を継承するかどうか

getExceptionContent

プロシージャ

`getExceptionContent(ctxHdl IN ctxType, errNo OUT NUMBER, errMsg OUT VARCHAR2)`
発生した例外のエラー・コードおよびエラー・メッセージ（SQL エラー・コード）を引数を介して戻します。これによって、例外が呼び出されるたびに Oracle8*i* JVM が例外を発生させることを防止できます。そのため、PL/SQL をレンダリングすると、元の例外にアクセスできません。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`errNo` - エラー番号

`errMsg` - エラー・メッセージ

getDTD

ファンクション

`getDTD(ctxHdl IN ctxType, withVer IN BOOLEAN := false)`

戻り値

CLOB

コンテキストを初期化するために使用される SQL 問合せに基づいて、DTD を生成します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`withVer` - バージョン情報を生成するかどうか

getDTD

プロシージャ

`getDTD(ctx IN ctxType, xDoc IN CLOB, withVer IN BOOLEAN := false)`
コンテキストを初期化するために使用される SQL 問合せに基づいて、DTD を生成します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`xDoc` - 生成された XML ドキュメントを書き込む LOB

`withVer` - バージョン情報を生成するかどうか

getXML

ファンクション

`getXML(ctxHdl IN ctxType, metaType IN NUMBER := NONE)`

戻り値

CLOB

コンテキストを初期化するために使用される SQL 問合せに基づいて、XML ドキュメントを生成します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`metaType` - XML メタ・データ型（NONE または DTD）

getXML

プロシージャ

`getXML(ctxHdl IN ctxType, xDoc IN CLOB, metaType IN NUMBER := NONE)`
コンテキストを初期化するために使用される SQL 問合せに基づいて、XML ドキュメントを生成します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`xDoc` - 生成された XML ドキュメントを書き込む LOB

`metaType` - XML メタ・データ型（NONE または DTD）

DBMS_XMLSave

型

ctxType

問合せコンテキスト・ハンドルの型です。DBMS_ XMLSave.newContext() の戻り型です。

定数

DEFAULT_ROWTAG

データベース・レコードに対応する要素のデフォルトのタグ名で、ROW と表されます。

DEFAULT_DATE_FORMAT

デフォルトの日付マスクで、'MM/dd/yyyy HH:mm:ss' と表されます。

MATCH_CASE

XML 要素をデータベース・エンティティへマッピングする場合、XSU が大文字 / 小文字を区別するように指定します。

IGNORE_CASE

XML 要素をデータベース・エンティティへマッピングする場合、XSU が大文字 / 小文字を区別しないように指定します。

ファンクションおよびプロシージャ（一覧）

プロシージャ

`clearKeyColumnList(ctxType)`
キー列リストを消去します。

プロシージャ

`clearUpdateColumnList(ctxType)`
更新列リストを消去します。

プロシージャ

`closeContext(ctxType)`
特定の保存コンテキストのクローズまたは割当て解除を行います。

ファンクション

deleteXML(ctxType, CLOB)

戻り値

NUMBER

コンテキストの作成時に指定した表から、XML ドキュメントのデータで指定したレコードを削除します。

ファンクション

deleteXML(ctxType, VARCHAR2)

戻り値

NUMBER

コンテキストの作成時に指定した表から、XML ドキュメントのデータで指定したレコードを削除します。

プロシージャ

getExceptionContent(ctxType, NUMBER, VARCHAR2)
発生した例外のエラー・コードおよびエラー・メッセージを引数を介して戻します。

ファンクション

insertXML(ctxType, CLOB)

戻り値

NUMBER

コンテキストの作成時に指定した表に、XML ドキュメントを挿入します。

ファンクション

insertXML(ctxType, VARCHAR2)

戻り値

NUMBER

コンテキストの作成時に指定した表に、XML ドキュメントを挿入します。

ファンクション

`newContext(targetTable IN VARCHAR2)`

戻り値

ctxType

保存コンテキストを作成し、コンテキスト・ハンドルを戻します。

プロシージャ

`propagateOriginalException(ctxType, BOOLEAN)`
例外が発生した場合、例外を OracleXMLSQLException でラップするかわりに、呼び出された実際の例外を発生させる必要があることを XSU に通知します。

プロシージャ

`setBatchSize(ctxType, NUMBER)`
DML 操作中に使用されるバッチ・サイズを変更します。

プロシージャ

`setCommitBatch(ctxType, NUMBER)`
コミットのバッチ・サイズを設定します。

プロシージャ

`setDateFormat(ctxType, VARCHAR2)`
XML ドキュメントの日付書式を、XSU に通知します。

プロシージャ

`setIgnoreCase(ctxType, NUMBER)`
XSU は、XML 要素をデータベースへマッピングします。

プロシージャ

`setKeyColumn(ctxType, VARCHAR2)`
キー列リストに列を追加します。

プロシージャ

`setRowTag(ctxType, VARCHAR2)`
データベースに対応する XML 要素を囲むために XML ドキュメントで使用するタグを指定します。

プロシージャ

`setUpdateColumn(ctxType, VARCHAR2)`
更新列リストに列を追加します。

プロシージャ

`getExceptionContent(ctxType, NUMBER, VARCHAR2)`
XML ドキュメントのデータを使用して、コンテキストの作成時に指定した表を更新します。

プロシージャ

`propagateOriginalException(ctxType, BOOLEAN)`
XML ドキュメントのデータを使用して、コンテキストの作成時に指定した表を更新します。

ファンクションおよびプロシージャ（詳細）

newContext

ファンクション

`newContext(targetTable IN VARCHAR2)`

戻り値

ctxType

保存コンテキストを作成し、コンテキスト・ハンドルを戻します。

パラメータ

`targetTable` - XML ドキュメントをロードするターゲット表

closeContext

プロシージャ

`closeContext(ctxHdl IN ctxType)`
特定の保存コンテキストのクローズまたは割当て解除を行います。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

setRowTag

プロシージャ

`setRowTag(ctxHdl IN ctxType, tag IN VARCHAR2)`
データベース・レコードに対応する XML 要素を囲むために XML ドキュメントで使用するタグを指定します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`tag` - タグ名

setIgnoreCase

プロシージャ

`setIgnoreCase(ctxHdl IN ctxType, flag IN NUMBER)`
XSU は、要素名（XML タグ）に基づいて、XML 要素をデータベースの列 / 属性にマッピングします。このファンクションは、XSU が大文字 / 小文字を区別しないように通知します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`flag` - XML ドキュメント内でタグの大文字 / 小文字を無視するかどうか（0 - FALSE、1 - TRUE）

setDateFormat

プロシージャ

`setDateFormat(ctxHdl IN ctxType, mask IN VARCHAR2)`
XML ドキュメントに生成日付の書式を設定します。日付書式パターン（日付マスク）の構文は、java.text.SimpleDateFormat クラスの要件を満たす必要があります。マスクに NULL または空の文字列を設定すると、デフォルトのマスク DEFAULT_DATE_FORMAT が使用されます。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`mask` - 日付マスク

setBatchSize

プロシージャ

`setBatchSize(ctxHdl IN ctxType, batchSize IN NUMBER)`

DML 操作中に使用されるバッチ・サイズを変更します。挿入、更新または削除を実行する場合、別々の文ではなく、1 回の処理でそれらの操作が実行されるように、バッチ処理を行う方が適しています。欠点は、その操作を実行する前に、すべてのバインド値を保持するためのメモリーがさらに多く必要になることです。バッチ処理を行う場合、バッチ処理の実行後にのみコミットが発生することに注意してください。そのため、バッチ内の 1 文が正常に実行されなかった場合、バッチ処理全体がロールバックされます。バッチ処理によるパフォーマンスの向上とを考えると、大きな問題ではありませんが、これを防止するには、バッチ・サイズを 1 に設定します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`batchSize` - バッチ・サイズ

参照

DEFAULT_BATCH_SIZE

setCommitBatch

プロシージャ

`setCommitBatch(ctxHdl IN ctxType, batchSize IN NUMBER)`

コミットのバッチ・サイズを設定します。コミットのバッチ・サイズとは、実行後にコミットする必要がある挿入の回数または記録を意味します。commitBatch が 1 未満またはセッションが自動コミット・モードの場合、XSU は明示的なコミットを行わないことに注意してください。コミットのデフォルトのバッチ・サイズは 0（ゼロ）です。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`batchSize` - コミットのバッチ・サイズ

setUpdateColumn

プロシージャ

`setUpdateColumn(ctxHdl IN ctxType, colName IN VARCHAR2)`
更新列リストに列を追加します。挿入の場合、デフォルトでは、表のすべての列に値が挿入されます。更新の場合、デフォルトでは、XML ドキュメントの ROW 要素にあるタグに対応する列のみが更新されます。更新列リストを指定すると、このリストを構成する列のみが更新または挿入されます。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`colName` - 更新列リストに追加される列

clearUpdateColumnList

プロシージャ

`clearUpdateColumnList(ctxHdl IN ctxType)`
更新列リストを消去します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

参照

setUpdateColumn

setKeyColumn

プロシージャ

`setKeyColumn(ctxHdl IN ctxType, colName IN VARCHAR2)`
キー列リストに列を追加します。更新または削除の場合、UPDATE/DELETE 文の WHERE 句を構成するキー列リストの列です。キー列リストは更新される前に指定する必要がありますが、削除操作ではオプションです。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`colName` - キー列リストに追加される列

clearKeyColumnList

プロシージャ

`clearKeyColumnList(ctxHdl IN ctxType)`
キー列リストを消去します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

参照

setKeyColumn

insertXML

ファンクション

`insertXML(ctxHdl IN ctxType, xDoc IN VARCHAR2)`

戻り値

NUMBER - 挿入された行の数

コンテキストの作成時に指定した表に、XML ドキュメントを挿入します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`xDoc` - XML ドキュメントを含む文字列

insertXML

ファンクション

`insertXML(ctxHdl IN ctxType, xDoc IN CLOB)`

戻り値

NUMBER - 挿入された行の数

コンテキストの作成時に指定した表に、XML ドキュメントを挿入します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`xDoc` - XML ドキュメントを含む文字列

updateXML

ファンクション

```
updateXML(ctxHdl IN ctxType, xDoc IN VARCHAR2)
```

戻り値

NUMBER - 更新された行の数

XML ドキュメントのデータを使用して、コンテキストの作成時に指定した表を更新します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`xDoc` - XML ドキュメントを含む文字列

updateXML

ファンクション

```
updateXML(ctxHdl IN ctxType, xDoc IN CLOB)
```

戻り値

NUMBER - 更新された行の数

XML ドキュメントのデータを使用して、コンテキストの作成時に指定した表を更新します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`xDoc` - XML ドキュメントを含む文字列

deleteXML

ファンクション

```
deleteXML(ctxHdl IN ctxType, xDoc IN VARCHAR2)
```

戻り値

NUMBER - 削除された行の数

コンテキストの作成時に指定した表から、XML ドキュメントのデータで指定したレコードを削除します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`xDoc` - XML ドキュメントを含む文字列

deleteXML

ファンクション

```
deleteXML(ctxHdl IN ctxType, xDoc IN CLOB)
```

戻り値

NUMBER - 削除された行の数

コンテキストの作成時に指定した表から、XML ドキュメントのデータで指定したレコードを削除します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`xDocl` - XML ドキュメントを含む文字列

propagateOriginalException

プロシージャ

`propagateOriginalException(ctxHdl IN ctxType, flag IN BOOLEAN)`
例外が発生した場合、例外を OracleXMLSQLException でラップするかわりに、呼び出された実際の例外を発生させる必要があることを XSU に通知します。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`flag` - 元の例外を継承するかどうか（0 - FALSE、1 - TRUE）

getExceptionContent

プロシージャ

`getExceptionContent(ctxHdl IN ctxType, errNo OUT NUMBER, errMsg OUT VARCHAR2)`
発生した例外のエラー・コードおよびエラー・メッセージ（SQL エラー・コード）を引数を介して戻します。これによって、例外が呼び出されるたびに Oracle8*i* JVM が例外を発生させることを防止できます。そのため、PL/SQL をレンダリングすると、元の例外にアクセスできません。

パラメータ

`ctxHdl` - コンテキスト・ハンドル

`errNo` - エラー番号

`errMsg` - エラー・メッセージ

索引

C

close() - oracle.xml.sql.dml.OracleXMLSave.close(), 6-4
close() - oracle.xml.sql.query.OracleXMLQuery.close(), 6-12
collectTimingInfo(boolean) - oracle.xml.sql.dml.OracleXMLSave.collectTimingInfo(boolean), 6-4
createTimestamp(String) - oracle.xml.sql.dml.OracleXMLSave.createTimestamp(java.lang.String), 6-4

D

DATE_FORMAT - oracle.xml.sql.dml.OracleXMLSave.DATE_FORMAT, 6-4
DEFAULT_BATCH_SIZE - oracle.xml.sql.dml.OracleXMLSave.DEFAULT_BATCH_SIZE, 6-4
deleteXML(Document) - oracle.xml.sql.dml.OracleXMLSave.deleteXML(org.w3c.dom.Document), 6-4
deleteXML(InputStream) - oracle.xml.sql.dml.OracleXMLSave.deleteXML(java.io.InputStream), 6-4
deleteXML(String) - oracle.xml.sql.dml.OracleXMLSave.deleteXML(java.lang.String), 6-4
deleteXML(URL) - oracle.xml.sql.dml.OracleXMLSave.deleteXML(java.net.URL), 6-5
DTD - oracle.xml.sql.query.OracleXMLQuery.DTD, 6-11

E

ERROR_TAG - oracle.xml.sql.query.OracleXMLQuery.ERROR_TAG, 6-11

G

getErrorCode() - oracle.xml.sql.OracleXMLSQLException.getErrorCode(), 6-20
getInternalObj() - oracle.xml.sql.query.OracleXMLQuery.getInternalObj(), 6-12
getParentException() - oracle.xml.sql.OracleXMLSQLException.getParentException(), 6-20
getStructVal(Node, OracleColumnName) - oracle.xml.sql.dml.OracleXMLSave.getStructVal(org.w3c.dom.Node, oracle.xml.sql.name.OracleColumnName), 6-5
getXMLDOM() - oracle.xml.sql.query.OracleXMLQuery.getXMLDOM(), 6-12
getXMLDOM(Node, int) - oracle.xml.sql.query.OracleXMLQuery.getXMLDOM(oracle.xml.sql.query.Node, int), 6-12
getXMLErrorString() - oracle.xml.sql.OracleXMLSQLException.getXMLErrorString(), 6-20
getXMLMetaData(int, boolean) - oracle.xml.sql.query.OracleXMLQuery.getXMLMetaData(int, boolean), 6-13

getXMLMetaData(int, boolean, OracleXMLDocGen) - oracle.xml.sql.query.OracleXMLQuery. getXMLMetaData(int, boolean, oracle.xml.sql.docgen.OracleXMLDocGen), 6-13
getXMLSQLErrorString() - oracle.xml.sql.OracleXMLSQLException. getXMLSQLErrorString(), 6-20
getXMLString() - oracle.xml.sql.query.OracleXMLQuery. getXMLString(), 6-13
getXMLString(boolean) - oracle.xml.sql.query.OracleXMLQuery. getXMLString(boolean), 6-13
getXMLString(Node) - oracle.xml.sql.query.OracleXMLQuery. getXMLString(oracle.xml.sql.query.Node), 6-13
getXMLString(Node, int) - oracle.xml.sql.query.OracleXMLQuery. getXMLString(oracle.xml.sql.query.Node, int), 6-13

I

insertXML(Document) - oracle.xml.sql.dml.OracleXMLSave.insertXML (org.w3c.dom.Document), 6-5
insertXML(InputStream) - oracle.xml.sql.dml.OracleXMLSave.insertXML (java.io.InputStream), 6-5
insertXML(String) - oracle.xml.sql.dml.OracleXMLSave.insertXML (java.lang.String), 6-5
insertXML(URL) - oracle.xml.sql.dml.OracleXMLSave.insertXML (java.net.URL), 6-5

M

MAXROWS_ALL - oracle.xml.sql.query.OracleXMLQuery. MAXROWS_ALL, 6-11
MAXROWS_DEFAULT - oracle.xml.sql.query.OracleXMLQuery. MAXROWS_DEFAULT, 6-11

N

NONE - oracle.xml.sql.query.OracleXMLQuery.NONE, 6-11

O

Oracle, 6-1
OracleXMLQuery(Connection, ResultSet) - oracle.xml.sql.query.OracleXMLQuery.OracleXML Query(java.sql.Connection, java.sql.ResultSet), 6-12
OracleXMLQuery(Connection, String) - oracle.xml.sql.query.OracleXMLQuery.OracleXML Query(java.sql.Connection, java.lang.String), 6-12
OracleXMLQuery(OracleXMLDataSet) - oracle.xml.sql.query.OracleXMLQuery.OracleXML Query(oracle.xml.sql.dataset.OracleXMLDataSet), 6-12
OracleXMLSave(Connection, OracleColumnName[]) - oracle.xml.sql.dml.OracleXMLSave. OracleXMLSave(java.sql.Connection, oracle.xml.sql.name.OracleColumnName[]), 6-4
OracleXMLSave(Connection, String) - oracle.xml.sql.dml.OracleXMLSave. OracleXMLSave(java.sql.Connection, java.lang.String), 6-4
oracle.xml.sql.dml - oracle.xml.sql.dml, 6-1
OracleXMLSQLException(Exception) - oracle.xml.sql.OracleXMLSQLException. OracleXMLSQLException(java.lang.Exception), 6-19
OracleXMLSQLException(Exception, String) - oracle.xml.sql.OracleXMLSQLException. OracleXMLSQLException(java.lang.Exception, java.lang.String), 6-19
OracleXMLSQLException(String) - oracle.xml.sql.OracleXMLSQLException. OracleXMLSQLException(java.lang.String), 6-19
OracleXMLSQLException(String, Exception) - oracle.xml.sql.OracleXMLSQLException. OracleXMLSQLException(java.lang.String, java.lang.Exception), 6-19

OracleXMLSQLException(String, Exception, String) - oracle.xml.sql.OracleXMLSQLException. OracleXMLSQLException(java.lang.String, java.lang.Exception, java.lang.String), 6-19
OracleXMLSQLException(String, int) - oracle.xml.sql.OracleXMLSQLException. OracleXMLSQLException(java.lang.String, int), 6-20
OracleXMLSQLException(String, int, String) - oracle.xml.sql.OracleXMLSQLException. OracleXMLSQLException(java.lang.String, int, java.lang.String), 6-20
OracleXMLSQLException(String, String) - oracle.xml.sql.OracleXMLSQLException. OracleXMLSQLException(java.lang.String, java.lang.String), 6-20
OracleXMLSQLNoRowsException() - oracle.xml.sql.OracleXMLSQLNoRowsException. OracleXMLSQLNoRowsException(), 6-22
OracleXMLSQLNoRowsException(String) - oracle.xml.sql.OracleXMLSQLNoRowsException. OracleXMLSQLNoRowsException(java.lang.String), 6-22

R

ROW_TAG - oracle.xml.sql.query.OracleXMLQuery. ROW_TAG, 6-11
ROWIDATTR_TAG - oracle.xml.sql.query.OracleXMLQuery. ROWIDATTR_TAG, 6-11
ROWSET_TAG - oracle.xml.sql.query.OracleXMLQuery. ROWSET_TAG, 6-11

S

SCHEMA - oracle.xml.sql.query.OracleXMLQuery.SCHEMA, 6-11
setBatchSize(int) - oracle.xml.sql.dml.OracleXMLSave.setBatchSize (int), 6-5
setCollIdAttr(String) - oracle.xml.sql.query.OracleXMLQuery. setCollIdAttr(java.lang.String), 6-14
setCollIdAttrName(String) - oracle.xml.sql.query.OracleXMLQuery. setCollIdAttrName(java.lang.String), 6-14
setCommitBatch(int) - oracle.xml.sql.dml.OracleXMLSave. setCommitBatch(int), 6-5
setDataHeader(Reader, String) - oracle.xml.sql.query.OracleXMLQuery. setDataHeader(java.io.Reader, java.lang.String), 6-14
setDateFormat(String) - oracle.xml.sql.dml.OracleXMLSave. setDateFormat(java.lang.String), 6-6
setDateFormat(String) - oracle.xml.sql.query.OracleXMLQuery.setDateFor mat(java.lang.String), 6-14
setEncoding(String) - oracle.xml.sql.query.OracleXMLQuery. setEncoding(java.lang.String), 6-14
setErrorTag(String) - oracle.xml.sql.OracleXMLSQLException. setErrorTag(java.lang.String), 6-20
setErrorTag(String) - oracle.xml.sql.query.OracleXMLQuery. setErrorTag(java.lang.String), 6-15
setException(Exception) - oracle.xml.sql.query.OracleXMLQuery. setException(java.lang.Exception), 6-15
setIgnoreCase(boolean) - oracle.xml.sql.dml.OracleXMLSave.setIgnoreCase (boolean), 6-6
setKeyColumnList(String[]) - oracle.xml.sql.dml.OracleXMLSave. setKeyColumnList(java.lang.String[]), 6-6
setMaxRows(int) - oracle.xml.sql.query.OracleXMLQuery. setMaxRows(int), 6-15
setMetaHeader(Reader) - oracle.xml.sql.query.OracleXMLQuery. setMetaHeader(java.io.Reader), 6-15
setRaiseException(boolean) - oracle.xml.sql.query.OracleXMLQuery. setRaiseException(boolean), 6-15
setRaiseNoRowsException(boolean) - oracle.xml.sql.query.OracleXMLQuery. setRaiseNoRowsException(boolean), 6-15

setRowIdAttrName(String) -
oracle.xml.sql.query.OracleXMLQuery.
setRowIdAttrName(java.lang.String), 6-15
setRowIdAttrValue(String) -
oracle.xml.sql.query.OracleXMLQuery.
setRowIdAttrValue(java.lang.String), 6-16
setRowsetTag(String) -
oracle.xml.sql.query.OracleXMLQuery.
setRowsetTag(java.lang.String), 6-16
setRowTag(String) -
oracle.xml.sql.dml.OracleXMLSave.setRowTag
(java.lang.String), 6-6
setRowTag(String) -
oracle.xml.sql.query.OracleXMLQuery.setRowTag
(java.lang.String), 6-16
setSkipRows(int) -
oracle.xml.sql.query.OracleXMLQuery.
setSkipRows(int), 6-16
setStyleSheet(String) -
oracle.xml.sql.query.OracleXMLQuery.
setStyleSheet(java.lang.String), 6-16
setStylesheetHeader(String) -
oracle.xml.sql.query.OracleXMLQuery.
setStylesheetHeader(java.lang.String), 6-16
setStylesheetHeader(String, String) -
oracle.xml.sql.query.OracleXMLQuery.
setStylesheetHeader(java.lang.String,
java.lang.String), 6-16
setUpdateColumnList(String[]) -
oracle.xml.sql.dml.OracleXMLSave.
setUpdateColumnList(java.lang.String[]), 6-6
setXSLT(Reader, String), 6-16
setXSLT(String, String), 6-17
SKIPROWS_ALL -
oracle.xml.sql.query.OracleXMLQuery.
SKIPROWS_ALL, 6-11
SKIPROWS_DEFAULT -
oracle.xml.sql.query.OracleXMLQuery.
SKIPROWS_DEFAULT, 6-12

U

updateXML(Document) -
oracle.xml.sql.dml.OracleXMLSave.updateXML
(org.w3c.dom.Document), 6-7
updateXML(InputStream) -
oracle.xml.sql.dml.OracleXMLSave.updateXML
(java.io.InputStream), 6-7
updateXML(String) -
oracle.xml.sql.dml.OracleXMLSave.updateXML
(java.lang.String), 6-7
updateXML(URL) -
oracle.xml.sql.dml.OracleXMLSave.updateXML
(java.net.URL), 6-7
useLowerCaseTagNames() -
oracle.xml.sql.query.OracleXMLQuery.
useLowerCaseTagNames(), 6-17
useNullAttributeIndicator(boolean) -
oracle.xml.sql.query.OracleXMLQuery.
useNullAttributeIndicator(boolean), 6-17
useUpperCaseTagNames() -
oracle.xml.sql.query.OracleXMLQuery.
useUpperCaseTagNames(), 6-17