Oracle *inter*Media Java Classes

ユーザーズ・ガイドおよびリファレンス

リリース 9.2

2002 年 7 月

部品番号 : J06312-01

ORACLE®

Oracle *inter*Media Java Classes ユーザーズ・ガイドおよびリファレンス , リリース 9.2

部品番号：J06312-01

原本名：Oracle *inter*Media Java Classes User's Guide and Reference, Release 9.2

原本部品番号：A96121-01

原本著者：Sue Pelski

原本協力者：Sue Mavris、Simon Oxbury、Manjari Yalavarthy、Susan Shepard、Rod Ward

Printed in Japan.

目次

例目次

はじめに

このマニュアルでは、Oracle *inter*Media Java Classes の使用方法について説明します。

対象読者

このマニュアルは、Oracle データベースでマルチメディア・データを格納、検索、操作する開発者およびデータベース管理者（マルチメディアに特化したアプリケーションの開発者を含む）を対象としています。このマニュアルのユーザーは、Java および JDBC の経験が必要です。

このマニュアルの構成

このマニュアルは、次の章および付録で構成されています。

章番号	説明
第 1 章	一般的な概要を説明します。
第 2 章	Java クラスのインストールに含まれる例の情報を示します。
第 3 章	OrdAudio クラスのリファレンス情報を示します。
第 4 章	OrdDoc クラスのリファレンス情報を示します。
第 5 章	OrdImage クラスのリファレンス情報を示します。
第 6 章	OrdImageSignature クラスのリファレンス情報を示します。
第 7 章	JAI 用に設計された入力ストリームおよび出力ストリームのリファレンス情報を示します。
第 8 章	OrdVideo クラスのリファレンス情報を示します。
第 9 章	サーブレットおよび JSP 用の Java クラスのリファレンス情報を示します。
付録 A	Java クラスに含まれるサンプル・ファイルの実行方法を示します。
付録 B	発生する可能性がある例外およびエラーについて説明します。
付録 C	使用されなくなったメソッドを示します。

関連文書

このマニュアルは、単独での使用を目的としていません。『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』の補足情報として使用してください。Java インタフェースを使用して *inter*Media オブジェクトを適切に操作するには、両方のマニュアルが必要です。

開発環境で *inter*Media を使用する場合の詳細は、Oracle マニュアル・セットの次のマニュアルを参照してください。

- 『Oracle Call Interface プログラマーズ・ガイド』
- 『Oracle9*i* データベース概要』
- 『PL/SQL ユーザーズ・ガイドおよびリファレンス』

JDBC の使用方法の詳細は、『Oracle9*i* JDBC 開発者ガイドおよびリファレンス』を参照してください。

Javadoc 形式のリファレンス情報は、Oracle API ドキュメント（Javadoc）を参照してください。次のディレクトリには、Oracle *inter*Media Java Classes および Oracle *inter*Media Java Classes for Servlets and JSP のリファレンス情報が Javadoc 形式で含まれている ZIP ファイルが入っています。

`<ORACLE_HOME>/ord/im/javadoc` （UNIX の場合）

`<ORACLE_HOME>¥ord¥im¥javadoc` （Windows NT の場合）

Java の詳細は、Sun 社の次の URL で提供される API ドキュメントを参照してください。

`http://java.sun.com/docs`

Java Advanced Imaging（JAI）API の詳細は、（Sun 社がメンテナンスしている）次の Web サイトを参照してください。

`http://java.sun.com/products/java-media/jai/index.html`

このマニュアルのリリース後の追加情報については、Oracle ホーム・ディレクトリに格納されているオンラインの README.txt ファイルを参照してください。オペレーティング・システムによっては、このファイルは次のディレクトリに格納されている場合もあります。

`<*Oracle_Home*>/ord/im/admin/README.txt`

詳細は、ご使用のオペレーティング・システムのインストレーション・ガイドを参照してください。

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

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

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

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

表記規則

特に指示がないかぎり、例にある各行の終わりには改行があるものとします。各入力行の最後で、[Enter] キーを押してください。

java.lang.String オブジェクトは、String と略記します。

「Boolean」は正しい名称ですが、Java コードで使用する場合は大文字と小文字を区別する必要があるため、このマニュアルでは、「boolean」と表記します。

他に、このマニュアルでは、次の表記規則を使用します。

表記規則	意味
. . .	コード例で使用される縦の省略記号は、その例に直接関係しない情報が省略されていることを示します。
...	文やコマンド内で使用される横の省略記号は、その例に直接関係しない文やコマンドが省略されていることを示します。
太字	太字は、本文中で定義されている用語を示します。
	コード例にある大カッコで囲まれた太字の数字（たとえば、**[1]**）は、その部分のコードが、後述の番号付きリストで詳細に説明されていることを示します。
イタリック体	イタリック体は、変数名を示します。
< >	山カッコは、ユーザーが指定する名前を示します。
[ ]	大カッコは、省略可能なオプション句を示します。

1

概要

Oracle *inter*Media では、ユーザーが *inter*Media オブジェクトを使用して Java アプリケーションを記述できるように、Java Classes を提供しています。

1.1 Oracle *inter*Media オブジェクト

*inter*Media オブジェクトの機能には、Oracle9*i* によって管理されるメディア・データの格納、検索、管理および操作があります。Oracle *inter*Media で格納、検索および管理をサポートしているマルチメディアには、次のものがあります。

- Oracle9*i* データベースのローカルに格納される、メディア・データを含むバイナリ・ラージ・オブジェクト（BLOB）
- オペレーティング・システム固有のファイル・システムのローカルに格納される、メディア・データを含むファイル・ベースのラージ・オブジェクト（BFILE）
- 専用メディア・サーバーに格納されるデータ

*inter*Media は、エンドユーザー向けのアプリケーションというよりは、様々なマルチメディア・アプリケーションの構成部品として機能します。その中には、オブジェクト型、およびメディア・データ管理と処理関連のメソッドが含まれます。*inter*Media オブジェクトを活用したアプリケーションの例を次に示します。

- CD 並みの音質で音楽を提供するインターネット・ミュージック・ストア
- デジタル・サウンドのリポジトリ
- 口述筆記および電話による会話のリポジトリ
- オーディオの保管および収集（音楽家向け）
- デジタル・アート・ギャラリ
- 不動産マーケティング
- ドキュメントの電子化、アーカイブおよびカタログ化
- 写真のコレクション（プロの写真家向け）
- インターネット・ビデオ・ストアおよびデジタル・ビデオクリップ・プレビュー
- ストリーミング・ビデオ配信システム用のデジタル・ビデオ・ソース
- デジタル・ビデオ・トレーニング・プログラムのライブラリ

1.2 オーディオの概念

ここでは、デジタル・オーディオの概念、および *inter*Media OrdAudio オブジェクトを使用したオーディオ・アプリケーションまたは高度な *inter*Media Audio オブジェクトの作成方法について説明します。

1.2.1 デジタル・オーディオ

*inter*Media Audio サービスは、Oracle9*i* を使用して Oracle データベースのデジタル・オーディオ・データの格納、検索および管理機能を統合します。

オーディオ制作は、オーディオ・レコーダ、マイクロフォンなどのオーディオ・ソース、デジタル・オーディオ、他の特殊オーディオ録音用デバイス、またはプログラム・アルゴリズムを使用して行うことができます。オーディオ録音用デバイスは、マイクロフォンや磁気メディアに録音されたサウンドなどのアナログ（継続した）信号を受け取り、特定のオーディオ特性（フォーマット、エンコーディング・タイプ、チャネル数、サンプリング・レート、サンプル・サイズ、圧縮タイプ、再生時間など）を持つデジタル値に変換します。

1.2.2 オーディオ・コンポーネント

デジタル・オーディオは、オーディオ・データ（デジタル・ビット）、およびオーディオ・データの情報や特性を示す属性で構成されます。オーディオ・アプリケーションは、データベース表内のオーディオ・クリップの説明、録音日、制作者およびアーティストなどのアプリケーション固有の情報を、属性またはデータベース表の列に説明文を格納することによって、オーディオ・データと関連付ける場合があります。

オーディオ・データは、そのデジタル録音の方法によって、フォーマット、エンコーディング・タイプ、圧縮タイプ、チャネル数、サンプリング率、サンプル・サイズおよび再生時間が異なります。*inter*Media Audio サービスは、どのようなデータ・フォーマットのオーディオ・データでも格納および検索できます。*inter*Media Audio サービスは、一般的で多様なオーディオ・フォーマットのオーディオ・データから自動的にメタデータを抽出します。また、*inter*Media Audio サービスは、アプリケーション属性を抽出し、Oracle *inter*Media Annotator が提供する XML 形式で、オブジェクトのコメント・フィールドに格納することもできます。サポートされるオーディオ属性は、使用可能なハードウェアの機能や、ユーザーが定義したフォーマットについての処理能力によって異なります。*inter*Media Audio サービスが属性を抽出および格納できるデータ・フォーマットのリスト、およびその他のオーディオ機能の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

*inter*Media Audio サービスは拡張可能で、追加のオーディオ・フォーマットをサポートすることができます。

数値、テキストなどの従来のコンピュータで使用するオブジェクトに比べると、デジタル・オーディオのサイズ（バイト数）は、大きくなる傾向があります。このため、オーディオ・データのバイト数を圧縮するコード体系を使用し、記憶デバイスおよびネットワークの負荷を削減します。

1.3 メディアの概念

ここでは、デジタル・メディアの概念、および *inter*Media OrdDoc オブジェクトを使用したメディア・アプリケーションまたは高度な *inter*Media オブジェクトの作成方法について説明します。

1.3.1 デジタル・メディア・データ

*inter*Media Media サービスは、Oracle9*i* を使用して Oracle データベースのメディア・ファイルの格納、検索および管理機能を統合します。

*inter*Media OrdDoc オブジェクトは、単一データベース列にオーディオ、イメージおよびビデオ・データを格納できます。オーディオ、イメージおよびビデオ・オブジェクトを別々の列に格納するかわりに、1 つの *inter*Media OrdDoc オブジェクトの列を使用して、メディアのすべての型を表現することができます。

1.3.2 メディア・コンポーネント

メディア・ファイルは、メディア・データ（デジタル・ビット）およびデータの情報や特性を示す属性で構成されます。

*inter*Media OrdDoc オブジェクトは、どのようなデータ・フォーマットのメディア・データでも格納および検索できます。OrdDoc オブジェクト型は、異なる種類のメディア・データ（オーディオ、イメージ、ビデオ、およびその他のメディア・ファイル）を、同じ列に格納する必要があるアプリケーションで使用できます。このため、OrdDoc オブジェクト型を使用して、すべての異なる種類のメディア・ファイルで共通のメタデータ索引を作成できます。この索引を使用すると、すべての異なる種類のファイルを検索できます。ただし、異なる種類のメディア・データが、異なるオブジェクト型でリレーショナル表の別々の列に格納されている場合、この検索方法は使用できません。

*inter*Media は、一般的で多様なオーディオ、イメージおよびビデオ・データ・フォーマットから、自動的にメタデータを抽出できます。また、アプリケーション属性を抽出し、XML 形式でオブジェクトのコメント属性に格納することもできます。*inter*Media が、属性を抽出および格納できるデータ・フォーマットのリスト、およびその他の機能の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』の付録 A を参照してください。*inter*Media は拡張可能で、追加のフォーマットを認識し、サポートすることができます。

1.4 イメージの概念

ここでは、デジタル・イメージの概念、および *inter*Media OrdImage オブジェクトを使用したイメージ・アプリケーションまたは高度な *inter*Media Image オブジェクトの作成方法について説明します。

1.4.1 デジタル・イメージ

*inter*Media Image サービスは、Oracle9*i* を使用して Oracle データベースのデジタル・イメージの格納、検索および管理機能を統合します。

*inter*Media Image サービスは、実世界のオブジェクトや情景をバイナリ表現で格納した、2次元の静的デジタル・ラスター・イメージをサポートします。イメージは、ドキュメントや写真のスキャナ、ビデオ・キャプチャ装置に接続されたカメラやビデオデッキなどのビデオ・ソース、その他の特殊イメージ・キャプチャ・デバイス、またはプログラム・アルゴリズムを使用して制作できます。キャプチャ・デバイスは、カメラのフィルムに映写した光などのアナログ（連続）信号を受け取り、ピクセルと呼ばれるデータの点で構成される二次元グリッド上のデジタル値に変換します。イメージのキャプチャおよび表示を行うデバイスは、アプリケーションから制御します。

1.4.2 イメージ・コンポーネント

デジタル・イメージは、イメージ・データ（デジタル・ビット）およびイメージ・データの情報や特性を示す属性で構成されます。イメージ・アプリケーションは、被写体の名前、イメージの説明、撮影日、カメラマンなどのアプリケーション固有の情報を、属性またはデータベース表内の列に説明文を格納することによって、イメージ・データに関連付ける場合があります。

イメージ・データ（ピクセル）は、イメージのキャプチャ方法によって、様々な深度（各ピクセルのビット数）を表現でき、様々な方法で構成できます。イメージ・データの構成は、**データ・フォーマット**と呼ばれます。

*inter*Media Image サービスは、どのようなデータ・フォーマットのイメージ・データでも格納および検索できます。*inter*Media Image サービスは、一般的で多様なフォーマットのイメージのプロパティを処理し、自動的に抽出することができます。*inter*Media Image サービスがメタデータを処理および抽出できるデータ・フォーマットのリストについては、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。また、一部の外部イメージ（*inter*Media Image サービスがネイティブにサポートしていないフォーマット）についてもイメージ処理を部分的にサポートしています。

デジタル・イメージに必要な記憶域は、数値やテキストなど、従来の属性データに比べて大きくなる傾向があります。様々な圧縮方法によってイメージを圧縮し、記憶デバイスやネットワークの負荷を減らすことができます。**可逆圧縮**を利用してイメージを圧縮すると、元のイメージとビット単位で同一になります。**非可逆圧縮**を利用した場合、展開後のイメージは元のイメージと同一ではありませんが、その差はごくわずかです。

イメージの**互換フォーマット**には、イメージの構成や使用方法、データおよび圧縮方法が完全に記述されているため、異なるアプリケーションでイメージを作成、変換および使用することができます。多くの場合、互換フォーマットは、ディスク・ファイル内またはディスク・ファイルとして格納されます。これは、ネットワーク上で連続的に変換されるため、1つの**プロトコル**とみなされます。デジタル・イメージの世界には、多数のアプリケーション・サブドメインがあり、その中には、デジタル・イメージを作成し、使用する多数のアプリケーションがあります。*inter*Media Image サービスは、すべてのフォーマットの格納と検索をサポートし、多数のフォーマットの処理および属性の抽出ができます。

1.4.3 コンテンツ・ベース検索の概要

コストの低いイメージ・キャプチャおよび格納のテクノロジによって、デジタル・イメージの大規模なコレクションを作成できます。ただし、データベースの規模が大きくなると、関連するイメージの検索が困難になります。イメージの検索にメタデータを使用して、次の方法でこの問題を解決します。

- タイトル、限定した単語による説明キーワード、事前に決められた分類スキームなど、手動で入力された情報、または表の設計に含まれる情報を使用する。
- イメージ・コンテンツを分類するため、自動化されたイメージ抽出機能およびオブジェクト認識機能（コンテンツ・ベース検索のための独自の機能を使用）を使用する。

*inter*Media Image サービスを使用すると、イメージを保存する表の設計において、従来のテキスト列を使用して、イメージが表現するものの意味（たとえば、特別な賞を受賞した自動車の画像、その自動車のエンジンに 6 または 8 のシリンダがあることなど）を記述する方法と、イメージに OrdImage オブジェクト型を使用して、イメージの基本的な属性に基づいたコンテンツ・ベース問合せ（たとえば、指定した自動車に色および形がどの程度一致するかなど）を実行する方法を組み合せることができます。

データベース設計者は、イメージに関連する属性をイメージとは別の列に定義するかわりに、特殊なコンポジット・データ型を作成して、*inter*Media Image サービスと適切なテキスト、数値およびデータ属性を組み合せることができます。

コンテンツ・ベース検索を使用する最大の利点は、イメージに基づく情報を、短時間で簡単に入手できることです。大規模データベースに格納されたイメージを頻繁に追加および更新する場合、問合せに必要なすべての属性を手動で入力することは非効率的ですが、コンテンツ・ベース検索を使用すると、柔軟性の高い、実用的な値が提供されます。また、コンテンツ・ベース検索では、テクスチャや形など、キーワードでは表現が困難な属性の問合せに有効です。

コンテンツ・ベース検索が有効なデータベース・アプリケーションの例（問合せが「このオブジェクトに類似したオブジェクトを検索せよ」という意味である場合）を次に示します。

- 商標、著作権およびロゴ
- 画廊および美術館
- 小売業者
- ファッションおよび布のデザイン
- 内装のデザインおよび装飾

たとえば、洋服のカタログを小売する Web ベースのインタフェースでは、従来のカテゴリ（スタイルまたは価格の範囲など）でも、またイメージ・プロパティ（色、テクスチャなど）によっても検索できます。これによって、ユーザーは、特定の価格帯にある細い縦縞のオフホワイトのフォーマル・シャツを検索します。同様に、ファッション・デザイナは、布見本、デザイン、コンセプト・スケッチおよび完成した洋服のイメージを持つデータベースを使用して、作業を簡略化できます。

1.4.4 コンテンツ・ベース検索の機能

コンテンツ・ベース検索システムは、イメージ・データに含まれる情報を処理し、視覚属性として、その内容を抽象化します。問合せ操作では、イメージ自体ではなく、この抽象化されたコンテンツのみに対して処理を行います。つまり、データベースに挿入されるすべてのイメージは分析され、フィーチャ・ベクトルまたは**シグネチャ**に、コンテンツの縮小表現が格納されます。

シグネチャに含まれる視覚属性の情報には、次のものがあります。

- 色 : イメージ全体の色の分布を表します。この分布には、それぞれの色の量は含まれますが、色の位置は含まれません。
- テクスチャ : 粒状性、滑らかさなどのイメージ内の低水準のパターンおよびテクスチャを表します。形と異なり、テクスチャはイメージ内に頻繁に存在する特性を識別します。
- 形 : 色に基づいた区分化技術によって判断される、イメージ内に存在する形を表します。形は、均一な色の領域によって特徴付けられます。
- 位置 : 形、色およびテクスチャ・コンポーネントの位置を表します。たとえば、イメージの上半分に青色が存在する、またはあるテクスチャがイメージの右下に存在するなどの情報です。

すべての視覚属性の特性データは、サイズの通常の範囲が 3000 ～ 4000 バイトであるシグネチャに格納されます。イメージ・シグネチャに基づいた索引を作成して、大きいイメージを持つデータベースでパフォーマンスを向上させることができます。

比較イメージと一致させることによって、データベースのイメージを検索します。**比較イメージ**は、現行のデータベースの内部または外部、見本、アルゴリズムによって生成されたイメージなどのすべてのイメージです。

マッチング処理では、比較イメージに対して生成されたシグネチャが必要です。イメージ・マッチングは、シグネチャの比較を基にします。**スコア**は、各属性の値の合計を重み付けしたもので、イメージを比較する場合に一致度の判断に使用され、差異が少ないほどより一致していることになります。**重み**は正の実数で、この値は、与えられた属性に対する 2 つのイメージ間のマッチング処理で一致または不一致の精度を表します。視覚属性との違いを重み付けした合計が**しきい値**以下の場合、イメージは一致し、しきい値より大きい場合は一致しません。

Oracle *inter*Media Java Classes には、OrdImageSignature オブジェクト用の API があり、Java オブジェクトにおけるイメージ・シグネチャ情報の格納および比較に使用されます。

視覚属性、重み、一致度の計算、しきい値および索引付けの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

1.4.5 入力ストリームおよび出力ストリーム

Java の拡張機能として、Sun 社から JAI API が提供されています。JAI を使用すると、Java アプリケーションに、高度なイメージ処理操作を組み込むことができます。Oracle *inter*Media を使用すると、JAI アプリケーションから、データベースに格納されたイメージの読込みおよび書込みができます。

Oracle *inter*Media Java Classes が提供する 3 種類のストリーム API によって、BLOB および BFILE からデータを読み込み、JAI アプリケーションの BLOB に書き込むことができます。これらのストリーム・オブジェクトは、Sun 社が提供する入力ストリームおよび出力ストリームを置き換えるものではなく、JAI が使用可能な OrdImage オブジェクトに BLOB および BFILE で格納されたイメージ・データに対する、パフォーマンスを低下させないインタフェースを提供します。

これらのストリーム・オブジェクトの詳細は、第 7 章を参照してください。

1.5 ビデオの概念

ここでは、デジタル・ビデオの概念、および *inter*Media OrdVideo オブジェクトを使用したビデオ・アプリケーションまたは高度な *inter*Media Video オブジェクトの作成方法について説明します。

1.5.1 デジタル・ビデオ

*inter*Media Video サービスは、Oracle9*i* を使用して Oracle データベースのデジタル・ビデオ・データの格納、検索および管理機能を統合します。

ビデオは、ビデオ・レコーダ、ビデオ・カメラ、デジタル・アニメーション・ビデオ、その他の特殊ビデオ録画用デバイス、またはプログラム・アルゴリズムを使用して作成します。ビデオ録画デバイスは、ビデオ・カメラから取得したビデオや磁気メディアに録画されたビデオなどのアナログ（連続した）信号を受け取り、ビデオ・フォーマット、エンコーディング・タイプ、フレーム・レート、フレーム・サイズ（幅および高さ）、フレームの解像度、ビデオの長さ、圧縮タイプ、色数、ビット・レートなどの特定のビデオ特性を持つデジタル値に変換します。

1.5.2 ビデオ・コンポーネント

デジタル・ビデオは、ビデオ・データ（デジタル・ビット）、およびビデオ・データの情報や特性を示す属性で構成されます。ビデオ・アプリケーションは、ビデオ・トレーニング・テープの説明、録画日、インストラクタ名、プロデューサ名などのアプリケーション固有情報を、属性（データベース表の列）内に説明文を格納することによって、ビデオ・データと関連付ける場合があります。

ビデオ・データは、デジタル録画の方式によって、データ・フォーマット、圧縮タイプ、フレーム・レート、フレーム・サイズ、フレームの解像度、再生時間、色数およびビット・レートが異なります。*inter*Media Video サービスは、どのようなデータ・フォーマットのビデオ・データでも格納および検索できます。また、一般的で多様なビデオ・フォーマットのビデオ・データから自動的にメタデータを抽出します。*inter*Media Video サービスは、アプリケーション属性を抽出し、XML 形式で、オブジェクトのコメント・フィールドに格納することもできます。サポートされるビデオ属性は、使用可能なハードウェアの機能や、ユーザーが定義したフォーマットについての処理能力によって異なります。*inter*Media Video サービスを使用して属性を抽出および格納できるデータ・フォーマットのリストおよびその他のビデオ機能の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

*inter*Media Video サービスは拡張可能で、追加のビデオ・フォーマットを認識およびサポートすることができます。

数値やテキストなどの従来のコンピュータで使用するオブジェクトに比べて、デジタル・ビデオのサイズ（バイト数）は大きくなる傾向があります。このため、いくつかのエンコーディング方法でビデオ・データを圧縮し、記憶デバイスやネットワークの負荷を減らすことができます。

1.6 Java アプリケーションのサポート

Oracle *inter*Media を使用すると、データベース表にマルチメディア情報を格納し、その表からデータを検索して操作することができます。また、独自の Java アプリケーションを記述して、Oracle データベースに格納されたメディア・データを使用、操作および修正できます。

*inter*Media Java Classes を使用するアプリケーションは、結果セットからオブジェクトを検索し、オブジェクトのコンテンツを操作することができます。

1.7 *inter*Media オブジェクトにアクセスする Java アプリケーションの記述

Oracle データベース表にある *inter*Media オブジェクトにアクセスするアプリケーションは、次のように記述します。

1. Java アプリケーションから Oracle データベースへの JDBC 接続を確立します。

   getConnection( ) メソッドをコールし、OracleConnection オブジェクトを取得します。データベースに接続するメソッドの例については、例 2-3 を参照してください。

2. アプリケーションが *inter*Media オブジェクトを修正する場合は、次の操作を行います。

   a. setAutoCommit( ) メソッドをコールし、自動コミット・モードを使用禁止にします。

   b. データベース表で、SELECT... FOR UPDATE 文を実行します。

   自動コミット・モードを使用禁止にするメソッドの例については、例 2-3 を参照してください。

   アプリケーション内に、OracleStatement オブジェクトまたは OraclePreparedStatement オブジェクトを作成します。executeQuery( ) メソッドをコールして SELECT... FOR UPDATE 文を実行して OracleResultSet オブジェクトを戻し、その結果セットから行をフェッチします。例 2-4 の手順 1 および 2 を参照してください。

3. アプリケーションが *inter*Media オブジェクトを修正しない場合は、データベース表で SELECT 文を実行します。

   例 2-6 を参照してください。

4. 結果セットから *inter*Media オブジェクトを検索します。

   *inter*Media Java クラスの 1 つのインスタンスとして、結果セットから *inter*Media オブジェクトを検索する方法については、例 2-4 の手順 5 を参照してください。

5. Java アプリケーション・オブジェクトに対し操作を実行します。実行可能な操作の例については、第 2 章を参照してください。

   結果セットから *inter*Media Java オブジェクトを取得すると、アプリケーションは、新しいデータをオブジェクトにロードできます。また、アプリケーションは、オブジェクトの既存のデータを検索または操作できます。新しいデータをオブジェクトにロードする方法については、例 2-4 の手順 6 を参照してください。

6. アプリケーションが *inter*Media オブジェクトを変更した場合、操作結果を反映するためにデータベース・オブジェクトを更新し、変更をコミットします。

   前述の手順でアプリケーションがオブジェクトを変更した場合、データベース・オブジェクトを更新する SQL 文を含む OraclePreparedStatement オブジェクトを作成し、文を実行します。例 2-4 の手順 9 を参照してください。

   commit( ) メソッドをコールし、トランザクションをコミットします。例 2-2 の手順 4 を参照してください。

7. データベース表への接続をクローズします。

   例 2-2 の手順 5 を参照してください。

   JDBC の使用方法の詳細は、『Oracle9*i* JDBC 開発者ガイドおよびリファレンス』を参照してください。

1.8 *inter*Media の以前のリリースとの互換性

今後リリースする *inter*Media では、新規のオブジェクト属性を追加して、*inter*Media オブジェクト型が改善される可能性があります。サーバーをアップグレードしてオブジェクト型を変更した後も、リリース 9.0.1 の *inter*Media オブジェクト型（OrdAudio、OrdDoc、OrdImage および OrdVideo）とクライアント側アプリケーションとの互換性を保つことはできます。ただし、アプリケーションの開始時に、互換性初期化ファンクションをコールする必要があります。

Java で記述された *inter*Media Java Classes を使用するクライアント側アプリケーションは、Oracle データベースに接続後、OrdMediaUtil.imCompatibilityInit( ) メソッドをコールする必要があります。

```
public static void imCompatibilityInit(OracleConnection con)
    throws Exception
```

この Java ファンクションでは、引数に OracleConnection を指定します。

imCompatibilityInit( ) メソッドの例については、例 2-2 の手順 2 を参照してください。

2

Java クラスを使用したプログラム例

この章では、*inter*Media Java Classes を使用したユーザー定義クラスの例について説明します。データベース・サーバーでのスキーマの設定方法を示す SQL スクリプトのサンプルも含まれています。

このサンプル・コードは、*inter*Media Java Classes のインストールで提供される AudioExample.java、DocumentExample.java、ImageExample.java または VideoExample.java のコードと一致しない場合があります。システム上で例を実行する場合は、*inter*Media Java Classes のインストールによって提供されたファイルを使用してください。この章で示されているコードは、コンパイルまたは実行しないでください。

注意： この章では、Java および SQL コードの例が示されています。コード例に大カッコで囲まれた太字の数字が記述されている場合があります。これらは、そのコードの詳細が例の直後の番号付きリストで説明されていることを示しています。

2.1 OrdAudio の例

オーディオの例（AudioExample.sql および AudioExample.java）では、SQL、JDBC および *inter*Media Java Classes API を使用して次の操作を行う、ユーザー定義メソッドについて説明します。

- テスト・コンテンツを含むデータベース・サーバー表を作成します。
- ローカル・ファイルからアプリケーションおよびデータベース内の OrdAudio オブジェクトにデータをロードします。
- ローカル・ストリームからアプリケーションおよびデータベース内の OrdAudio オブジェクトにデータをロードします。
- ローカル・バイト配列からアプリケーションおよびデータベース内の OrdAudio オブジェクトにデータをロードします。
- アプリケーション内の OrdAudio オブジェクトからプロパティを抽出し、出力します。
- データベース・メソッドへの違反コールを使用して、エラー処理を示します。

2.1.1 AudioExample.sql

例 2-1 は、AudioExample.sql サンプル・ファイルの内容をすべて示しています。

例 2-1 AudioExample.sql の内容

```
set echo on

-- PLEASE change system password
connect system/manager
drop user  AUDIOUSER cascade;

[1] create user AUDIOUSER identified by AUDIOUSER;
grant connect,resource to AUDIOUSER identified by AUDIOUSER;

[2] connect AUDIOUSER/AUDIOUSER

[3] CREATE TABLE TAUD(n NUMBER, aud ORDSYS.ORDAUDIO);

--
-- Note - the OrdAudio.init method was added in interMedia 8.1.7.
-- If you are running against an older release of interMedia and the
-- Oracle database, you will have to modify the following INSERT statements
-- to use the OrdAudio default constructor.
--
[4] INSERT INTO TAUD VALUES(1, ORDSYS.ORDAudio.init( ));
INSERT INTO TAUD VALUES(2, ORDSYS.ORDAudio.init( ));
INSERT INTO TAUD VALUES(3, ORDSYS.ORDAudio.init( ));
commit;
```

AudioExample.sql の SQL 文は、次の操作を行います。

1. ユーザー AUDIOUSER を作成し、必要な権限を付与します。
2. AUDIOUSER でデータベース・サーバーに接続します。
3. 表 TAUD を作成します。TAUD には、2 つの列（数値、OrdAudio オブジェクト）があります。
4. 表に、空の OrdAudio オブジェクトを含む行を 3 つ追加します。

init メソッドの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

2.1.2 AudioExample.java

2.1.2.1 項～ 2.1.2.8 項では、AudioExample.java サンプル・ファイルに含まれるメソッドを示します。

2.1.2.1 main( ) メソッド

例 2-2 に、main( ) メソッドを示します。

例 2-2 main( ) メソッド（Audio）

```
public static void main (String args[ ]){
     byte[ ] ctx = new byte[4000];
     OracleConnection con = null;
     try {
          AudioExample tk = new AudioExample( );
          [1] con = tk.connect( );
          //Include the following line only if you are running
          //an Oracle 8.1.7 database or later.
          //If you are running a database server prior to 8.1.7,
          //the call will fail.
          [2] OrdMediaUtil.imCompatibilityInit(con);
          [3] tk.loadDataFromFile(con);
          tk.extractProperties(con);
          tk.printProperties(con);
          tk.otherMethods(con);
          tk.loadDataFromStream(con);
          tk.loadDataFromByteArray(con);
          [4] con.commit( );
          [5] con.close( );
          System.out.println("Done.");
     }
```

```
     [6] catch (Exception e) {
          try {
               System.out.println("Exception : " + e);
               con.close( );
          }
          catch(Exception ex) {
               System.out.println("Close Connection Exception : " + ex);
          }
     }
}
```

main( ) メソッドは、次の操作を行います。

1. connect( ) メソッドを使用して、データベース表への接続を確立します。
2. ご使用のアプリケーションと新しいリリースの Oracle データベースの互換性を保証します。詳細は、1.8 項を参照してください。
3. データベース・サーバーおよびローカル・マシン上のオブジェクトを操作する、複数のメソッド（AudioExample.java で定義済）をコールします。
4. データベース表への変更をすべてコミットします。
5. データベースへの接続をクローズします。
6. コードによって発生したエラーまたは例外を処理します。

2.1.2.2 項～ 2.1.2.8 項では、main( ) メソッドからコールされたメソッドについて、AudioExample.java に記述された順序ではなく、それがコールされた順序で説明します。

2.1.2.2 connect( ) メソッド

例 2-3 に、ユーザー定義メソッド connect( ) を示します。このメソッドでは、アプリケーションからデータベースへの接続を確立します。

例 2-3 connect( ) メソッド（Audio）

```
public OracleConnection connect( ) throws Exception {
     String connectString;
     [1] Class.forName ("oracle.jdbc.driver.OracleDriver");
     [2] connectString = "jdbc:oracle:oci8:@";
     [3] OracleConnection con = (OracleConnection)DriverManager.getConnection
          (connectString,"AUDIOUSER","AUDIOUSER");
     [4] con.setAutoCommit(false);
     return con;
}
```

connect( ) メソッドは、次の操作を行います。

1. Oracle データベースでは、JDK 準拠の Java Virtual Machine（JVM）を使用するため、JDBC ドライバを直接ロードします。
2. 接続するデータベースの URL を含む文字列を定義します。ご使用のデータベースにあわせて、この文字列を変更してください。
3. connectString に指定された URL、ユーザー名 AUDIOUSER およびパスワード AUDIOUSER を使用してデータベースへの接続を設定します。ユーザー名およびパスワードは、AudioExample.sql で作成済です。
4. 自動コミット・モードを使用禁止にします。これは、commit( ) メソッドまたは rollback( ) メソッドを使用して、それぞれ手動でコミットまたはロールバックする必要があることを意味します。

2.1.2.3 loadDataFromFile( ) メソッド

例 2-4 に、ユーザー定義メソッド loadDataFromFile( ) を示します。このメソッドでは、*inter*Media の loadDataFromFile( ) メソッドを使用してメディア・データをアプリケーション・オブジェクトに移入します。

例 2-4 loadDataFromFile( ) メソッド（Audio）

```
public void loadDataFromFile(OracleConnection con) {
     try {
          [1] Statement s = con.createStatement( );
          [2] OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TAUD where n = 1 for update ");
          int index = 0;
          [3] while(rs.next( )){
               [4] index = rs.getInt(1);
               [5] OrdAudio audObj = (OrdAudio) rs.getCustomDatum
                    (2, OrdAudio.getFactory( ));
               [6] audObj.loadDataFromFile("testaud.dat");
               [7] audObj.getDataInFile("output1.dat");
               System.out.println("***********AFTER getDataInFile ");
               [8] System.out.println(" getContentLength output : " +
                    audObj.getContentLength( ));
               [9] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update taud set aud = ? where
                    n = " + index);
               stmt1.setCustomDatum(1,audObj);
               stmt1.execute( );
               stmt1.close( ) ;
```

```
            }
            System.out.println("loading successful");
        }
        [10] catch(Exception e) {
            System.out.println("exception raised " + e);
            System.out.println("loading unsuccessful");
        }
}
```

loadDataFromFile( ) メソッドは、次の操作を行います。

1. OracleStatement オブジェクトを作成します。
2. 所定の SQL 問合せを実行し、その結果をローカル OracleResultSet オブジェクトに格納します。この場合、SQL 問合せは、n=1 の条件を満たすデータベース行のデータを選択します。
3. OracleResultSet に処理されていない結果がある場合、while ループ内の操作を行います。この場合、OracleResultSet に含まれる行は 1 行のみのため、ループ内の操作は一度のみ実行されます。
4. 変数 index に、OracleResultSet の 1 行目の 1 列目にある整数値（この場合、1）を設定します。
5. ローカル OrdAudio オブジェクトとして audObj を作成します。OracleResultSet の現行の行の 2 列目にある OrdAudio オブジェクトのコンテンツを audObj に移入します。
6. OrdAudio の loadDataFromFile( ) メソッドを使用して、testaud.dat 内のメディア・データをデータベース内の OrdAudio オブジェクトおよび audObj にロードします。これによって、ローカル・フィールドが audObj に設定されますが、データベース・オブジェクトには設定されません。
7. getDataInFile( ) メソッドを使用して、audObj からメディア・データを取得し、ローカル・システム上のファイル output1.dat にロードします。
8. ロードが成功したことを確認するため、audObj のコンテンツ長を取得し、画面に出力します。
9. データベース内の OrdAudio オブジェクトを、audObj のコンテンツで更新する SQL 文を作成および実行します。
10. コードによって発生したエラーまたは例外を処理します。

2.1.2.4 extractProperties( ) メソッド

例 2-5 に、ユーザー定義メソッド extractProperties( ) を示します。このメソッドでは、アプリケーション・オブジェクトにプロパティを設定します。

例 2-5 extractProperties( ) メソッド（Audio）

```
public void extractProperties(OracleConnection con){
     byte[ ] ctx[ ] = new byte [4000][1];
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TAUD where n = 1 for update");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdAudio audObj = (OrdAudio) rs.getCustomDatum
                    (2, OrdAudio.getFactory( ));
               [2] audObj.setProperties(ctx);
               System.out.println("set Properties called");
               [3] if(audObj.checkProperties(ctx)){
                    System.out.println("checkProperties called");
                    System.out.println("setProperties successful");
                    System.out.println("checkProperties successful");
                    System.out.println("extraction  successful");
               }
               else{
                    System.out.println("checkProperties called");
                    System.out.println("extraction not successful");
                    System.out.println("checkProperties successful");
               }
               [4] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update taud set aud = ?
                    where n = " + index);
               stmt1.setCustomDatum(1,audObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          rs.close( );
          s.close( );
     }
     [5] catch(Exception e) {
          System.out.println("exception raised " + e);
          System.out.println("extract properties unsuccessful");
     }
}
```

extractProperties( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdAudio オブジェクトとして audObj を作成し、例 2-4 の手順 1 ～ 5 と同様の処理で audObj にメディア・データを移入します。このメソッドでは、データベース表の n=1 の条件を満たす行の 2 列目のコンテンツを操作しています。
2. setProperties( ) をコールして、メディア・データからプロパティ値を抽出し、その値をアプリケーション内の OrdAudio オブジェクトに設定します。抽出および設定されたプロパティ値については、第 3 章の「setProperties(byte[ ][ ])」を参照してください。
3. checkProperties( ) をコールして、アプリケーション・オブジェクトのプロパティ値とメディア・データの値を比較します。値がすべて同じである場合、checkProperties( ) は true を戻し、画面に適切なメッセージを表示します。値が異なる場合は false を戻し、画面に適切なメッセージを表示します。
4. データベース内の OrdAudio オブジェクトを、audObj のコンテンツ（setProperties( ) によって抽出されたプロパティを含む）で更新する SQL 文を作成および実行します。
5. コードによって発生したエラーまたは例外を処理します。

2.1.2.5 printProperties( ) メソッド

例 2-6 に、ユーザー定義メソッド printProperties( ) を示します。このメソッドでは、アプリケーション・オブジェクトの属性を画面に出力します。

例 2-6 printProperties( ) メソッド（Audio）

```
public void printProperties(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)
               s.executeQuery("select * from TAUD where n = 1 ");
          int index = 0;
          while(rs.next( )) {
               index = rs.getInt(1);
               OrdAudio audObj = (OrdAudio) rs.getCustomDatum
                    (2, OrdAudio.getFactory( ));
               [2] System.out.println("format: " + audObj.getFormat( ));
               System.out.println("mimeType: " + audObj.getMimeType( ));
               System.out.println("encoding: " + audObj.getEncoding( ));
               System.out.println("numberOfChannels: " +
                    audObj.getNumberOfChannels( ));
               System.out.println("samplingRate: " +
                    audObj.getSamplingRate( ));
               System.out.println("sampleSize: " + audObj.getSampleSize( ));
               System.out.println("compressionType : " +
                    audObj.getCompressionType( ));
               System.out.println("audioDuration: " +
```

```
                    audObj.getAudioDuration( ));
               System.out.println("description: " + audObj.getDescription( ));
          }
     }
     [3] catch(Exception e){
          System.out.println("exception raised " + e);
          System.out.println("print proerties unsuccessful");
     }
}
```

printProperties( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdAudio オブジェクトとして audObj を作成し、例 2-4 の手順 1 ～ 5 と同様の処理で audObj にメディア・データを移入します。このメソッドでは、データベース表の n=1 の条件を満たす行の 2 列目のコンテンツを操作しています。
2. audObj のプロパティ値を取得し、その値を画面に出力します。
3. コードによって発生したエラーまたは例外を処理します。

2.1.2.6 otherMethods( ) メソッド

例 2-7 に、ユーザー定義メソッド otherMethods( ) を示します。このメソッドでは、processSourceCommand( ) メソッドを使用します。

例 2-7 otherMethods( ) メソッド（Audio）

```
public void otherMethods(OracleConnection con){
     byte[ ] ctx[ ] = {new byte[4000]};
     byte[ ] res[ ] = {new byte[20]};
     [1] int suc = 1;
     try {
          [2] Statement s1 = con.createStatement( );
          OracleResultSet rs1 = (OracleResultSet) s1.executeQuery
               ("select * from TAUD where n = 1 for update ");
          int index1 = 0;
          while(rs1.next( )) {
               index1 = rs1.getInt(1);
               OrdAudio audObj = (OrdAudio) rs1.getCustomDatum
                    (2, OrdAudio.getFactory( ));
               [3] try {
                    byte[ ] pSRes = audObj.processSourceCommand(ctx,
                         "", "", res);
                    suc = 0;
               }
               [4] catch (Exception e) {
                    System.out.println("Expected Exception raised in
```

```
                    processSourceCommand(...)" );
               }
               [5] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update taud set aud = ? where
                    n = " + index1);
               stmt1.setCustomDatum(1,audObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          rs1.close( );
          s1.close( );
     }
     [6] catch(Exception e){
          System.out.println("Exception raised " );
     }
     [7] if(suc == 1)
          System.out.println("other methods successful");
     else
          System.out.println("other methods unsuccessful");
}
```

otherMethods( ) メソッドは、次の操作を行います。

1. メソッドの成功または失敗を示すために使用する整数を作成し、これを 1（成功）に初期化します。
2. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdAudio オブジェクトとして audObj を作成し、例 2-4 の手順 1 ～ 5 と同様の処理で audObj にメディア・データを移入します。このメソッドでは、データベース表の n=1 の条件を満たす行の 2 列目のコンテンツを操作しています。
3. サーバー側でコールされるコマンドを指定せずに processSourceCommand( ) をコールします。これによって例外が発生します。これは、processSourceCommand( ) コールの次のコードは実行されず、catch ループのコードが実行されることを意味します。例外が発生しない場合、メソッドは失敗し、成功インジケータが 0（失敗）に設定されます。
4. 手順 3 で発生した例外を出力します。
5. データベース内の OrdAudio オブジェクトを、audObj のコンテンツで更新する SQL 文を作成および実行します。
6. コードによって発生した、予期しないエラーまたは例外を処理します。
7. メソッドの結果に基づいて、適切なメッセージを画面に出力します。

2.1.2.7 loadDataFromStream( ) メソッド

例 2-8 に、ユーザー定義メソッド loadDataFromStream( ) を示します。このメソッドでは、*inter*Media の loadDataFromInputStream( ) メソッドを使用して、メディア・データをアプリケーション・オブジェクトにロードします。

例 2-8 loadDataFromStream( ) メソッド（Audio）

```
public void loadDataFromStream(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TAUD where n = 2 for update ");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdAudio audObj = (OrdAudio) rs.getCustomDatum
                    (2, OrdAudio.getFactory( ));
               [2] FileInputStream fStream = new
                    FileInputStream("testaud.dat");
               [3] audObj.loadDataFromInputStream(fStream);
               [4] audObj.getDataInFile("output2.dat");
               [5] fStream.close( );
               System.out.println("************AFTER getDataInFile ");
               [6] System.out.println(" getContentLength output : " +
                    audObj.getContentLength( ));
               [7] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update taud set aud = ? where
                    n = " + index);
               stmt1.setCustomDatum(1,audObj);
               stmt1.execute( );
               stmt1.close( );
          }
          System.out.println("load data from stream successful");
     }
     [8] catch(Exception e) {
          System.out.println("exception raised " + e);
          System.out.println("load data from stream unsuccessful");
     }
}
```

loadDataFromStream( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdAudio オブジェクトとして audObj を作成し、例 2-4 の手順 1 ～ 5 と同様の処理で audObj にメディア・データを移入します。このメソッドでは、データベース表の n=2 の条件を満たす行の 2 列目のコンテンツを操作しています。
2. 新規の FileInputStream オブジェクトを作成します。この入力ストリームには、ローカル・ファイル testaud.dat のコンテンツが含まれています。
3. loadDataFromInputStream( ) メソッドを使用して、入力ストリームにあるメディア・データを、データベース内の OrdAudio オブジェクトおよび audObj にロードします。これによって、ローカル・フィールドが audObj に設定されますが、データベース・オブジェクトには設定されません。
4. getDataInFile( ) メソッドを使用して、アプリケーション内の OrdAudio オブジェクトからメディア・データを取得し、ローカル・システム上のファイル output2.dat にロードします。
5. ローカル入力ストリームをクローズします。
6. ロードが成功したことを確認するため、audObj のコンテンツ長を取得し、画面に出力します。
7. データベース内の OrdAudio オブジェクトを、audObj のコンテンツで更新する SQL 文を作成および実行します。この更新によって、アプリケーション・オブジェクトに一致するようにデータベース・オブジェクトの属性が設定されます。
8. コードによって発生したエラーまたは例外を処理します。

2.1.2.8 loadDataFromByteArray( ) メソッド

例 2-9 に、ユーザー定義メソッド loadDataFromByteArray( ) を示します。このメソッドでは、*inter*Media の loadDataFromByteArray( ) メソッドを使用して、メディア・データをアプリケーション・オブジェクトにロードします。

例 2-9 loadDataFromByteArray( ) メソッド（Audio）

```
public void loadDataFromByteArray(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TAUD where n = 3 for update ");
          int index = 0;
          while(rs.next( )) {
               index = rs.getInt(1);
               OrdAudio audObj = (OrdAudio) rs.getCustomDatum
                    (2, OrdAudio.getFactory( ));
               [2] File ff = new File("testaud.dat");
               int fileLength = (int) ff.length( );
```

```
            byte[ ] data = new byte[fileLength];
            [3] FileInputStream fStream = new
                 FileInputStream("testaud.dat");
            [4] fStream.read(data,0,fileLength);
            [5] audObj.loadDataFromByteArray(data);
            [6] fStream.close( );
            [7] audObj.getDataInFile("output3.dat");
            [8] byte[ ] resArr = audObj.getDataInByteArray( );
            [9] System.out.println("byte array length : " +
                 resArr.length);
            [10] FileOutputStream outStream = new FileOutputStream
                 ("output4.dat");
            [11] outStream.write(resArr);
            [12] outStream.close( );
            [13] InputStream inpStream = audObj.getDataInStream( );
            int length = 32768;
            byte[ ] tempBuffer = new byte[32768];
            [14] int numRead = inpStream.read(tempBuffer,0,length);
            try {
                 [15] outStream = new FileOutputStream("output5.dat");
                 [16] while (numRead != -1) {
                      [17] if (numRead < 32768) {
                           length = numRead;
                           outStream.write(tempBuffer,0,length);
                           break;
                      }
                      [18] else
                           outStream.write(tempBuffer,0,length);
                      [19] numRead = inpStream.read(tempBuffer,0,length);
                 }
            }
            [20] finally {
                 outStream.close( );
                 inpStream.close( );
            }
            System.out.println("************AFTER getDataInFile ");
            [21] System.out.println("getContentLength output : " +
                 audObj.getContentLength( ));
            [22] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                 con.prepareCall("update taud set aud = ? where
                 n = " + index);
            stmt1.setCustomDatum(1,audObj);
            stmt1.execute( );
            stmt1.close( ) ;
       }
}
```

```
      [23] catch(Exception e) {
           System.out.println("exception raised " + e);
           System.out.println("load data from byte array unsuccessful");
      }
}
```

loadDataFromByteArray( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdAudio オブジェクトとして audObj を作成し、例 2-4 の手順 1 ～ 5 と同様の処理で audObj にメディア・データを移入します。このメソッドでは、データベース表の n=3 の条件を満たす行の 2 列目のコンテンツを操作しています。
2. ローカル・ファイル testaud.dat のサイズ（バイト単位）を判断し、同じサイズのバイト配列を作成します。
3. 新規の FileInputStream オブジェクトを作成します。この入力ストリームには、testaud.dat のコンテンツが含まれています。
4. 入力ストリームのコンテンツをバイト配列に読み込みます。
5. loadDataFromByteArray( ) メソッドを使用して、バイト配列にあるメディア・データをデータベース内の OrdAudio オブジェクトおよび audObj にロードします。これによって、ローカル・フィールドが audObj に設定されますが、データベース・オブジェクトには設定されません。
6. 入力ストリームをクローズします。
7. getDataInFile( ) メソッドを使用して、アプリケーション内の OrdAudio オブジェクトからメディア・データを取得し、ローカル・システム上のファイル output3.dat にロードします。
8. getDataInByteArray( ) メソッドを使用して、アプリケーション内の OrdAudio オブジェクトからメディア・データを取得し、バイト配列 resArr にロードします。
9. ロードが成功したことを確認するため、resArr の長さを取得し、画面に出力します。
10. 新規の FileOutputStream オブジェクト outStream を作成します。この出力ストリームは、ローカル・ファイル output4.dat にデータを書き込みます。
11. resArr のコンテンツを output4.dat に書き込みます。
12. 出力ストリームをクローズします。
13. 新規の入力ストリーム inpStream を作成します。getDataInStream( ) メソッドを使用して、アプリケーション内の OrdAudio オブジェクトからメディア・データを取得し、inpStream に格納します。
14. inpStream の先頭（オフセット 0）から 32768 バイトをバイト配列 tempBuffer に読み込みます。整数 numRead には、読込みバイトの合計数、または入力ストリームが最後まで到達した場合は -1 が設定されます。ロードが成功した場合、numRead は 32768 となります。

15. OutStream を再オープンします。この場合、ローカル・ファイル output5.dat にデータを書き込みます。
16. numRead が -1 でない場合は、while ループ内の操作を実行します。プログラムは、このループを実行します。
17. numRead が 32768 未満（データが最後まで読み込まれた）の場合は、if ループを実行します。if ループは、tempBuffer に読み込まれたバイト数を outStream に書き込み、ループから出ます。
18. numRead が 32768 の場合、outStream に 32768 バイトを書き込みます。
19. 入力ストリームからさらにデータをバイト配列に読み込みます。データが最後まで正常に読み込まれた場合、numRead は -1 に設定され、プログラムはループを終了します。入力ストリームに読み込まれていないデータがある場合は、バイト配列に読み込まれ、手順 17、18 が繰り返されます。
20. while ループの終了後、入力ストリームおよび出力ストリームをクローズします。
21. ロードが成功したことを確認するため、audObj のコンテンツ長を取得し、画面に出力します。
22. データベース内の OrdAudio オブジェクトを、audObj のコンテンツで更新する SQL 文を作成および実行します。この更新によって、アプリケーション・オブジェクトに一致するようにデータベース・オブジェクトの属性が設定されます。
23. コードによって発生したエラーまたは例外を処理します。

2.2 OrdDoc の例

OrdDoc の例（DocumentExample.sql および DocumentExample.java）では、SQL、JDBC および *inter*Media Java Classes API を使用して次の操作を行う、ユーザー定義メソッドについて説明します。

- テスト・コンテンツを含むデータベース・サーバー表を作成します。
- ローカル・ファイルからアプリケーションおよびデータベース内の OrdDoc オブジェクトにデータをロードします。
- ローカル・ストリームからアプリケーションおよびデータベース内の OrdDoc オブジェクトにデータをロードします。
- ローカル・バイト配列からアプリケーションおよびデータベース内の OrdDoc オブジェクトにデータをロードします。
- アプリケーション内の OrdDoc オブジェクトからプロパティを抽出し、出力します。
- データベース・メソッドへの違反コールを使用して、エラー処理を示します。

2.2.1 DocumentExample.sql

例 2-10 は、DocumentExample.sql サンプル・ファイルの内容をすべて示しています。

例 2-10 DocumentExample.sql の内容

```
set echo on

--PLEASE change system password
connect system/manager
drop user DOCUSER cascade;

[1] create user DOCUSER identified by DOCUSER ;
grant connect,resource to DOCUSER identified by DOCUSER;

[2] connect DOCUSER/DOCUSER

[3] CREATE TABLE TDOC(n NUMBER, doc ORDSYS.ORDDOC);

[4] INSERT INTO TDOC VALUES(1, ORDSYS.ORDDoc.init( ));
INSERT INTO TDOC VALUES(2, ORDSYS.ORDDoc.init( ));
INSERT INTO TDOC VALUES(3, ORDSYS.ORDDoc.init( ));

commit;
```

DocumentExample.sql の SQL 文は、次の操作を行います。

1. ユーザー DOCUSER を作成し、必要な権限を付与します。
2. DOCUSER でデータベース・サーバーに接続します。
3. 表 TDOC を作成します。TDOC には、2 つの列（数値、OrdDoc オブジェクト）があります。
4. 表に、空の OrdDoc オブジェクトを含む行を 3 つ追加します。

2.2.2 DocumentExample.java

2.2.2.1 項～ 2.2.2.8 項では、DocumentExample.java サンプル・ファイルで定義されているメソッドを示します。

2.2.2.1 main( ) メソッド

例 2-11 に、main( ) メソッドの内容を示します。

例 2-11 main( ) メソッド（Doc）

```
public static void main (String args[ ]){
     byte[ ] ctx = new byte[4000];
     OracleConnection con = null;
     try {
          DocumentExample tk = new DocumentExample( );
          [1] con = tk.connect( );
          [2] OrdMediaUtil.imCompatibilityInit(con);
          [3] tk.loadDataFromFile(con);
          tk.extractProperties(con);
          tk.printProperties(con);
          tk.loadDataFromStream(con);
          tk.otherMethods(con);
          tk.loadDataFromByteArray(con);
          [4] con.commit( );
          [5] con.close( );
          System.out.println("Done.");
     }
     [6] catch (Exception e) {
          try {
               System.out.println("Exception : " + e);
               con.close( );
          }
          catch(Exception ex) {
          System.out.println("Close Connection Exception : " + ex);
          }
     }
}
```

main( ) メソッドは、次の操作を行います。

1. connect( ) メソッドを使用して、データベース表への接続を確立します。
2. ご使用のアプリケーションと新しいリリースの Oracle データベースの互換性を保証します。詳細は、1.8 項を参照してください。
3. データベース・サーバーおよびローカル・マシン上のオブジェクトを操作する、複数のメソッド（DocumentExample.java で定義済）をコールします。

4. データベース表への変更をすべてコミットします。
5. データベースへの接続をクローズします。
6. コードによって発生したエラーまたは例外を処理します。

2.2.2.2 項～ 2.2.2.8 項では、main( ) メソッドからコールされたメソッドについて、DocumentExample.java に記述された順序ではなく、それがコールされた順序で説明します。

2.2.2.2 connect( ) メソッド

例 2-12 に、ユーザー定義メソッド connect( ) を示します。このメソッドでは、アプリケーションからデータベースへの接続を確立します。

例 2-12 connect( ) メソッド（Doc）

```
public OracleConnection connect( ) throws Exception{
     String connectString;
     [1] Class.forName ("oracle.jdbc.driver.OracleDriver");
     [2] connectString = "jdbc:oracle:oci8:@";
     [3] OracleConnection con = (OracleConnection)
          DriverManager.getConnection(connectString,"DOCUSER","DOCUSER");
     [4] con.setAutoCommit(false);
     return con;
}
```

connect( ) メソッドは、次の操作を行います。

1. Oracle データベースでは、JDK 準拠の JVM を使用するため、JDBC ドライバを直接ロードします。
2. 接続するデータベースの URL を含む文字列を定義します。ご使用のデータベースにあわせて、この文字列を変更してください。
3. connectString に指定された URL、ユーザー名 DOCUSER およびパスワード DOCUSER を使用してデータベースへの接続を設定します。ユーザー名およびパスワードは、DocumentExample.sql で作成済です。
4. 自動コミット・モードを使用禁止にします。これは、commit( ) メソッドまたは rollback( ) メソッドを使用して、それぞれ手動でコミットまたはロールバックする必要があることを意味します。

2.2.2.3 loadDataFromFile( ) メソッド

例 2-13 に、ユーザー定義メソッド loadDataFromFile( ) を示します。このメソッドでは、*inter*Media の loadDataFromFile( ) メソッドを使用してメディア・データをアプリケーション・オブジェクトに移入します。

例 2-13 loadDataFromFile( ) メソッド（Doc）

```
public void loadDataFromFile(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          [2] OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TDOC where n = 1 for update");
          int index = 0;
          [3] while(rs.next( )){
               [4] index = rs.getInt(1);
               [5] OrdDoc docObj = (OrdDoc) rs.getCustomDatum(2,
                    OrdDoc.getFactory( ));
               [6] docObj.loadDataFromFile("testaud.dat");
               [7] docObj.getDataInFile("output1.dat");
               System.out.println("************AFTER getDataInFile ");
               [8] System.out.println("getContentLength output: " +
                    docObj.getContent( ).length( ));
               [9] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update tdoc set doc = ? where n = " +
                    index);
               stmt1.setCustomDatum(1,docObj);
               stmt1.execute( );
               stmt1.close( );
          }
     System.out.println("loading successful");
     }
     [10] catch(Exception e) {
          System.out.println("exception raised " + e);
          System.out.println("loading unsuccessful");
     }
}
```

loadDataFromFile( ) メソッドは、次の操作を行います。

1. OracleStatement オブジェクトを作成します。
2. 所定の SQL 問合せを実行し、その結果をローカル OracleResultSet オブジェクトに格納します。この場合、SQL 問合せは、n=1 の条件を満たすデータベース行のデータを選択します。

3. OracleResultSet に処理されていない結果がある場合、while ループ内の操作を行います。この場合、OracleResultSet に含まれる行は 1 行のみのため、ループ内の操作は一度のみ実行されます。
4. 変数 index に、OracleResultSet の 1 行目の 1 列目にある整数値（この場合、1）を設定します。
5. ローカル OrdDoc オブジェクトとして docObj を作成します。OracleResultSet の現行の行の 2 列目にある OrdDoc オブジェクトのコンテンツを docObj に移入します。
6. OrdDoc の loadDataFromFile( ) メソッドを使用して、testaud.dat 内のメディア・データをデータベース内の OrdDoc オブジェクトおよび docObj にロードします。これによって、ローカル・フィールドが docObj に設定されますが、データベース・オブジェクトには設定されません。
7. getDataInFile( ) メソッドを使用して、docObj からメディア・データを取得し、ローカル・システム上のファイル output1.dat にロードします。
8. ロードが成功したことを確認するため、docObj のコンテンツ長を取得し、画面に出力します。
9. データベース内の OrdDoc オブジェクトを、docObj のコンテンツで更新する SQL 文を作成および実行します。
10. コードによって発生したエラーまたは例外を処理します。

2.2.2.4 extractProperties( ) メソッド

例 2-14 に、ユーザー定義メソッド extractProperties( ) を示します。このメソッドでは、アプリケーション・オブジェクトにプロパティを設定します。

例 2-14 extractProperties( ) メソッド（Doc）

```
public void extractProperties(OracleConnection con){
     byte[ ] ctx[ ] = new byte [4000][1];
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)
          s.executeQuery("select * from TDOC where n = 1 for update");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdDoc docObj = (OrdDoc)rs.getCustomDatum(2,
                    OrdDoc.getFactory( ));
               [2] docObj.setProperties(ctx,false);
               System.out.println("setProperties successful");
               [3] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update tdoc set doc = ? where n = " +
                    index);
```

```
                stmt1.setCustomDatum(1,docObj);
                stmt1.execute( );
                stmt1.close( ) ;
           }
           rs.close( );
           s.close( );
      }
      [4] catch(Exception e) {
           System.out.println("exception raised " + e);
           System.out.println("extract prop unsuccessful");
      }
}
```

extractProperties( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdDoc オブジェクトとして docObj を作成し、例 2-13 の手順 1 ～ 5 と同様の処理でメディア・データを docObj に移入します。このメソッドでは、n=1 の条件を満たす行のコンテンツを操作しています。
2. setProperties をコールして、メディア・データからプロパティ値を抽出し、その値をアプリケーション内の OrdDoc オブジェクトに設定します。抽出および設定されたプロパティ値については、第 4 章の「setProperties( )」を参照してください。
3. データベース内の OrdDoc オブジェクトを、docObj のコンテンツ（setProperties( ) によって抽出されたプロパティを含む）で更新する SQL 文を作成および実行します。
4. コードによって発生したエラーまたは例外を処理します。

2.2.2.5 printProperties( ) メソッド

例 2-15 に、ユーザー定義メソッド printProperties( ) を示します。このメソッドでは、アプリケーション・オブジェクトの属性を画面に出力します。

例 2-15 printProperties( ) メソッド（Doc）

```
public void printProperties(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery("select * from
               TDOC where n = 1 ");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdDoc docObj = (OrdDoc) rs.getCustomDatum(2,
                    OrdDoc.getFactory( ));
               [2] System.out.println("format: " + docObj.getFormat( ));
               System.out.println("mimetype: " + docObj.getMimeType( ));
```

```
                System.out.println("contentlength: " +
                     docObj.getContentLength( ));
           }
      }
      [3] catch(Exception e) {
           System.out.println("exception raised " + e);
           System.out.println("print properties unsuccessful");
      }
}
```

printProperties( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdDoc オブジェクトとして docObj を作成し、例 2-13 の手順 1 ～ 5 と同様の処理でメディア・データを docObj に移入します。このメソッドでは、n=1 の条件を満たす行のコンテンツを操作しています。
2. docObj のプロパティの値を取得し、その値を画面に出力します。
3. コードによって発生した例外を処理します。

2.2.2.6 loadDataFromStream( ) メソッド

例 2-16 に、ユーザー定義メソッド loadDataFromStream( ) を示します。このメソッドでは、*inter*Media の loadDataFromInputStream( ) メソッドを使用して、メディア・データをアプリケーション・オブジェクトにロードします。

例 2-16 loadDataFromStream( ) メソッド（Doc）

```
public void loadDataFromStream(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery("select * from
               TDOC where n = 2 for update ");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdDoc docObj = (OrdDoc) rs.getCustomDatum(2,
                    OrdDoc.getFactory( ));
               [2] FileInputStream  fStream  = new FileInputStream
                    ("testaud.dat");
               [3] docObj.loadDataFromInputStream(fStream);
               [4] docObj.getDataInFile("output2.dat");
               [5] fStream.close( );
               System.out.println("************AFTER getDataInFile ");
               [6] System.out.println("getContentLength output: " +
               docObj.getContent( ).length( ));
               [7] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
```

```
                    con.prepareCall("update tdoc set doc = ? where n = " +
                    index);
               stmt1.setCustomDatum(1,docObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          System.out.println("load data from stream successful");
     }
     [8] catch(Exception e) {
          System.out.println("exception raised " + e);
          System.out.println("load data from stream unsuccessful");
     }
}
```

loadDataFromStream( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdDoc オブジェクトとして docObj を作成し、例 2-13 の手順 1 ～ 5 と同様の処理でメディア・データを docObj に移入します。このメソッドでは、n=2 の条件を満たす行のコンテンツを操作しています。
2. 新規の FileInputStream オブジェクトを作成します。この入力ストリームには、ローカル・ファイル testaud.dat のコンテンツが含まれています。
3. loadDataFromInputStream( ) メソッドを使用して、入力ストリームにあるメディア・データを、データベース内の OrdDoc オブジェクトおよび docObj にロードします。これによって、ローカル・フィールドが docObj に設定されますが、データベース・オブジェクトには設定されません。
4. getDataInFile( ) メソッドを使用して、アプリケーション内の OrdDoc オブジェクトからメディア・データを取得し、ローカル・システム上のファイル output2.dat にロードします。
5. ローカル入力ストリームをクローズします。
6. ロードが成功したことを確認するため、docObj のコンテンツ長を取得し、画面に出力します。
7. データベース内の OrdDoc オブジェクトを、docObj のコンテンツで更新する SQL 文を作成および実行します。この更新によって、アプリケーション・オブジェクトに一致するようにデータベース・オブジェクトの属性が設定されます。
8. コードによって発生したエラーまたは例外を処理します。

2.2.2.7 otherMethods( ) メソッド

例 2-17 に、ユーザー定義メソッド otherMethods( ) を示します。このメソッドでは、processSourceCommand( ) メソッドを使用します。

例 2-17 otherMethods( ) メソッド（Doc）

```
public void otherMethods(OracleConnection con){
     byte[ ] ctx[ ] = {new byte[4000]};
     byte[ ] res[ ] = {new byte[20]};
     [1] int suc = 1;
     try {
          [2] Statement s1 = con.createStatement( );
          OracleResultSet rs1 = (OracleResultSet)
          s1.executeQuery("select * from TDOC where n = 1 for update ");
          int index1 = 0;
          while(rs1.next( )){
               index1 = rs1.getInt(1);
               OrdDoc docObj = (OrdDoc) rs1.getCustomDatum(2,
                    OrdDoc.getFactory( ));
               [3] try {
                    byte[ ] pSRes = docObj.processSourceCommand(ctx, "", "",
                         res);
                    suc = 0;
               }
               [4] catch (Exception e) {
                    System.out.println("Expected Exception raised in
                         processSourceCommand(...)" );
               }
               [5] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update tdoc set doc = ? where n = " +
                    index1);
               stmt1.setCustomDatum(1,docObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          rs1.close( );
          s1.close( );
     }
     [6] catch(Exception e){
          System.out.println("Exception raised " );
     }
     [7] if(suc ==1)
          System.out.println("other methods successful");
     else
          System.out.println("other methods unsuccessful");
}
```

otherMethods( ) メソッドは、次の操作を行います。

1. メソッドの成功または失敗を示すために使用する整数を作成し、これを 1（成功）に初期化します。
2. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdDoc オブジェクトとして docObj を作成し、例 2-13 の手順 1 ～ 5 と同様の処理でメディア・データを docObj に移入します。このメソッドでは、n=1 の条件を満たす行のコンテンツを操作しています。
3. サーバー側でコールされるコマンドを指定せずに processSourceCommand( ) をコールします。これによって例外が発生します。これは、processSourceCommand( ) コールの次のコードは実行されず、catch ループのコードが実行されることを意味します。例外が発生しない場合、メソッドは失敗し、成功インジケータが 0（失敗）に設定されます。
4. 手順 3 で発生した例外を出力します。
5. データベース内の OrdDoc オブジェクトを、docObj のコンテンツで更新する SQL 文を作成および実行します。
6. コードによって発生した、予期しないエラーまたは例外を処理します。
7. メソッドの結果に基づいて、適切なメッセージを画面に出力します。

2.2.2.8 loadDataFromByteArray( ) メソッド

例 2-18 に、ユーザー定義メソッド loadDataFromByteArray( ) を示します。このメソッドでは、*inter*Media の loadDataFromByteArray( ) メソッドを使用して、メディア・データをアプリケーション・オブジェクトにロードします。

例 2-18 loadDataFromByteArray( ) メソッド（Doc）

```
public void loadDataFromByteArray(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet) s.executeQuery("select * from
               TDOC where n = 3 for update ");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdDoc docObj = (OrdDoc) rs.getCustomDatum(2,
                    OrdDoc.getFactory( ));
               [2] File ff = new File("testaud.dat");
               int fileLength = (int) ff.length( );
               byte[ ] data = new byte[fileLength];
               [3] FileInputStream  fStream  = new FileInputStream
                    ("testaud.dat");
               [4] fStream.read(data,0,fileLength);
               [5] docObj.loadDataFromByteArray(data);
```

```
                    [6] fStream.close( );
                    [7] docObj.getDataInFile("output3.dat");
                    [8] byte[ ] resArr = docObj.getDataInByteArray( );
                    [9] System.out.println("byte array length: " + resArr.length);
                    [10] FileOutputStream outStream = new FileOutputStream
                         ("output4.dat");
                    [11] outStream.write(resArr);
                    [12] outStream.close( );
                    [13] InputStream inpStream = docObj.getDataInStream( );
                    int length = 32768;
                    byte[ ] tempBuffer = new byte[32768];
                    [14] int numRead = inpStream.read(tempBuffer,0,length);
                    try {
                         [15] outStream = new FileOutputStream("output5.dat");
                         [16] while(numRead != -1) {
                              [17] if (numRead < 32768) {
                                   length = numRead;
                                   outStream.write(tempBuffer,0,length);
                                   break;
                              }
                              [18] else
                                   outStream.write(tempBuffer,0,length);
                              [19] numRead = inpStream.read(tempBuffer,0,length);
                         }
                    }
                    [20] finally {
                         outStream.close( );
                         inpStream.close( );
                    }
                    System.out.println("************AFTER getDataInFile ");
                    [21] System.out.println("getContentLength output: " +
                         docObj.getContent( ).length( ));
                    [22] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                         con.prepareCall("update tdoc set doc = ? where n = " +
                         index);
                    stmt1.setCustomDatum(1,docObj);
                    stmt1.execute( );
                    stmt1.close( ) ;
               }
          }
          [23] catch(Exception e) {
               System.out.println("exception raised " + e);
               System.out.println("loadData from byte array unsuccessful");
          }
     }
```

loadDataFromByteArray( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdDoc オブジェクトとして docObj を作成し、例 2-13 の手順 1 ～ 5 と同様の処理でメディア・データを docObj に移入します。このメソッドでは、n=3 の条件を満たす行のコンテンツを操作しています。
2. ローカル・ファイル testaud.dat のサイズ（バイト単位）を判断し、同じサイズのバイト配列を作成します。
3. 新規の FileInputStream オブジェクトを作成します。この入力ストリームには、testaud.dat のコンテンツが含まれています。
4. 入力ストリームのコンテンツをバイト配列に読み込みます。
5. loadDataFromByteArray( ) メソッドを使用して、バイト配列にあるメディア・データをデータベース内の OrdDoc オブジェクトおよび docObj にロードします。これによって、ローカル・フィールドが docObj に設定されますが、データベース・オブジェクトには設定されません。
6. 入力ストリームをクローズします。
7. getDataInFile( ) メソッドを使用して、アプリケーション内の OrdDoc オブジェクトからメディア・データを取得し、ローカル・システム上のファイル output3.dat にロードします。
8. getDataInByteArray( ) メソッドを使用して、アプリケーション内の OrdDoc オブジェクトからメディア・データを取得し、バイト配列 resArr にロードします。
9. ロードが成功したことを確認するため、resArr の長さを取得し、画面に出力します。
10. 新規の FileOutputStream オブジェクト outStream を作成します。この出力ストリームは、ローカル・ファイル output4.dat にデータを書き込みます。
11. resArr のコンテンツを output4.dat に書き込みます。
12. 出力ストリームをクローズします。
13. 新規の入力ストリーム inpStream を作成します。getDataInStream( ) メソッドを使用して、アプリケーション内の OrdDoc オブジェクトからメディア・データを取得し、inpStream に格納します。
14. inpStream の先頭（オフセット 0）から 32768 バイトをバイト配列 tempBuffer に読み込みます。整数 numRead には、読込みバイトの合計数、または入力ストリームが最後まで到達した場合は -1 が設定されます。ロードが成功した場合、numRead は 32768 となります。
15. OutStream を再オープンします。この場合、ローカル・ファイル output5.dat にデータを書き込みます。
16. numRead が -1 でない場合は、while ループ内の操作を実行します。プログラムは、このループを実行します。

17. numRead が 32768 未満（データが最後まで読み込まれた）の場合は、if ループを実行します。if ループは、tempBuffer に読み込まれたバイト数を outStream に書き込み、ループから出ます。
18. numRead が 32768 の場合、outStream に 32768 バイトを書き込みます。
19. 入力ストリームからさらにデータをバイト配列に読み込みます。データが最後まで正常に読み込まれた場合、numRead は -1 に設定され、プログラムはループを終了します。入力ストリームに読み込まれていないデータがある場合は、バイト配列に読み込まれ、手順 17、18 が繰り返されます。
20. while ループの終了後、入力ストリームおよび出力ストリームをクローズします。
21. ロードが成功したことを確認するため、docObj のコンテンツ長を取得し、画面に出力します。
22. データベース内の OrdDoc オブジェクトを、docObj のコンテンツで更新する SQL 文を作成および実行します。この更新によって、アプリケーション・オブジェクトに一致するようにデータベース・オブジェクトの属性が設定されます。
23. コードによって発生したエラーまたは例外を処理します。

2.3 OrdImage の例

イメージの例（ImageExample.sql および ImageExample.java）では、SQL、JDBC および *inter*Media Java Classes API を使用して次の操作を行う、ユーザー定義メソッドについて説明します。

- テスト・コンテンツを含むデータベース・サーバー表を作成します。
- ローカル・ファイルからアプリケーションおよびデータベース内の OrdImage オブジェクトにデータをロードします。
- ローカル・ストリームからアプリケーションおよびデータベース内の OrdImage オブジェクトにデータをロードします。
- ローカル・バイト配列からアプリケーションおよびデータベース内の OrdImage オブジェクトにデータをロードします。
- アプリケーション内の OrdImage オブジェクトからプロパティを抽出し、出力します。
- process( ) メソッドおよび processCopy( ) メソッドの例を示します。
- イメージ・シグネチャを生成します。
- 異なる基準で 2 つのイメージ・シグネチャを比較します。
- 2 つのイメージ・シグネチャを比較して、一致するかどうかを判断します。

2.3.1 ImageExample.sql

例 2-19 に ImageExample.sql の内容を示します。

例 2-19 ImageExample.sql の内容

```
set echo on

-- Please Change system password.
connect / as sysdba;
drop user IMAGEUSER cascade;

[1] grant connect,resource to IMAGEUSER identified by IMAGEUSER;

-- Replace C:\Oracle\Ora' with your ORACLE HOME
[2] create or replace directory ORDIMAGEDIR as 'C:\Oracle\Ora\ord\img\demo';
grant read on directory ORDIMAGEDIR to public with grant option;

[3] connect IMAGEUSER/IMAGEUSER;

[4] create table ordimagetab(id number, image ORDSYS.ORDImage, image2
ORDSYS.ORDImage);

-- Note - the OrdImage.init method was added in interMedia 8.1.7.
-- If you are running against an older release of interMedia and the
-- Oracle database, you will have to modify the following INSERT statements
-- to use the OrdImage default constructor.
--
[5] insert into ordimagetab values
(1, ORDSYS.ORDImage.init( ),
    ORDSYS.ORDImage.init( ));

insert into ordimagetab values
(2, ORDSYS.ORDImage.init( ),
    ORDSYS.ORDImage.init( ));

insert into ordimagetab values
(3, ORDSYS.ORDImage.init( ),
    ORDSYS.ORDImage.init( ));

insert into ordimagetab values
(4, ORDSYS.ORDImage.init( ),
    ORDSYS.ORDImage.init( ));

[6] insert into ordimagetab values
(5, ORDSYS.ORDImage.init('file','ORDIMAGEDIR','imgdemo.dat'),
    ORDSYS.ORDImage.init( ));
```

```
insert into ordimagetab values
(6, ORDSYS.ORDImage.init('file','ORDIMAGEDIR','imgdemo.dat'),
    ORDSYS.ORDImage.init( ));

[7] insert into ordimagetab values
(10, ORDSYS.ORDImage.init('file','ORDIMAGEDIR','cbrdemo1.dat'),
     ORDSYS.ORDImage.init( ));
[8] insert into ordimagetab values
(11, ORDSYS.ORDImage.init('file','ORDIMAGEDIR','cbrdemo2.dat'),
     ORDSYS.ORDImage.init( ));

[9] create table sigtable(id number, sig ORDSYS.ORDImageSignature);

[10] insert into sigtable values
(10, ORDSYS.ORDImageSignature.init( ));
insert into sigtable values
(11, ORDSYS.ORDImageSignature.init( ));

commit;
set echo off
exit;
```

ImageExample.sql の SQL 文は、次の操作を行います。

1. ユーザー IMAGEUSER を作成し、必要な権限を付与します。
2. ディレクトリ ORDIMAGEDIR を作成し、必要な権限を設定します。ご使用の Oracle ホームにあわせて、ディレクトリを変更してください。
3. IMAGEUSER でデータベース・サーバーに接続します。
4. 表 ordimagetab を作成します。ordimagetab には、数値の列が 1 列、OrdImage オブジェクトの列が 2 列あります。
5. init メソッドを使用して、2 つの空オブジェクトを持つ行を 4 行追加します。
6. init メソッドを使用して、imgdemo.dat に基づくオブジェクトおよび空オブジェクトを 1 つずつ持つ行を 2 行追加します。
7. init メソッドを使用して、cbrdemo1.dat に基づくオブジェクトおよび空オブジェクトを 1 つずつ持つ行を 1 行追加します。
8. init メソッドを使用して、cbrdemo2.dat に基づくオブジェクトおよび空オブジェクトを 1 つずつ持つ行を 1 行追加します。

9. 表 sigtable を作成します。sigtable には、数値の列が 1 列、OrdImageSignature 数値の列が 1 列あります。
10. init メソッドを使用して、空の OrdImageSignature オブジェクトを持つ行を 2 行追加します。

init メソッドの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

2.3.2 ImageExample.java

2.3.2.1 項～ 2.3.2.12 項では、ImageExample.java サンプル・ファイルに含まれているメソッドを示します。

2.3.2.1 main( ) メソッド

例 2-20 に、main( ) メソッドを示します。

例 2-20 main( ) メソッド（Image）

```
public static void main (String args[ ]){
     byte[ ] ctx = new byte[4000];
     OracleConnection con = null;
     try{
          ImageExample ie = new ImageExample( );
          [1] con = ie.connect( );
          //Include the following line only if you are running
          //an Oracle 8.1.7 database or later.
          //If you are running a database server prior to 8.1.7,
          //the call will fail.
          [2] OrdMediaUtil.imCompatibilityInit(con);
          [3] ie.setPropertiesExample(con);
          ie.displayPropertiesExample(con);
          ie.fileBasedExample(con);
          ie.streamBasedExample(con);
          ie.byteArrayBasedExample(con);
          ie.processExample(con);
          [4] ie.sigGeneration(con);
          ie.sigSimilarity(con);
          ie.imageMatchingScore(con);
          [5] con.commit( );
          [6] con.close( );
          System.out.println("Done.");
     }
```

```
     [7] catch (Exception e){
          try{
               System.out.println("Exception : " + e);
               con.close( );
          }
          catch(Exception ex){
               System.out.println("Close Connection Exception : " + ex);
          }
     }
}
```

main( ) メソッドは、次の操作を行います。

1. connect( ) メソッドを使用して、データベース表への接続を確立します。
2. ご使用のアプリケーションと新しいリリースの Oracle データベースの互換性を保証します。詳細は、1.8 項を参照してください。
3. データベース・サーバーおよびローカル・マシン上のオブジェクトを操作する、複数のメソッド（ImageExample.java で定義済）をコールします。
4. OrdImageSignature オブジェクトをテストする複数のメソッド（ImageExample.java で定義済）をコールします。
5. データベース表への変更をすべてコミットします。
6. データベースへの接続をクローズします。
7. コードによって発生したエラーまたは例外を処理します。

2.3.2.2 項～ 2.3.2.12 項では、main( ) メソッドからコールされるメソッドについて、説明します。

2.3.2.2 connect( ) メソッド

例 2-21 に、ユーザー定義メソッド connect( ) を示します。このメソッドでは、アプリケーションからデータベースへの接続を確立します。

例 2-21 connect( ) メソッド（Image）

```
public OracleConnection connect( ) throws Exception{
     String connectString;
     [1] Class.forName ("oracle.jdbc.driver.OracleDriver");
     [2] connectString = "jdbc:oracle:oci8:@";
     [3] OracleConnection con = (OracleConnection)DriverManager.getConnection
          (connectString,"IMAGEUSER","IMAGEUSER");
     [4] con.setAutoCommit(false);
     return con;
}
```

connect( ) メソッドは、次の操作を行います。

1. Oracle データベースでは、JDK 準拠の JVM を使用するため、JDBC ドライバを直接ロードします。
2. 接続するデータベースの URL を含む文字列を定義します。ご使用のデータベースにあわせて、この文字列を変更してください。
3. connectString に指定された URL、ユーザー名 IMAGEUSER およびパスワード IMAGEUSER を使用してデータベースへの接続を設定します。ユーザー名およびパスワードは、ImageExample.sql で作成済です。
4. 自動コミット・モードを使用禁止にします。これは、commit( ) メソッドまたは rollback( ) メソッドを使用して、それぞれ手動でコミットまたはロールバックする必要があることを意味します。

2.3.2.3 setPropertiesExample( ) メソッド

例 2-22 に、ユーザー定義メソッド setPropertiesExample( ) を示します。このメソッドでは、アプリケーション・オブジェクトにプロパティを設定します。

例 2-22 setPropertiesExample( ) メソッド（Image）

```
public void setPropertiesExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          [2] OracleResultSet rs = (OracleResultSet)s.executeQuery
               ("select * from ordimagetab where id = 5 for update");
          [3] while(rs.next( )){
               [4] index = rs.getInt(1);
               [5] OrdImage imgObj = (OrdImage)rs.getCustomDatum
                    (2, OrdImage.getFactory( ));
               [6] imgObj.setProperties( );
               System.out.println("set Properties called");
               [7] if(imgObj.checkProperties( )){
                    System.out.println("checkProperties called");
                    System.out.println("setProperties successful");
                    System.out.println("checkProperties successful");
                    System.out.println("successful");
               }
               else{
                    System.out.println("checkProperties called");
                    System.out.println("setProperties not successful");
                    System.out.println("checkProperties successful");
               }
```

```
                [8] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                     con.prepareCall("update ordimagetab set
                     image = ? where id = " + index);
                stmt1.setCustomDatum(1,imgObj);
                stmt1.execute( );
                stmt1.close( ) ;
            }
            rs.close( );
            s.close( );
        }
        [9] catch(Exception e){
            System.out.println("exception raised " + e);
        }
}
```

setPropertiesExample( ) メソッドは、次の操作を行います。

1. OracleStatement オブジェクトを作成します。
2. 所定の SQL 問合せを実行し、その結果をローカル OracleResultSet オブジェクトに格納します。この場合、SQL 問合せは、id=5 の条件を満たすデータベース行のデータを選択します。
3. OracleResultSet に処理されていない結果がある場合、while ループ内の操作を行います。この場合、OracleResultSet に含まれる行は 1 行のみのため、ループ内の操作は一度のみ実行されます。
4. 変数 index に、OracleResultSet の 1 行目の 1 列目にある整数値（この場合、5）を設定します。
5. ローカル OrdImage オブジェクト imgObj を作成します。OracleResultSet の現行の行の 2 列目にある OrdImage オブジェクトのコンテンツを、imgObj に移入します。
6. setProperties( ) をコールして、メディア・データからプロパティ値を抽出し、その値をアプリケーション内の OrdImage オブジェクトに設定します。抽出および設定されたプロパティ値については、第 5 章の「setProperties( )」を参照してください。
7. checkProperties( ) をコールして、アプリケーション・オブジェクトのプロパティ値とメディア・データの値を比較します。値がすべて同じである場合、checkProperties( ) は true を戻し、画面に適切なメッセージを表示します。値が異なる場合は false を戻し、画面に適切なメッセージを表示します。
8. データベース内の OrdImage オブジェクトを、imgObj のコンテンツで更新する SQL 文を作成および実行します。
9. コードによって発生したエラーまたは例外を処理します。

2.3.2.4 displayPropertiesExample( ) メソッド

例 2-23 に、ユーザー定義メソッド displayPropertiesExample( ) を示します。このメソッドでは、アプリケーション・オブジェクトの属性を画面に出力します。

例 2-23 displayPropertiesExample( ) メソッド（Image）

```
public void displayPropertiesExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery(
               "select * from ordimagetab where id = 5 ");
          while(rs.next( )){
               index = rs.getInt(1);
               OrdImage imgObj = (OrdImage) rs.getCustomDatum(2,
                    OrdImage.getFactory( ));
               [2] System.out.println("format : " + imgObj.getFormat( ));
               System.out.println("mimeType: " + imgObj.getMimeType( ));
               System.out.println("height: " + imgObj.getHeight( ));
               System.out.println("width: " + imgObj.getWidth( ));
               System.out.println("contentLength: " +
                    imgObj.getContentLength( ));
               System.out.println("contentFormat: " +
                    imgObj.getContentFormat( ));
               System.out.println("compressionFormat: " +
                    imgObj.getCompressionFormat( ));
               System.out.println("source type: " +
                    imgObj.getSourceType( ));
               System.out.println("source loc: " +
                    imgObj.getSourceLocation( ));
               System.out.println("source name: " + imgObj.getSourceName( ));
               System.out.println("source : " + imgObj.getSource( ));
               [3] try{
                    String attrString = getAllAttributesAsString(imgObj);
                    System.out.println(attrString);
               }
               [4] catch (Exception e){
                    System.out.println("Exception raised in
                         getAllAttributesAsString:");
               }
               System.out.println("successful");
          }
     }
```

```
     [5] catch(Exception e) {
          System.out.println("exception raised " + e);
     }
}
```

displayPropertiesExample( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdImage オブジェクトとして imgObj を作成し、例 2-22 の手順 1 ～ 5 と同様の手順で imgObj にメディア・データを移入します。このメソッドでは、id=5 の条件を満たす行のコンテンツを操作しています。この行は、例 2-22 で操作した行と同じです。
2. imgObj のプロパティ値を取得し、その値を画面に出力します。
3. getAllAttributesAsString( ) メソッドを使用して、imgObj の属性を取得して文字列に格納し、その文字列の内容を画面に出力します。getAllAttributesAsString( ) の詳細は、2.3.2.5 項を参照してください。
4. getAllAttributesAsString( ) をコールして発生したエラーまたは例外を処理します。
5. 一般的なコードによって発生したエラーまたは例外を処理します。

2.3.2.5 getAllAttributesAsString( ) メソッド

例 2-24 に、ユーザー定義メソッド getAllAttributesAsString( ) を示します。このメソッドでは、アプリケーション・オブジェクト属性の値を含む String オブジェクトを作成します。

例 2-24 getAllAttributesAsString( ) メソッド（Image）

```
public String getAllAttributesAsString (OrdImage imgObj) throws Exception{
     [1] String attStr = imgObj.getSource( ) + " mimeType = " +
          imgObj.getMimeType( ) + ", fileFormat = " +
          imgObj.getFormat( ) + ", height = " + imgObj.getHeight( )
          + ", width = " + imgObj.getWidth( ) + ", contentLength = "
          + imgObj.getContentLength( ) + ", contentFormat = " +
          imgObj.getContentFormat( ) + ", compressionFormat = " +
          imgObj.getCompressionFormat( );
     [2] return attStr;
}
```

getAllAttributesAsString( ) メソッドは、次の操作を行います。

1. String オブジェクト attStr を作成します。アプリケーション・イメージ・オブジェクトから複数の属性の値を取得し、attStr に格納します。
2. attStr を、このメソッドをコールしたメソッドに戻します。

2.3.2.6 fileBasedExample( ) メソッド

例 2-25 に、ユーザー定義メソッド fileBasedExample( ) を示します。このメソッドでは、loadDataFromFile( ) メソッドを使用して、アプリケーション・オブジェクトにメディア・データをロードします。

例 2-25 fileBasedExample( ) メソッド（Image）

```
public void fileBasedExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery(
               "select * from ORDIMAGETAB where id = 2 for update ");
          while(rs.next( )){
               index = rs.getInt(1);
               OrdImage imgObj = (OrdImage) rs.getCustomDatum(2,
                    OrdImage.getFactory( ));
               [2] imgObj.loadDataFromFile("imgdemo.dat");
               [3] imgObj.setProperties( );
               [4] imgObj.getDataInFile("fileexample.dat");
               [5] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update ordimagetab set image =
                    ? where id = " + index);
               stmt1.setCustomDatum(1,imgObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          System.out.println("successful");
     }
     [6] catch(Exception e){
          System.out.println("exception raised " + e);
     }
}
```

fileBasedExample( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdImage オブジェクトとして imgObj を作成し、例 2-22 の手順 1 ～ 5 と同様の手順で imgObj にメディア・データを移入します。このメソッドでは、id=2 の条件を満たす行のコンテンツを操作しています。
2. loadDataFromFile( ) メソッドを使用して、ローカル・ファイル imgdemo.dat からメディア・データをデータベース内の OrdImage オブジェクトおよび imgObj にロードします。これによって、ローカル・フィールドが imgObj に設定されますが、データベース・オブジェクトには設定されません。

3. setProperties( ) をコールして、メディア・データからプロパティ値を抽出し、その値をアプリケーション内の OrdImage オブジェクトに設定します。抽出および設定されたプロパティ値については、第 5 章の「setProperties( )」を参照してください。
4. getDataInFile( ) メソッドを使用して、アプリケーション OrdImage オブジェクトからメディア・データを取得し、ローカル・システム上のファイル fileexample.dat にロードします。
5. データベース内の OrdImage オブジェクトを、imgObj のコンテンツで更新する SQL 文を作成および実行します。この更新によって、アプリケーション・オブジェクトに一致するように、データベース・オブジェクトの属性が設定されます。
6. コードによって発生したエラーまたは例外を処理します。

2.3.2.7 streamBasedExample( ) メソッド

例 2-26 に、ユーザー定義メソッド streamBasedExample( ) を示します。このメソッドでは、loadDataFromInputStream( ) メソッドを使用して、アプリケーション・オブジェクトにメディア・データをロードします。

例 2-26 streamBasedExample( ) メソッド（Image）

```
public void streamBasedExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery(
               "select * from ORDIMAGETAB where id = 3 for update ");
          while(rs.next( )){
               index = rs.getInt(1);
               OrdImage imgObj = (OrdImage) rs.getCustomDatum(2,
                    OrdImage.getFactory( ));
               [2] FileInputStream fStream = new FileInputStream
                    ("imgdemo.dat");
               [3] imgObj.loadDataFromInputStream(fStream);
               [4] fStream.close( );
               [5] imgObj.setProperties( );
               [6] InputStream inpStream = imgObj.getDataInStream( );
               int length = 32300;
               byte[ ] tempBuffer = new byte[length];
               [7] int numRead = inpStream.read(tempBuffer,0,length);
               FileOutputStream outStream=null;
               try{
                    [8] outStream = new FileOutputStream
                         ("streamexample.dat");
                    [9] while(numRead != -1){
                         [10] if (numRead < length){
                              length = numRead;
```

```
                    outStream.write(tempBuffer,0,length);
                    break;
                }
                [11] else
                    outStream.write(tempBuffer,0,length);
                [12] numRead = inpStream.read(tempBuffer,0,
                    length);
            }
        }
        [13] finally{
            if (outStream != null)
                outStream.close( );
            inpStream.close( );
        }
        [14] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
            con.prepareCall("update ordimagetab set
            image = ? where id = " + index);
        stmt1.setCustomDatum(1,imgObj);
        stmt1.execute( );
        stmt1.close( ) ;
    }
    System.out.println("successful");
  }
  [15] catch(Exception e){
    System.out.println("exception raised " + e);
  }
}
```

streamBasedExample( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdImage オブジェクトとして imgObj を作成し、例 2-22 の手順 1 ～ 5 と同様の手順で imgObj にメディア・データを移入します。このメソッドでは、id=3 の条件を満たす行のコンテンツを操作しています。
2. 新規の FileInputStream オブジェクトを作成します。この入力ストリームには、ローカル・ファイル imgdemo.dat のコンテンツが含まれています。
3. loadDataFromInputStream( ) メソッドを使用して、入力ストリームにあるメディア・データをデータベース内の OrdImage オブジェクトおよび imgObj にロードします。これによって、ローカル・フィールドが imgObj に設定されますが、データベース・オブジェクトには設定されません。
4. 入力ストリームをクローズします。
5. setProperties( ) をコールして、メディア・データからプロパティ値を抽出し、その値をアプリケーション内の OrdImage オブジェクトに設定します。抽出および設定されたプロパティ値については、第 5 章の「setProperties( )」を参照してください。

6. 新規の入力ストリーム inpStream を作成します。getDataInStream( ) メソッドをコールして、アプリケーション内の OrdImage オブジェクトからメディア・データを取得し、inpStream に格納します。
7. inpStream の先頭（オフセット 0）から 32300 バイトをバイト配列 tempBuffer に読み込みます。整数 numRead には、読込みバイトの合計数、または入力ストリームが最後まで到達した場合は -1 が設定されます。ロードが成功した場合、numRead は 32300 となります。
8. 新規の FileOutputStream オブジェクト outStream を作成します。この出力ストリームは、ローカル・ファイル streamexample.dat にデータを書き込みます。
9. numRead が -1 でない場合は、while ループ内の操作を実行します。プログラムは、このループを実行します。
10. numRead が 32300 未満（データが最後まで読み込まれなかった）の場合は、tempBuffer に読み込まれたバイト数を、outStream に書き込みます。
11. numRead が 32300 の場合、outStream に 32300 バイトを書き込みます。
12. 入力ストリームからさらにデータをバイト配列に読み込みます。データが最後まで正常に読み込まれた場合、numRead は -1 に設定され、プログラムはループを終了します。入力ストリームに読み込まれていないデータがある場合は、バイト配列に読み込まれ、手順 10、11 が繰り返されます。
13. while ループの終了後、入力ストリームおよび出力ストリームをクローズします。
14. データベース内の OrdImage オブジェクトを、imgObj のコンテンツで更新する SQL 文を作成および実行します。この更新によって、アプリケーション・オブジェクトに一致するようにデータベース・オブジェクトの属性が設定されます。
15. コードによって発生したエラーまたは例外を処理します。

2.3.2.8 byteArrayBasedExample( ) メソッド

例 2-27 に、ユーザー定義メソッド byteArrayBasedExample( ) を示します。このメソッドでは、loadDataFromByteArray( ) メソッドを使用して、アプリケーション・オブジェクトにメディア・データをロードします。

例 2-27 byteArrayBasedExample( ) メソッド（Image）

```
public void byteArrayBasedExample(OracleConnection con){
     try{
          int index = 0;
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery
               ("select * from ORDIMAGETAB where id = 4 for update ");
          while(rs.next( )){
               index = rs.getInt(1);
               OrdImage imgObj = (OrdImage) rs.getCustomDatum(2,
```

```
                    OrdImage.getFactory( ));
                [2] File ff = new File("imgdemo.dat");
                int fileLength = (int) ff.length( );
                byte[ ] data = new byte[fileLength];
                [3] FileInputStream fStream = new
                    FileInputStream("imgdemo.dat");
                [4] fStream.read(data,0,fileLength);
                [5] imgObj.loadDataFromByteArray(data);
                [6] fStream.close( );
                [7] imgObj.setProperties( );
                [8] byte[ ] resArr = imgObj.getDataInByteArray( );
                [9] System.out.println("byte array length : " +
                    resArr.length);
                [10] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update ordimagetab set image =
                    ? where id = " + index);
                stmt1.setCustomDatum(1,imgObj);
                stmt1.execute( );
                stmt1.close( ) ;
            }
            System.out.println("successful");
        }
        [11] catch(Exception e){
            System.out.println("exception raised " + e);
        }
}
```

byteArrayBasedExample( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdImage オブジェクトとして imgObj を作成し、例 2-22 の手順 1 ～ 5 と同様の手順で imgObj にメディア・データを移入します。このメソッドでは、id=4 の条件を満たす行のコンテンツを操作しています。
2. ローカル・ファイル imgdemo.dat のサイズ（バイト単位）を判断し、同じサイズのバイト配列を作成します。
3. 新規の FileInputStream オブジェクトを作成します。この入力ストリームには、imgdemo.dat のコンテンツが含まれています。
4. 入力ストリームのコンテンツをバイト配列に読み込みます。
5. loadDataFromByteArray( ) メソッドを使用して、バイト配列にあるメディア・データをデータベース内の OrdImage オブジェクトおよび imgObj にロードします。これによって、ローカル・フィールドが imgObj に設定されますが、データベース・オブジェクトには設定されません。
6. 入力ストリームをクローズします。

7. setProperties( ) をコールして、メディア・データからプロパティ値を抽出し、その値をアプリケーション内の OrdImage オブジェクトに設定します。抽出および設定されたプロパティ値については、第 5 章の「setProperties( )」を参照してください。
8. getDataInByteArray( ) メソッドを使用して、アプリケーション内の OrdImage オブジェクトからメディア・データを取得し、バイト配列 resArr にロードします。
9. ロードが成功したことを確認するため、resArr の長さを取得し、画面に出力します。
10. データベース内の OrdImage オブジェクトを、imgObj のコンテンツで更新する SQL 文を作成および実行します。この更新によって、アプリケーション・オブジェクトに一致するようにデータベース・オブジェクトの属性が設定されます。
11. コードによって発生したエラーまたは例外を処理します。

2.3.2.9 processExample( ) メソッド

例 2-28 に、ユーザー定義メソッド processExample( ) を示します。このメソッドでは、process( ) メソッドおよび processCopy( ) メソッドを使用して、アプリケーション・オブジェクトにあるメディア・データを操作します。

例 2-28 processExample( ) メソッド（Image）

```
public void processExample(OracleConnection con){
     try{
          int index1 = 0;
          [1] Statement s1 = con.createStatement( );
          OracleResultSet rs1 = (OracleResultSet)s1.executeQuery
               ("select * from ORDIMAGETAB where id = 2 for update ");
          while(rs1.next( )){
               index1 = rs1.getInt(1);
               OrdImage imgObj = (OrdImage) rs1.getCustomDatum(2,
                    OrdImage.getFactory( ));
               [2] OrdImage imgObj2 = (OrdImage) rs1.getCustomDatum(3,
                    OrdImage.getFactory( ));
               try{
                    [3] imgObj.processCopy("maxScale=32 32, fileFormat=
                         GIFF", imgObj2);
                    [4] imgObj.process("fileFormat=JFIF");
                    [5] System.out.println(getAllAttributesAsString
                         (imgObj));
                    [6] System.out.println(getAllAttributesAsString(imgObj2));
               }
               [7] catch (Exception e){
                    System.out.println("Exception raised in process"
                         + e );
               }
```

```
                [8] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                     con.prepareCall("update ordimagetab set image =
                     ?, image2 = ? where id = " + index1);
                stmt1.setCustomDatum(1,imgObj);
                stmt1.setCustomDatum(2,imgObj2);
                stmt1.execute( );
                stmt1.close( ) ;
            }
            rs1.close( );
            s1.close( );
        }
        [9] catch(Exception e){
            System.out.println("Exception raised: " + e);
        }
        System.out.println("successful");
}
```

processExample( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdImage オブジェクトとして imgObj を作成し、例 2-22 の手順 1 ～ 5 と同様の手順で imgObj にメディア・データを移入します。このメソッドでは、id=2 の条件を満たす行の 2 列目のコンテンツを操作しています。データベース内の OrdImage オブジェクトは、image という名前です。
2. ローカル OrdImage オブジェクト imgObj2 を作成します。OracleResultSet の現行の行の 3 列目にある OrdImage オブジェクトのコンテンツを、imgObj2 に移入します。データベース内の OrdImage 列は、image2 という名前です。
3. imgObj にあるイメージ・データから生成された 32 × 32 の GIF サムネイル・イメージを、imgObj2 のイメージ・データに移入します。imgObj は、この操作では変更されません。
4. process( ) メソッドを使用して、imgObj にあるイメージを JPEG（JFIF）イメージに変換します。
5. getAllAttributesAsString( ) メソッドを使用して imgObj の属性を取得し、その属性を画面に出力します。getAllAttributesAsString( ) の詳細は、2.3.2.5 項を参照してください。
6. getAllAttributesAsString( ) メソッドを使用して imgObj2 の属性を取得し、その属性を画面に出力します。getAllAttributesAsString( ) の詳細は、2.3.2.5 項を参照してください。
7. 手順 3 ～ 6 のコードによって発生したエラーまたは例外を処理します。
8. 適切なデータベース内の OrdImage オブジェクトを、imgObj および imgObj2 のコンテンツで更新する SQL 文を作成および実行します。
9. コードによって発生したエラーまたは例外を処理します。

2.3.2.10 sigGeneration( ) メソッド

例 2-29 に、ユーザー定義メソッド sigGeneration( ) を示します。このメソッドでは、指定された OrdImage オブジェクトのシグネチャを生成し、データベースに格納します。

例 2-29 sigGeneration( ) メソッド（Image）

```
public void sigGeneration(OracleConnection con){
     byte[ ] ctx[ ] = new byte [4000][1];
     try{
          [1] int IMG_ID=10;
          int SIG_ID=11;
          int IMG_COL=2;
          int SIG_COL=2;
          [2] Statement get_image_obj = con.createStatement( );
          Statement get_sig_obj = con.createStatement( );
          [3] OracleResultSet get_image_obj_result =
               (OracleResultSet)get_image_obj.executeQuery ("select * from
               ordimagetab where id =" + IMG_ID + " for update");
          OracleResultSet get_sig_result = (OracleResultSet)
               get_sig_obj.executeQuery("select * from sigtable where id =" +
               SIG_ID + " for update");
          [4] get_image_obj_result.next( );
          get_sig_result.next( );
          [5] OrdImage img_obj = (OrdImage)get_image_obj_result.getCustomDatum
               (IMG_COL, OrdImage.getFactory( ));
          [6] OrdImageSignature img_sig = (OrdImageSignature)
               get_sig_result.getCustomDatum(SIG_COL,
               OrdImageSignature.getFactory( ));
          [7] img_obj.setProperties( );
          [8] img_obj.importData(ctx);
          [9] img_sig.generateSignature(img_obj);
          [10] OraclePreparedStatement update_sig = (OraclePreparedStatement)
               con.prepareCall("update sigtable set sig = ?
               where id =" + SIG_ID);
          update_sig.setCustomDatum(1,img_sig);
          update_sig.execute( );
          update_sig.close( ) ;
          [11] get_sig_result.close( );
          get_image_obj_result.close( );
          get_sig_obj.close( );
          get_image_obj.close( );
          System.out.println("image signature generation complete.");
     }
```

```
    [12] catch(Exception e){
         System.out.println("exception raised " + e);
         System.out.println("image signature generation failed");
    }
}
```

sigGeneration( ) メソッドは、次の操作を行います。

1. 次の値を持つ 4 つの整数を作成します。
   - IMG_ID は、ORDIMAGETAB の OrdImage オブジェクトの主キーで、シグネチャが生成されます。
   - SIG_ID は、SIGTABLE の OrdImageSignature オブジェクトの主キーで、シグネチャが格納されます。
   - IMG_COL は、ORDIMAGETAB の OrdImage オブジェクトの列番号です。
   - SIG_COL は、SIGTABLE の OrdImageSignature オブジェクトの列番号です。
2. 2 つの Statement オブジェクトを作成します。
3. 所定の SQL 問合せを実行し、その結果を 2 つのローカル OracleResultSet オブジェクトに格納します。この場合、SQL 問合せは次の位置にあるデータを選択します。
   - ORDIMAGETAB の主キーが 10 であるデータベース行
   - SIGTABLE の主キーが 11 であるデータベース行
4. 結果セットの適切な行に進みます。
5. ローカル OrdImage オブジェクトとして img_obj を作成します。結果セットの現行の行の指定された列にある OrdImage オブジェクトのコンテンツを、img_obj に移入します。
6. ローカル OrdImageSignature オブジェクトとして img_sig を作成します。結果セットの現行の行の指定された列にある OrdImageSignature オブジェクトのコンテンツを、img_sig に移入します。
7. setProperties( ) をコールして、メディア・データからプロパティ値を抽出し、その値をアプリケーション内の OrdImage オブジェクトに設定します。抽出および設定されたプロパティ値については、第 5 章の「setProperties( )」を参照してください。
8. データをローカル OrdImage オブジェクトにインポートします。
9. img_obj のシグネチャを生成して、img_sig に格納します。
10. データベース内の OrdImageSignature オブジェクトを、img_sig のコンテンツで更新する SQL 文を作成および実行します。
11. OracleResultSet および Statement オブジェクトを明示的にクローズします。
12. コードによって発生したエラーまたは例外を処理します。

2.3.2.11 imageMatchingScore( ) メソッド

例 2-30 に、imageMatchingScore( ) メソッドを示します。このメソッドは、2 つの OrdImage オブジェクトのシグネチャを生成し、異なる基準で比較します。

例 2-30 imageMatchingScore( ) メソッド（Image）

```
public void imageMatchingScore(OracleConnection con){
     byte[ ] ctx[ ] = new byte [4000][1];
     try {
          [1] int IMG_ID1=10;
          int SIG_ID1=10;
          int IMG_ID2=11;
          int SIG_ID2=11;
          int IMG_COL=2;
          int SIG_COL=2;
          [2] Statement get_image_obj = con.createStatement( );
          Statement get_sig_obj = con.createStatement( );
          [3] OracleResultSet get_image_obj_result1 = (OracleResultSet)
               get_image_obj.executeQuery("select * from ordimagetab
               where id ="+ IMG_ID1 +" for update");
          OracleResultSet get_sig_result1 = (OracleResultSet)
               get_sig_obj.executeQuery("select * from sigtable
               where id =" +SIG_ID1 + " for update");
          OracleResultSet get_image_obj_result2 = (OracleResultSet)
               get_image_obj.executeQuery("select * from ordimagetab
               where id ="+ IMG_ID2 +" for update");
          OracleResultSet get_sig_result2 = (OracleResultSet)
               get_sig_obj.executeQuery("select * from sigtable
               where id =" +SIG_ID2 + " for update");
          [4] get_image_obj_result1.next( );
          get_sig_result1.next( );
          get_image_obj_result2.next( );
          get_sig_result2.next( );
          [5] OrdImage img_obj1 = (OrdImage)
               get_image_obj_result1.getCustomDatum(IMG_COL,
               OrdImage.getFactory( ));
          [6] OrdImageSignature img_sig1 = (OrdImageSignature)
               get_sig_result1.getCustomDatum(
               SIG_COL,OrdImageSignature.getFactory( ));
          [7] img_obj1.setProperties( );
          [8] img_obj1.importData(ctx);
          [9] img_sig1.generateSignature(img_obj1);
          [10] OrdImage img_obj2 = (OrdImage)get_image_obj_result2.
               getCustomDatum(IMG_COL, OrdImage.getFactory( ));
          [11] OrdImageSignature img_sig2 = (OrdImageSignature)
               get_sig_result2.getCustomDatum(
               SIG_COL,OrdImageSignature.getFactory( ));
```

```
            [12] img_obj2.setProperties( );
            [13] img_obj2.importData(ctx);
            [14] img_sig2.generateSignature(img_obj2);
            [15] float gscore = OrdImageSignature.evaluateScore(img_sig1,
                 img_sig2,"color=1", con);
                 System.out.println("score value (global color comparison):" +
                 gscore);
            [16] float lscore = OrdImageSignature.evaluateScore(img_sig1,
                 img_sig2,"color=1 location=1", con);
                 System.out.println("Score value (local color comparison):" + lscore);
            [17] get_sig_result2.close( );
                 get_image_obj_result2.close( );
                 get_sig_result1.close( );
                 get_image_obj_result1.close( );
                 get_sig_obj.close( );
                 get_image_obj.close( );
        }
        [18] catch(Exception e){
             System.out.println("exception raised " + e);
             System.out.println("Image matching score computation failed");
        }
}
```

imageMatchingScore( ) メソッドは、次の操作を行います。

1. 次の値を持つ 6 つの整数を作成します。
   - IMG_ID1 は、ORDIMAGETAB の OrdImage オブジェクトの主キーで、1 つ目のシグネチャが生成されます。
   - SIG_ID1 は、SIGTABLE の OrdImageSignature オブジェクトの主キーで、1 つ目のシグネチャが格納されます。
   - IMG_ID2 は、ORDIMAGETAB の OrdImage オブジェクトの主キーで、2 つ目のシグネチャが生成されます。
   - SIG_ID2 は、SIGTABLE の OrdImageSignature オブジェクトの主キーで、2 つ目のシグネチャが格納されます。
   - IMG_COL は、ORDIMAGETAB の OrdImage オブジェクトの列番号です。
   - SIG_COL は、SIGTABLE の OrdImageSignature オブジェクトの列番号です。
2. 2 つの Statement オブジェクトを作成します。

3. 所定の SQL 問合せを実行し、その結果を 4 つのローカル OracleResultSet オブジェクトに格納します。この場合、SQL 問合せは次の位置にあるデータを選択します。
   - ORDIMAGETAB の主キーが 10 であるデータベース行
   - SIGTABLE の主キーが 10 であるデータベース行
   - ORDIMAGETAB の主キーが 11 であるデータベース行
   - SIGTABLE の主キーが 11 であるデータベース行
4. 結果セットの適切な行に進みます。
5. ローカル OrdImage オブジェクトとして img_obj1 を作成します。ORDIMAGETAB の主キーが 10 である行の 2 列目にある OrdImage オブジェクトのコンテンツを、img_obj1 に移入します。
6. ローカル OrdImageSignature オブジェクトとして img_sig1 を作成します。SIGTABLE の主キーが 10 である行の 2 列目にある OrdImageSignature オブジェクトのコンテンツを、img_sig1 に移入します。
7. setProperties( ) をコールし、メディア・データからプロパティ値を抽出し、その値を img_obj1 に設定します。抽出および設定されたプロパティ値については、第 5 章の「setProperties( )」を参照してください。
8. データを img_obj1 にインポートします。
9. img_obj1 のシグネチャを生成して、img_sig1 に格納します。
10. ローカル OrdImage オブジェクトとして img_obj2 を作成します。ORDIMAGETAB の主キーが 11 である行の 2 列目にある OrdImage オブジェクトのコンテンツを、img_obj2 に移入します。
11. ローカル OrdImageSignature オブジェクトとして img_sig2 を作成します。SIGTABLE の主キーが 11 である行の 2 列目にある OrdImageSignature オブジェクトのコンテンツを、img_sig2 に移入します。
12. setProperties( ) をコールし、メディア・データからプロパティ値を抽出し、その値を img_obj2 に設定します。抽出および設定されたプロパティ値については、第 5 章の「setProperties( )」を参照してください。
13. データを img_obj2 にインポートします。
14. img_obj2 のシグネチャを生成して、img_sig2 に格納します。
15. 2 つのイメージ・シグネチャを色属性に基づいて比較して、新しく生成されたスコアを画面に表示します。
16. 2 つのイメージ・シグネチャを色および位置属性に基づいて比較して、新しく生成されたスコアを画面に出力します。
17. Statement および OracleResultSet オブジェクトを明示的にクローズします。
18. メソッドによって発生した例外またはエラーを処理します。

2.3.2.12 sigSimilarity( ) メソッド

例 2-31 に、sigSimilarity( ) メソッドを示します。このメソッドは、2 つのオブジェクトのイメージ・シグネチャを比較して、一致するかどうかを判断します。

例 2-31 sigSimilarity( ) メソッド（Image）

```
public void sigSimilarity(OracleConnection con){
     byte[ ] ctx[ ] = new byte [4000][1];
     try {
          [1] int IMG_ID1=10;
          int SIG_ID1=10;
          int IMG_ID2=11;
          int SIG_ID2=11;
          int IMG_COL=2;
          int SIG_COL=2;
          [2] double similarity_threshold= 20.0;
          [3] Statement get_image_obj = con.createStatement( );
          Statement get_sig_obj = con.createStatement( );
          [4] OracleResultSet get_image_obj_result1 = (OracleResultSet)
               get_image_obj.executeQuery("select * from ordimagetab
               where id ="+ IMG_ID1 +" for update");
          OracleResultSet get_sig_result1 = (OracleResultSet)
               get_sig_obj.executeQuery("select * from sigtable
               where id =" +SIG_ID1 + " for update");
          OracleResultSet get_image_obj_result2 = (OracleResultSet)
               get_image_obj.executeQuery("select * from ordimagetab
               where id ="+ IMG_ID2 +" for update");
          OracleResultSet get_sig_result2 = (OracleResultSet)
               get_sig_obj.executeQuery("select * from sigtable
               where id =" +SIG_ID2 + " for update");
          [5] get_image_obj_result1.next( );
          get_sig_result1.next( );
          get_image_obj_result2.next( );
          get_sig_result2.next( );
          [6] OrdImage img_obj1 = (OrdImage)get_image_obj_result1.
               getCustomDatum(IMG_COL, OrdImage.getFactory( ));
          [7] OrdImageSignature img_sig1 = (OrdImageSignature)get_sig_result1.
               getCustomDatum(SIG_COL,OrdImageSignature.getFactory( ));
          [8] img_obj1.setProperties( );
          [9] img_obj1.importData(ctx);
          [10] img_sig1.generateSignature(img_obj1);
          [11] OrdImage img_obj2 = (OrdImage) get_image_obj_result2.
               getCustomDatum(IMG_COL, OrdImage.getFactory( ));
          [12] OrdImageSignature img_sig2 = (OrdImageSignature)get_sig_result2.
               getCustomDatum(SIG_COL,OrdImageSignature.getFactory( ));
          [13] img_obj2.setProperties( );
          [14] img_obj2.importData(ctx);
```

```
            [15] img_sig2.generateSignature(img_obj2);
            [16] int result = OrdImageSignature.isSimilar(img_sig1, img_sig2,
                 "color=1 texture=1 shape=1 location=1",
                 (float)similarity_threshold,con);
            if (result==1){
                 System.out.println("Signatures are similar ");
            }
            else{
                 System.out.println("Signatures are different ");
            }
            [17] get_sig_result2.close( );
            get_image_obj_result2.close( );
            get_sig_result1.close( );
            get_image_obj_result1.close( );
            get_sig_obj.close( );
            get_image_obj.close( );
       }
       [18] catch(Exception e){
            System.out.println("exception raised " + e);
            System.out.println("Signature similarity computation failed");
       }
}
```

1. 次の値を持つ 6 つの整数を作成します。
   - IMG_ID1 は、ORDIMAGETAB の OrdImage オブジェクトの主キーで、1 つ目のシグネチャが生成されます。
   - SIG_ID1 は、SIGTABLE の OrdImageSignature オブジェクトの主キーで、1 つ目のシグネチャが格納されます。
   - IMG_ID2 は、ORDIMAGETAB の OrdImage オブジェクトの主キーで、2 つ目のシグネチャが生成されます。
   - SIG_ID2 は、SIGTABLE の OrdImageSignature オブジェクトの主キーで、2 つ目のシグネチャが格納されます。
   - IMG_COL は、ORDIMAGETAB の OrdImage オブジェクトの列番号です。
   - SIG_COL は、SIGTABLE の OrdImageSignature オブジェクトの列番号です。
2. イメージが一致するかどうかを判断するしきい値を設定します。
3. 2 つの Statement オブジェクトを作成します。

4. 所定の SQL 問合せを実行し、その結果を 4 つのローカル OracleResultSet オブジェクトに格納します。この場合、SQL 問合せは次の位置にあるデータを選択します。
   - ORDIMAGETAB の主キーが 10 であるデータベース行
   - SIGTABLE の主キーが 10 であるデータベース行
   - ORDIMAGETAB の主キーが 11 であるデータベース行
   - SIGTABLE の主キーが 11 であるデータベース行
5. 結果セットの適切な行に進みます。
6. ローカル OrdImage オブジェクトとして img_obj1 を作成します。ORDIMAGETAB の主キーが 10 である行の 2 列目にある OrdImage オブジェクトのコンテンツを、img_obj1 に移入します。
7. ローカル OrdImageSignature オブジェクトとして img_sig1 を作成します。SIGTABLE の主キーが 10 である行の 2 列目にある OrdImageSignature オブジェクトのコンテンツを、img_sig1 に移入します。
8. setProperties( ) をコールし、メディア・データからプロパティ値を抽出し、その値を img_obj1 に設定します。抽出および設定されたプロパティ値については、第 5 章の「setProperties( )」を参照してください。
9. データを img_obj1 にインポートします。
10. img_obj1 のシグネチャを生成して、img_sig1 に格納します。
11. ローカル OrdImage オブジェクトとして img_obj2 を作成します。ORDIMAGETAB の主キーが 11 である行の 2 列目にある OrdImage オブジェクトのコンテンツを、img_obj2 に移入します。
12. ローカル OrdImageSignature オブジェクトとして img_sig2 を作成します。SIGTABLE の主キーが 11 である行の 2 列目にある OrdImageSignature オブジェクトのコンテンツを、img_sig2 に移入します。
13. setProperties( ) をコールし、メディア・データからプロパティ値を抽出し、その値を img_obj2 に設定します。抽出および設定されたプロパティ値については、第 5 章の「setProperties( )」を参照してください。
14. データを img_obj2 にインポートします。
15. img_obj2 のシグネチャを生成して、img_sig2 に格納します。
16. 色、テクスチャ、形および位置属性に基づいて、2 つのイメージ・シグネチャが一致するかどうかを判断します。スコアが 20 以下の場合、結果は 1 となり、画面に適切なメッセージを表示します。21 以上の場合は、結果は 0 となり、画面に適切なメッセージを表示します。
17. Statement および OracleResultSet オブジェクトを明示的にクローズします。
18. メソッドによって発生した例外またはエラーを処理します。

2.4 OrdVideo の例

ビデオの例（VideoExample.sql および VideoExample.java）では、SQL、JDBC および *inter*Media Java Classes API を使用して次の操作を行う、ユーザー定義メソッドについて説明します。

- テスト・コンテンツを含むデータベース・サーバー表を作成します。
- ローカル・ファイルからアプリケーションおよびデータベース内の OrdVideo オブジェクトにデータをロードします。
- ローカル・ストリームからアプリケーションおよびデータベース内の OrdVideo オブジェクトにデータをロードします。
- ローカル・バイト配列からアプリケーションおよびデータベース内の OrdVideo オブジェクトにデータをロードします。
- アプリケーション内の OrdVideo オブジェクトからプロパティを抽出し、出力します。
- データベース・メソッドへの違反コールを使用して、エラー処理を示します。

2.4.1 VideoExample.sql

例 2-32 に VideoExample.sql の内容を示します。

例 2-32 VideoExample.sql の内容

```
set echo on

--PLEASE change system password
connect system/manager
drop user VIDEOUSER cascade;
[1] create user VIDEOUSER identified by VIDEOUSER ;
grant connect,resource to VIDEOUSER identified by VIDEOUSER;

[2] connect VIDEOUSER/VIDEOUSER

[3] CREATE TABLE TVID(n NUMBER, vid ORDSYS.ORDVIDEO);

-- Note - the OrdVideo.init method was added in interMedia 8.1.7.
-- If you are running against an older release of interMedia and the
-- Oracle database, you will have to modify the following INSERT statements
-- to use the OrdVideo default constructor.

[4] INSERT INTO TVID VALUES(1, ORDSYS.ORDVideo.init( ));
INSERT INTO TVID VALUES(2, ORDSYS.ORDVideo.init( ));
INSERT INTO TVID VALUES(3, ORDSYS.ORDVideo.init( ));
commit;
/
```

VideoExample.sql の SQL 文は、次の操作を行います。

1. ユーザー VIDEOUSER を作成し、必要な権限を付与します。
2. VIDEOUSER でデータベース・サーバーに接続します。
3. 表 TVID を作成します。TVID には、2 つの列（数値、OrdVideo オブジェクト）があります。
4. 表に、空の OrdVideo オブジェクトを含む行を 3 行追加します。

init メソッドの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

2.4.2 VideoExample.java

2.4.2.1 項～ 2.4.2.8 項では、VideoExample.java サンプル・ファイルに含まれるメソッドを示します。

2.4.2.1 main( ) メソッド

例 2-33 に、main( ) メソッドを示します。

例 2-33 main( ) メソッド（Video）

```
public static void main (String args[ ]){
     byte[ ] ctx = new byte[4000];
     OracleConnection con = null;
     try {
          VideoExample tk = new VideoExample( );
          [1] con = tk.connect( );
          //Include the following line only if you are running
          //an Oracle 8.1.7 database or later.
          //If you are running a database server prior to 8.1.7,
          //the call will fail.
          [2] OrdMediaUtil.imCompatibilityInit(con);
          [3] tk.loadDataFromFile(con);
          tk.extractProperties(con);
          tk.printProperties(con);
          tk.loadDataFromStream(con);
          tk.otherMethods(con);
          tk.loadDataFromByteArray(con);
          [4] con.commit( );
          [5] con.close( );
          System.out.println("Done.");
     }
```

```
     [6] catch (Exception e) {
          try {
               System.out.println("Exception : " + e);
               con.close( );
          }
          catch(Exception ex) {
               System.out.println("Close Connection Exception : " + ex);
          }
     }
}
```

main( ) メソッドは、次の操作を行います。

1. connect( ) メソッドを使用して、データベース表への接続を確立します。
2. ご使用のアプリケーションと新しいリリースの Oracle データベースの互換性を保証します。詳細は、1.8 項を参照してください。
3. データベース・サーバーおよびローカル・マシン上のオブジェクトを操作する、複数のメソッド（VideoExample.java で定義済）をコールします。
4. データベース表への変更をすべてコミットします。
5. データベースへの接続をクローズします。
6. コードによって発生したエラーまたは例外を処理します。

2.4.2.2 項～ 2.4.2.8 項では、main( ) メソッドからコールされたメソッドについて、VideoExample.java に記述された順序ではなく、それがコールされた順序で説明します。

2.4.2.2 connect( ) メソッド

例 2-34 に、ユーザー定義メソッド connect( ) を示します。このメソッドでは、アプリケーションからデータベースへの接続を確立します。

例 2-34 connect( ) メソッド（Video）

```
public OracleConnection connect( ) throws Exception{
     String connectString;
     [1] Class.forName ("oracle.jdbc.driver.OracleDriver");
     [2] connectString = "jdbc:oracle:oci8:@";
     [3] OracleConnection con = (OracleConnection)
          DriverManager.getConnection(connectString,"VIDEOUSER","VIDEOUSER");
     [4] con.setAutoCommit(false);
     return con;
}
```

connect( ) メソッドは、次の操作を行います。

1. Oracle データベースでは、JDK 準拠の JVM を使用するため、JDBC ドライバを直接ロードします。
2. 接続するデータベースの URL を含む文字列を定義します。ご使用のデータベースにあわせて、この文字列を変更してください。
3. connectString に指定された URL、ユーザー名 VIDEOUSER およびパスワード VIDEOUSER を使用してデータベースへの接続を設定します。ユーザー名およびパスワードは、VideoExample.sql で作成済です。
4. 自動コミット・モードを使用禁止にします。これは、commit( ) メソッドまたは rollback( ) メソッドを使用して、それぞれ手動でコミットまたはロールバックする必要があることを意味します。

2.4.2.3 loadDataFromFile( ) メソッド

例 2-35 に、ユーザー定義メソッド loadDataFromFile( ) を示します。このメソッドでは、*inter*Media の loadDataFromFile( ) メソッドを使用して、メディア・データをアプリケーション・オブジェクトにロードします。

例 2-35 loadDataFromFile( ) メソッド（Video）

```
public void loadDataFromFile(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          [2] OracleResultSet rs = (OracleResultSet)s.executeQuery
               ("select * from TVID where n = 1 for update ");
          int index = 0;
          [3] while(rs.next( )){
               [4] index = rs.getInt(1);
               [5] OrdVideo vidObj = (OrdVideo) rs.getCustomDatum(2,
                    OrdVideo.getFactory( ));
               [6] vidObj.loadDataFromFile("testvid.dat");
               [7] vidObj.getDataInFile("output1.dat");
               System.out.println("***********AFTER getDataInFile ");
               [8] System.out.println("getContentLength output : " +
                    vidObj.getContentLength( ));
               [9] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update tvid set vid = ?
                    where n = " + index);
               stmt1.setCustomDatum(1,vidObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          System.out.println("loading successful");
     }
```

```
      [10] catch(Exception e) {
           System.out.println("exception raised " + e);
           System.out.println("loading unsuccessful");
      }
}
```

loadDataFromFile( ) メソッドは、次の操作を行います。

1. OracleStatement オブジェクトを作成します。
2. 所定の SQL 問合せを実行し、その結果をローカル OracleResultSet オブジェクトに格納します。この場合、SQL 問合せは、n=1 の条件を満たすデータベース行のデータを選択します。
3. OracleResultSet に処理されていない結果がある場合、while ループ内の操作を行います。この場合、OracleResultSet に含まれる行は 1 行のみのため、ループ内の操作は一度のみ実行されます。
4. 変数 index に、OracleResultSet の 1 行目の 1 列目にある整数値（この場合、1）を設定します。
5. ローカル OrdVideo オブジェクト vidObj を作成します。OracleResultSet の現行の行の 2 列目にある OrdVideo オブジェクトのコンテンツを、vidObj に移入します。
6. loadDataFromFile( ) メソッドを使用して、testvid.dat にあるメディア・データをデータベース内の OrdVideo オブジェクトおよび vidObj にロードします。これによって、ローカル・フィールドが vidObj に設定されますが、データベース・オブジェクトには設定されません。
7. getDataInFile( ) メソッドを使用して、アプリケーション内の OrdVideo オブジェクトからメディア・データを取得し、ローカル・システム上のファイル output1.dat にロードします。
8. ロードが成功したことを確認するため、vidObj のコンテンツ長を取得し、画面に出力します。
9. データベース内の OrdVideo オブジェクトを、vidObj のコンテンツで更新する SQL 文を作成および実行します。この更新によって、アプリケーション・オブジェクトに一致するように、データベース・オブジェクトのローカル属性が設定されます。
10. コードによって発生したエラーまたは例外を処理します。

2.4.2.4 extractProperties( ) メソッド

例 2-36 に、ユーザー定義メソッド extractProperties( ) を示します。このメソッドでは、アプリケーション・オブジェクトにプロパティを設定します。

例 2-36 extractProperties( ) Method メソッド（Video）

```
public void extractProperties(OracleConnection con){
     byte[ ] ctx[ ] = new byte [4000][1];
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery
               ("select * from TVID where n = 1 for update");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdVideo vidObj = (OrdVideo) rs.getCustomDatum(2,
                    OrdVideo.getFactory( ));
               [2] vidObj.setProperties(ctx);
               System.out.println("set Properties called");
               [3] if(vidObj.checkProperties(ctx)) {
                    System.out.println("checkProperties called");
                    System.out.println("setBindParams successful");
                    System.out.println("setProperties successful");
                    System.out.println("checkProperties successful");
                    System.out.println("extraction  successful");
               }
               else {
                    System.out.println("checkProperties called");
                    System.out.println("extraction not successful");
                    System.out.println("checkProperties successful");
               }
               [4] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update tvid set vid = ? where
                    n = " + index);
               stmt1.setCustomDatum(1,vidObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          rs.close( );
          s.close( );
     }
     [5] catch(Exception e) {
          System.out.println("exception raised " + e);
          System.out.println("extract prop unsuccessful");
     }
}
```

extractProperties( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdVideo オブジェクトとして vidObj を作成し、例 2-35 の手順 1 ～ 5 と同様の処理で vidObj にメディア・データを移入します。このメソッドでは、n=1 の条件を満たす行のコンテンツを操作しています。
2. setProperties( ) をコールし、メディア・データからプロパティ値を抽出し、その値をアプリケーション OrdVideo オブジェクトに設定します。抽出および設定されたプロパティ値については、第 8 章の「setProperties(byte[ ][ ])」を参照してください。
3. checkProperties( ) をコールして、アプリケーション・オブジェクトのプロパティ値とメディア・データの値を比較します。値がすべて同じである場合、checkProperties( ) は true を戻し、画面に適切なメッセージを表示します。値が異なる場合は false を戻し、画面に適切なメッセージを表示します。
4. データベース内の OrdVideo オブジェクトを、vidObj のコンテンツで更新する SQL 文を作成および実行します。
5. コードによって発生したエラーまたは例外を処理します。

2.4.2.5 printProperties( ) メソッド

例 2-37 に、ユーザー定義メソッド printProperties( ) を示します。このメソッドでは、アプリケーション・オブジェクトの属性を画面に出力します。

例 2-37 printProperties( ) メソッド（Video）

```
public void printProperties(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet)s.executeQuery
               ("select * from TVID where n = 1 ");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdVideo vidObj = (OrdVideo) rs.getCustomDatum(2,
                    OrdVideo.getFactory( ));
               [2] System.out.println("format: " + vidObj.getFormat( ));
               System.out.println("mimetype: " + vidObj.getMimeType( ));
               System.out.println("width: " + vidObj.getWidth( ));
               System.out.println("height: " + vidObj.getHeight( ));
               System.out.println("frame resolution: " +
                    vidObj.getFrameResolution( ));
               System.out.println("frame rate: " + vidObj.getFrameRate( ));
               System.out.println("video duration: " +
                    vidObj.getVideoDuration( ));
               System.out.println("number of frames: " +
                    vidObj.getNumberOfFrames( ));
```

```
                System.out.println("description : " +
                     vidObj.getDescription( ));
                System.out.println("compression type: " +
                     vidObj.getCompressionType( ));
                System.out.println("bit rate: " + vidObj.getBitRate( ));
                System.out.println("num of colors: " +
                     vidObj.getNumberOfColors( ));
           }
     }
     [3] catch(Exception e) {
          System.out.println("exception raised " + e);
          System.out.println("print proerties unsuccessful");
     }
}
```

printProperties( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdVideo オブジェクトとして vidObj を作成し、例 2-35 の手順 1 ～ 5 と同様の処理で vidObj にメディア・データを移入します。このメソッドでは、n=1 の条件を満たす行のコンテンツを操作しています。
2. vidObj のプロパティの値を取得し、その値を画面に出力します。
3. コードによって発生したエラーまたは例外を処理します。

2.4.2.6 loadDataFromStream( ) メソッド

例 2-38 に、ユーザー定義メソッド loadDataFromStream( ) を示します。このメソッドでは、*inter*Media の loadDataFromInputStream( ) メソッドを使用して、メディア・データをアプリケーション・オブジェクトにロードします。

例 2-38 loadDataFromStream( ) メソッド（Video）

```
public void loadDataFromStream(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TVID where n = 2 for update ");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdVideo vidObj = (OrdVideo) rs.getCustomDatum(2,
                    OrdVideo.getFactory( ));
               [2] FileInputStream  fStream  = new FileInputStream
                    ("testvid.dat");
               [3] vidObj.loadDataFromInputStream(fStream);
               [4] vidObj.getDataInFile("output2.dat");
```

```
                [5] fStream.close( );
                System.out.println("************AFTER getDataInFile ");
                [6] System.out.println("getContentLength output : " +
                     vidObj.getContentLength( ));
                [7] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                     con.prepareCall("update tvid set vid = ?
                     where n = " + index);
                stmt1.setCustomDatum(1,vidObj);
                stmt1.execute( );
                stmt1.close( ) ;
           }
           System.out.println("load data from stream successful");
      }
      [8] catch(Exception e) {
           System.out.println("exception raised " + e);
           System.out.println("load data from stream unsuccessful");
      }
}
```

loadDataFromStream( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdVideo オブジェクトとして vidObj を作成し、例 2-35 の手順 1 ～ 5 と同様の処理で vidObj にメディア・データを移入します。このメソッドでは、n=2 の条件を満たす行のコンテンツを操作しています。
2. 新規の FileInputStream オブジェクトを作成します。この入力ストリームには、ローカル・ファイル testvid.dat のコンテンツが含まれています。
3. loadDataFromInputStream( ) メソッドを使用して、入力ストリームにあるメディア・データをデータベース内の OrdVideo オブジェクトおよび vidObj にロードします。これによって、ローカル・フィールドが vidObj に設定されますが、データベース・オブジェクトには設定されません。
4. getDataInFile( ) メソッドを使用して、アプリケーション内の OrdVideo オブジェクトからメディア・データを取得し、ローカル・システム上のファイル output2.dat にロードします。
5. ローカル入力ストリームをクローズします。
6. ロードが成功したことを確認するため、vidObj のコンテンツ長を取得し、画面に出力します。
7. データベース内の OrdVideo オブジェクトを、vidObj のコンテンツで更新する SQL 文を作成および実行します。この更新によって、アプリケーション・オブジェクトに一致するようにデータベース・オブジェクトの属性が設定されます。
8. コードによって発生したエラーまたは例外を処理します。

2.4.2.7 otherMethods( ) メソッド

例 2-39 に、ユーザー定義メソッド otherMethods( ) を示します。このメソッドでは、processSourceCommand( ) メソッドを使用します。

例 2-39 otherMethods( ) メソッド（Video）

```
public void otherMethods(OracleConnection con){
     byte[ ] ctx[ ] = {new byte[4000]};
     byte[ ] res[ ] = {new byte[20]};
     [1] int suc = 1;
     try {
          [2] Statement s1 = con.createStatement( );
          OracleResultSet rs1 = (OracleResultSet)
          s1.executeQuery("select * from TVID where n = 1 for
               update ");
          int index1 = 0;
          while(rs1.next( )) {
               index1 = rs1.getInt(1);
               OrdVideo vidObj = (OrdVideo) rs1.getCustomDatum(2,
                    OrdVideo.getFactory( ));
               [3] try {
                    byte[ ] pSRes = vidObj.processSourceCommand(ctx,
                         "", "", res);
                    suc = 0;
               }
               [4] catch (Exception e) {
                    System.out.println("Expected Exception raised in
                         processSourceCommand(...)" );
               }
               [5] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                    con.prepareCall("update tvid set vid = ? where
                         n = " + index1);
               stmt1.setCustomDatum(1,vidObj);
               stmt1.execute( );
               stmt1.close( ) ;
          }
          rs1.close( );
          s1.close( );
     }
     [6] catch(Exception e){
          System.out.println("Exception raised " );
     }
     [7] if(suc == 1)
          System.out.println("other methods successful");
     else
          System.out.println("other methods unsuccessful");
}
```

otherMethods( ) メソッドは、次の操作を行います。

1. メソッドの成功または失敗を示すために使用する整数を作成し、これを 1（成功）に初期化します。
2. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdVideo オブジェクトとして vidObj を作成し、例 2-35 の手順 1 ～ 5 と同様の処理で vidObj にメディア・データを移入します。このメソッドでは、n=1 の条件を満たす行のコンテンツを操作しています。
3. サーバー側でコールされるコマンドを指定せずに processSourceCommand( ) をコールします。これによって例外が発生します。これは、processSourceCommand( ) コールの次のコードは実行されず、catch ループのコードが実行されることを意味します。例外が発生しない場合、メソッドは失敗し、成功インジケータが 0（失敗）に設定されます。
4. 手順 3 で発生した例外を出力します。
5. データベース内の OrdVideo オブジェクトを、vidObj のコンテンツで更新する SQL 文を作成および実行します。
6. コードによって発生した、予期しないエラーまたは例外を処理します。
7. メソッドの結果に基づいて、適切なメッセージを画面に出力します。

2.4.2.8 loadDataFromByteArray( ) メソッド

例 2-40 に、ユーザー定義メソッド loadDataFromByteArray( ) を示します。このメソッドでは、*inter*Media の loadDataFromByteArray( ) メソッドを使用して、メディア・データをアプリケーション・オブジェクトにロードします。

例 2-40 loadDataFromByteArray( ) メソッド（Video）

```
public void loadDataFromByteArray(OracleConnection con){
     try {
          [1] Statement s = con.createStatement( );
          OracleResultSet rs = (OracleResultSet) s.executeQuery
               ("select * from TVID where n = 3 for update ");
          int index = 0;
          while(rs.next( )){
               index = rs.getInt(1);
               OrdVideo vidObj = (OrdVideo) rs.getCustomDatum(2,
                    OrdVideo.getFactory( ));
               [2] File ff = new File("testvid.dat");
               int fileLength = (int) ff.length( );
               byte[ ] data = new byte[fileLength];
               [3] FileInputStream  fStream  = new
                    FileInputStream("testvid.dat");
               [4] fStream.read(data,0,fileLength);
               [5] vidObj.loadDataFromByteArray(data);
```

```
                    [6] fStream.close( );
                    [7] vidObj.getDataInFile("output3.dat");
                    [8] byte[ ] resArr = vidObj.getDataInByteArray( );
                    [9] System.out.println("byte array length : " +
                         resArr.length);
                    [10] FileOutputStream outStream = new FileOutputStream
                         ("output4.dat");
                    [11] outStream.write(resArr);
                    [12] outStream.close( );
                    [13] InputStream inpStream = vidObj.getDataInStream( );
                    int length = 32768;
                    byte[ ] tempBuffer = new byte[32768];
                    [14] int numRead = inpStream.read(tempBuffer,0,length);
                    try {
                         [15] outStream = new FileOutputStream("output5.dat");
                         [16] while(numRead != -1) {
                              [17] if (numRead < 32768) {
                                   length = numRead;
                                   outStream.write(tempBuffer,0,length);
                                   break;
                              }
                              [18] else
                                   outStream.write(tempBuffer,0,length);
                              [19] numRead = inpStream.read(tempBuffer,0,length);
                         }
                    }
                    [20] finally {
                         outStream.close( );
                         inpStream.close( );
                    }
                    System.out.println("************AFTER getDataInFile ");
                    [21] System.out.println(" getContentLength output : " +
                         vidObj.getContentLength( ));
                    [22] OraclePreparedStatement stmt1 = (OraclePreparedStatement)
                         con.prepareCall("update tvid set vid = ? where
                         n = " + index);
                    stmt1.setCustomDatum(1,vidObj);
                    stmt1.execute( );
                    stmt1.close( ) ;
               }
          }
          [23] catch(Exception e) {
               System.out.println("exception raised " + e);
               System.out.println("loadData from byte array unsuccessful");
          }
     }
```

loadDataFromByteArray( ) メソッドは、次の操作を行います。

1. Statement オブジェクト、ローカル OracleResultSet オブジェクトおよびローカル OrdVideo オブジェクトとして vidObj を作成し、例 2-35 の手順 1 ～ 5 と同様の処理で vidObj にメディア・データを移入します。このメソッドでは、n=3 の条件を満たす行のコンテンツを操作しています。
2. ローカル・ファイル testvid.dat のサイズ（バイト単位）を決定し、同じサイズのバイト配列を作成します。
3. 新規の FileInputStream オブジェクトを作成します。この入力ストリームには、testvid.dat のコンテンツが含まれています。
4. 入力ストリームのコンテンツをバイト配列に読み込みます。
5. loadDataFromByteArray( ) メソッドを使用して、バイト配列にあるメディア・データをデータベース内の OrdVideo オブジェクトおよび vidObj にロードします。これによって、ローカル・フィールドが vidObj に設定されますが、データベース・オブジェクトには設定されません。
6. 入力ストリームをクローズします。
7. getDataInFile( ) メソッドを使用して、アプリケーション内の OrdVideo オブジェクトからメディア・データを取得し、ローカル・システム上のファイル output3.dat にロードします。
8. getDataInByteArray( ) メソッドを使用して、アプリケーション内の OrdVideo オブジェクトからメディア・データを取得し、バイト配列 resArr にロードします。
9. ロードが成功したことを確認するため、resArr の長さを取得し、画面に出力します。
10. 新規の FileOutputStream オブジェクト outStream を作成します。この出力ストリームは、ローカル・ファイル output4.dat にデータを書き込みます。
11. resArr のコンテンツを output4.dat に書き込みます。
12. 出力ストリームをクローズします。
13. 新規の入力ストリーム inpStream を作成します。getDataInStream( ) メソッドを使用して、アプリケーション内の OrdVideo オブジェクトからメディア・データを取得し、inpStream に格納します。
14. inpStream の先頭（オフセット 0）から 32768 バイトをバイト配列 tempBuffer に読み込みます。整数 numRead には、読込みバイトの合計数、または入力ストリームが最後まで到達した場合は -1 が設定されます。ロードが成功した場合、numRead は 32768 となります。
15. OutStream を再オープンします。この場合、ローカル・ファイル output5.dat にデータを書き込みます。
16. numRead が -1 でない場合は、while ループ内の操作を実行します。プログラムは、このループを実行します。

17. numRead が 32768 未満（データが最後まで読み込まれた）の場合は、tempBuffer に読み込まれたバイト数を、outStream に書き込みます。
18. numRead が 32768 の場合、outStream に 32768 バイトを書き込みます。
19. 入力ストリームからさらにデータをバイト配列に読み込みます。データが最後まで読み込まれた場合、numRead は -1 に設定され、プログラムはループを終了します。入力ストリームに読み込まれていないデータがある場合は、バイト配列に読み込まれ、手順 17、18 が繰り返されます。
20. while ループの終了後、入力ストリームおよび出力ストリームをクローズします。
21. ロードが成功したことを確認するため、vidObj のコンテンツ長を取得し、画面に出力します。
22. データベース内の OrdVideo オブジェクトを、vidObj のコンテンツで更新する SQL 文を作成および実行します。この更新によって、アプリケーション・オブジェクトに一致するようにデータベース・オブジェクトの属性が設定されます。
23. コードによって発生したエラーまたは例外を処理します。

3

OrdAudio リファレンス情報

OrdAudio クラスは、Java アプリケーションで ORDSYS.ORDAudio データベース型のインスタンスを表すために使用されます。OrdAudio クラスには、OrdAudio Java オブジェクトに対して様々な操作を行う一連のメソッドおよび様々なオブジェクト属性を取得し、設定するための一連のメソッドがあります。

ほぼすべてのメソッドで、アプリケーション内の OrdAudio Java オブジェクトの属性が操作されます。読込みまたは書込みを目的としてオーディオ・データにアクセスするメソッドは例外です。例外のメソッドには、次のものがあります。

- localData 属性で指定されたデータベース BLOB を操作し、データベース BLOB 内に格納されたデータの読込みおよび書込みを行うメソッド
- srcType 属性が file の場合に srcLocation および srcName 属性で指定されたデータベース BFILE を操作し、データベース・サーバー内の指定されたファイルからデータを読み込むメソッド
- srcType 属性が http の場合に srcType、srcLocation および srcName 属性で指定された URL を操作し、指定された URL でリソースからデータを読み込むメソッド

アプリケーションによって、データベース内の OrdAudio Java オブジェクトまたはオーディオ・データが変更される場合は、これらの変更を永続的なものにするために、データベース内の ORDAudio SQL オブジェクトを更新する必要があります。

OrdAudio Java クラスの一部のメソッドは、データベース・ソース・プラグインまたはデータベース・フォーマット・プラグインに渡されて処理され、コンテキスト・パラメータに `byte [ ] [ ] ctx` を取ります。ソース・プラグインまたはフォーマット・プラグインで必要になる可能性のあるコンテキスト情報を保持するには、アプリケーションで 64 バイトの配列を割り当てる必要があります。たとえば、プラグインでは、一度のコールでコンテキスト情報を初期化し、その情報を後続のコールで使用できます。配列は、ソース・プラグインのコンテキストおよびフォーマット・プラグインのコンテキストに 1 つずつ必要です。ほとんどのプラグインは、64 バイトで十分ですが、ユーザー定義のプラグインによっては、追加領域が必要になる場合もあります。次の例では、プラグインのコンテキスト情報配列の割当て方法を示します。

```
byte [] [] ctx = new byte[1][64];
```

注意： 今回のリリースでは、オラクル社が提供するソース・プラグインまたはフォーマット・プラグインでは、コンテキストは保持されません。また、ユーザー定義のソース・プラグインまたはフォーマット・プラグインでも、コンテキストが保持されない場合もあります。ただし、前述した方法でコンテキスト・パラメータを指定すると、アプリケーションは現行または今後のソース・プラグインまたはフォーマット・プラグインで動作します。

プラグインの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

3.1 前提条件

*inter*Media メソッドを実行するために、次のインポート文を Java ファイルに組み込む必要があります。

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
import oracle.ord.im.*;
```

この章で示す例は、次の操作がすでに実行済であることを前提にしています。

- OrdAudio 型の列を含む表への接続が確立されている。
- ローカル OrdAudio オブジェクトとして audObj が作成され、データが移入されている。

接続の確立またはローカル・オブジェクトの移入の例については、2.1.2 項を参照してください。

3.2 リファレンス情報

この項では、OrdAudio オブジェクトを操作するメソッドについて説明します。

checkProperties( )

構文

```
public boolean checkProperties(byte[ ] [ ] ctx )
```

説明

オーディオ・データのプロパティが、OrdAudio Java オブジェクトの属性と一貫性があるかどうかを確認します。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

戻り値

オーディオ・データのプロパティが、OrdAudio Java オブジェクトの属性と一貫性がある場合は true、一貫性がない場合は false を戻します。

例外

java.sql.SQLException

データベース内で対応する checkProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
if(audObj.checkProperties(ctx))
     System.out.println("checkProperties successful");
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。

clearLocal( )

構文

```
public void clearLocal( )
```

説明

ローカル属性を消去して、オーディオ・データが外部に格納されていることを示します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

ローカル属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.clearLocal( )
```

closeSource( )

構文

```
public int closeSource(byte[ ] [ ] ctx)
```

説明

データ・ソースをクローズします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

戻り値

ステータスを、整数で戻します。0（ゼロ）は成功を示し、0（ゼロ）以外の値はソース・プラグイン固有の障害コードを示します。

例外

java.sql.SQLException

データベース内で対応する closeSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
int i = audObj.closeSource(ctx);
if(i == 0)
     System.out.println("Source close successful");
else
     System.out.println("Source close unsuccessful");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。

deleteContent( )

構文

```
public void deleteContent( )
```

説明

localData 属性で指定されたデータベース BLOB に格納されているデータを削除します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する deleteContent( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.deleteContent( );
```

export( )

構文

```
public void export (byte[ ] [ ] ctx, String srcType,
  String srcLocation, String srcName)
```

説明

localData 属性で指定された BLOB からデータをエクスポートします。このメソッドは、データベース内の対応する export( ) メソッドをコールして、srcType、srcLocation および srcName の各パラメータで指定された位置に、オーディオ・データをエクスポートします。

ただし、export( ) メソッドをサポートしないソース・プラグインもあります。file ソース・タイプのみがネイティブにサポートされています。

このメソッドは、Oracle データベース・サーバーのリリース 8.1.7 以上で実行している場合のみ機能します。

後半部分では、export( ) メソッドおよびオラクル社が提供する file ソース・プラグインの使用方法について説明します。ユーザー定義のプラグインの動作は異なります。

file ソース・プラグインによって実装された export( ) メソッドは、localData 属性で指定されたデータベース BLOB 内に格納されているオーディオ・データのコピーのみ行い、変更は行いません。

オーディオ・データのエクスポート後も、すべてのオーディオ・プロパティ属性は変更されません。ただし、srcType、srcLocation および srcName 属性は、export( ) メソッドに渡された srcType、srcLocation および srcName の各パラメータの値で更新されます。export( ) メソッドへのコール後、データベース内部のオーディオ・データを管理する必要がない場合は、clearLocal( ) メソッドをコールして、オーディオ・データがデータベースの外部に格納されていることを示し、deleteContent( ) メソッドをコールして、データベース BLOB に格納されているオーディオ・データを削除します。

データベース内の export( ) メソッドは、ユーザーがアクセス権限を所有しているデータベース・ディレクトリ・オブジェクトに対してのみ書込みを行います。つまり、SQL の CREATE DIRECTORY 文を使用して作成したディレクトリ、または読込み権限を付与されたディレクトリにアクセスできます。CREATE DIRECTORY 文を実行するには、CREATE ANY DIRECTORY 権限が必要です。また、書込み可能なファイルを指定するには、DBMS_JAVA.GRANT_PERMISSION メソッドを使用する必要があります。

たとえば、次の SQL*Plus コマンドは、ユーザー MEDIAUSER にファイル filmtrack1.au の書込み権限を付与します。

```
CALL DBMS_JAVA.GRANT.PERMISSION(
    'MEDIAUSER',
    'java.io.FilePermission',
    '/audio/movies/filmtrack1.au',
    'write');
```

前述の例は、単一ファイルへの書込み権限を許可する方法を示しています。この他にも、ワイルド・カードを使用した、複数のディレクトリおよびファイル名への書込み権限を許可する様々なパス指定があります。たとえば、スラッシュとアスタリスク（/*）（スラッシュは、オペレーティング・システム固有のファイル・セパレータ文字）で終わるパス指定は、指定されたディレクトリに含まれているすべてのファイルを示します。スラッシュとハイフン（/-）で終わるパス指定は、指定されたディレクトリおよびそのすべてのサブディレクトリに含まれているすべてのファイルを示します。特別なトークン <<ALL FILES>> で構成されたパス名は、すべてのファイルへのアクセス権限を許可します。

セキュリティおよびパフォーマンスの詳細は、『Oracle9*i* Java Developer's Guide』および Java API の java.io.FilePermission クラスを参照してください。必要な権限の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

srcType
コンテンツのエクスポート先のソース・タイプ

srcLocation
コンテンツのエクスポート先のソースの位置

srcName
コンテンツのエクスポート先のソース名

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する export( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
audObj.export(ctx,"file","AUDIODIR","complete.wav");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- file: コンテンツのエクスポートに使用するソース・プラグインです。
- AUDIODIR: コンテンツのエクスポート先の位置です。
- complete.wav: コンテンツのエクスポート先のファイルです。

getAllAttributes( )

構文

```
public CLOB getAllAttributes(byte[ ] [ ] ctx)
```

説明

一時 CLOB 内のオーディオ・プロパティの値を、フォーマット・プラグインで定義された形式で戻します。ネイティブにサポートされているフォーマットの場合、これらの情報は、カンマで区切られた属性のリストとして表示されます。属性のリストは、attributeName=attributeValue という形式で示されます。このリストには、format、mimeType、encoding、numberOfChannels、samplingRate、sampleSize、compressionType および audioDuration 属性が含まれます。ユーザーが定義したフォーマットの場合、これらの情報は、フォーマット・プラグインで定義された形式で表示されます。

> **注意：** アプリケーションでは、一時 CLOB 内に含まれている情報を読み込んだ後、一時 CLOB を解放する必要があります。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

戻り値

属性の値を、一時 oracle.sql.CLOB で戻します。

例外

java.sql.SQLException

データベース内で対応する getAllAttributes( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] ctx[ ] = new byte[1][64];
CLOB attributes = audObj.getAllAttributes(ctx);
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。

getAttribute( )

構文

```
public String getAttribute(byte[ ] [ ] ctx, String name)
```

説明

要求されたオーディオ・プロパティの値を戻します。このメソッドは、ユーザー定義のフォーマット・プラグインで、OrdAudio Java オブジェクトの属性として使用できないオーディオ・プロパティの値を戻すために使用します。このメソッドは、オラクル社が提供するフォーマット・プラグインでは実装されません。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

name
属性名のプロパティ

戻り値

属性の値を、文字列で戻します。

例外

java.sql.SQLException

データベース内で対応する getAttribute( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
int attribute = audObj.getAttribute(ctx,"numberOfChannels")
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。
- numberOfChannels: オブジェクトから取得する属性の値です。

getAudioDuration( )

構文

```
public int getAudioDuration( )
```

説明

audioDuration 属性の値を戻します。

パラメータ

なし

戻り値

audioDuration 属性の値を、整数で戻します。

例外

java.sql.SQLException

audioDuration 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int audioDuration = audObj.getAudioDuration( );
```

getBFILE( )

構文

```
public oracle.sql.BFILE getBFILE( )
```

説明

srcType 属性が file の場合、データベースから BFILE ロケータを戻します。このメソッドは、データベース内の対応する getBFILE( ) メソッド（srcLocation および srcName 属性を使用して BFILE を作成）をコールします。

パラメータ

なし

戻り値

oracle.sql.BFILE ロケータを戻します。

例外

java.sql.SQLException

データベース内で対応する getBFILE( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
BFILE audioBFILE = audObj.getBFILE( );
```

getComments( )

構文

```
public oracle.sql.CLOB getComments( )
```

説明

コメント属性から CLOB ロケータを戻します。

パラメータ

なし

戻り値

コメント属性の値を、oracle.sql.CLOB ロケータで戻します。

例外

java.sql.SQLException

コメント属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
CLOB comments = audObj.getComments( )
```

getCompressionType( )

構文

```
public String getCompressionType( )
```

説明

compressionType 属性の値を戻します。

パラメータ

なし

戻り値

compressionType 属性の値を、文字列で戻します。

例外

java.sql.SQLException

compressionType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String compressionType = audObj.getCompressionType( );
```

getContent( )

構文

```
public oracle.sql.BLOB getContent( )
```

説明

localData 属性から BLOB ロケータを戻します。

パラメータ

なし

戻り値

oracle.sql.BLOB ロケータを戻します。

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
BLOB localContent = audObj.getContent( );
```

getContentInLob( )

構文

```
public oracle.sql.BLOB getContentInLob(byte[ ] [ ] ctx,
  String mimeType[ ], String format[ ])
```

説明

データベース内の一時 BLOB の localData 属性で指定された BLOB からデータを戻します。このメソッドは、データベース内に一時 BLOB を作成し、localData 属性で指定された BLOB からデータを読み込み、一部 BLOB にデータを書き込み、一時 BLOB ロケータをコール元に戻します。

注意： アプリケーションでは、一時 BLOB 内に含まれている情報にアクセスした後、一時 BLOB を解放する必要があります。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

mimeType
mimeType 属性が要素 0 として書き込まれる文字列配列（長さは 1 要素）

format
format 属性が要素 0 として書き込まれる文字列配列（長さは 1 要素）

戻り値

一時 oracle.sql.BLOB ロケータのオーディオ・データを戻します。

例外

java.sql.SQLException

一時 BLOB の作成中、またはデータベース内で対応する getContentInLob( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
String mimetype[ ] = new String[1];
String format[ ] = new String[1];
BLOB localContent = audObj.getContentInLob(ctx,mimetype,format);
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。
- mimeType: 最初の値に MIME タイプを含む文字列の配列です。この値は、サーバーによって生成されます。
- format: 最初の値にデータ・フォーマットを含む文字列の配列です。この値は、サーバーによって生成されます。

getContentLength( )

構文

```
public int getContentLength( )
```

説明

オーディオ・データ長を戻します。このメソッドは、データベース内の対応する getContentLength( ) メソッドをコールします。

このメソッドをサポートしないソース・タイプもあります。たとえば、http ソース・タイプではサポートされません。

パラメータ

なし

戻り値

contentLength の値を、整数で戻します。

例外

java.sql.SQLException

データベース内で対応する getContentLength( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
int contentLength = audObj.getContentLength( );
```

getContentLength(byte[ ][ ])

構文

```
public int getContentLength(byte[ ] [ ] ctx)
```

説明

ソース・プラグインのコンテキスト情報を使用して、オーディオ・データ長を戻します。このメソッドは、データベース内の対応する getContentLength( ) メソッドをコールします。

このメソッドをサポートしないソース・タイプもあります。たとえば、http ソース・タイプではサポートされません。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

戻り値

contentLength 属性の値を、整数で戻します。

例外

java.sql.SQLException

データベース内で対応する getContentLength( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
int contentLength = audObj.getContentLength(ctx);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。

getDataInByteArray( )

構文

```
public byte[ ] getDataInByteArray( )
```

説明

localData 属性で指定されたデータベース BLOB からのデータを含むバイト配列を戻します。

パラメータ

なし

戻り値

データを含むバイト配列を戻します。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

BLOB からのデータの読込み中にエラーが発生した場合、この例外がスローされます。

java.lang.OutOfMemoryError

データの保持のために十分なメモリーを割当てできない場合、この例外がスローされます。

例

```
byte[ ] byteArr = audObj.getDataInByteArray( );
```

getDataInFile( )

構文

```
public boolean getDataInFile(String filename)
```

説明

localData 属性で指定されたデータベース BLOB からローカル・ファイルに、データを書き込みます。

パラメータ

filename
データの書込み先のファイル名

戻り値

データが正常にファイルに書き込まれた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

BLOB からのデータの読込み中または出力ファイルへのデータの書込み中にエラーが発生した場合、この例外がスローされます。

例

```
boolean load = audObj.getDataInFile("output1.dat");
if(load)
     System.out.println("getDataInFile completed successfully");
else
     System.out.println("Error in getDataInFile");
```

パラメータは次のとおりです。

- output1.dat: データの書込み先のファイルです。

getDataInStream( )

構文

```
public InputStream getDataInStream( )
```

説明

localData 属性で指定されたデータベース BLOB のデータの読込み元となる InputStream オブジェクトを戻します。

パラメータ

なし

戻り値

データの読込み元から InputStream オブジェクトを戻します。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
InputStream inpStream = audObj.getDataInStream( );
```

getDescription( )

構文

```
public String getDescription( )
```

説明

description 属性の値を戻します。

パラメータ

なし

戻り値

description 属性の値を、文字列で戻します。

例外

java.sql.SQLException

description 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String desc = audObj.getDescription( );
```

getEncoding( )

構文

```
public String getEncoding( )
```

説明

encoding 属性の値を戻します。

パラメータ

なし

戻り値

encoding 属性の値を、文字列で戻します。

例外

java.sql.SQLException

encoding 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String encoding = audObj.getEncoding( );
```

getFactory( )

構文

```
public static oracle.sql.CustomDatumFactory getFactory( )
```

説明

getCustomDatum( ) メソッドが使用する OrdAudio CustomDatumFactory インタフェースを戻します。OracleResultSet または OracleCallableStatement オブジェクトから OrdAudio オブジェクトを取り出す場合は、getFactory( ) メソッドを getCustomDatum( ) メソッドの factory パラメータとして指定します。

パラメータ

なし

戻り値

CustomDatumFactory インタフェースの OrdAudio 実装を戻します。

例外

なし

例

```
OrdAudio aud = (OrdAudio)rset.getCustomDatum( 1, OrdAudio.getFactory() );
```

getFormat( )

構文

```
public String getFormat( )
```

説明

format 属性の値を戻します。

パラメータ

なし

戻り値

format 属性の値を、文字列で戻します。

例外

java.sql.SQLException

format 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String format = audObj.getFormat( );
```

getMimeType( )

構文

```
public String getMimeType( )
```

説明

mimeType 属性の値を戻します。

パラメータ

なし

戻り値

mimeType 属性の値を、文字列で戻します。

例外

java.sql.SQLException

mimeType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String mimeType = audObj.getMimeType( );
```

getNumberOfChannels( )

構文

```
public int getNumberOfChannels( )
```

説明

numberOfChannels 属性の値を戻します。

パラメータ

なし

戻り値

numberOfChannels 属性の値を、整数で戻します。

例外

java.sql.SQLException

numberOfChannels 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int channels = audObj.getNumberOfChannels( );
```

getSampleSize( )

構文

```
public int getSampleSize( )
```

説明

sampleSize 属性の値を戻します。

パラメータ

なし

戻り値

sampleSize 属性の値を、整数で戻します。

例外

java.sql.SQLException

sampleSize 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int sampleSize = audObj.getSampleSize( );
```

getSamplingRate( )

構文

```
public int getSamplingRate( )
```

説明

samplingRate 属性の値を戻します。

パラメータ

なし

戻り値

samplingRate 属性の値を、整数で戻します。

例外

java.sql.SQLException

samplingRate 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int samplingRate = audObj.getSamplingRate( );
```

getSource( )

構文

```
public String getSource( )
```

説明

srcType://srcLocation/srcName という形式でソース情報を戻します。

パラメータ

なし

戻り値

ソース情報を、文字列で戻します。

例外

java.sql.SQLException

データベース内で対応する getSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
String source = audObj.getSource( );
```

getSourceLocation( )

構文

```
public String getSourceLocation( )
```

説明

srcLocation 属性の値を戻します。

パラメータ

なし

戻り値

srcLocation 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcLocation 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String location = audObj.getSourceLocation( );
```

getSourceName( )

構文

```
public String getSourceName( )
```

説明

srcName 属性の値を戻します。

パラメータ

なし

戻り値

srcName 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcName 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String name = audObj.getSourceName( );
```

getSourceType( )

構文

```
public String getSourceType( )
```

説明

srcType 属性の値を戻します。

パラメータ

なし

戻り値

srcType 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String type = audObj.getSourceType( );
```

getUpdateTime( )

構文

```
public java.sql.Timestamp getUpdateTime( )
```

説明

updateTime 属性の値を戻します。

パラメータ

なし

戻り値

updateTime 属性の値を、java.sql.Timestamp オブジェクトで戻します。

例外

java.sql.SQLException

upateTime 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
Timestamp time = audObj.getUpdateTime( );
```

importData( )

構文

```
public void importData(byte[ ] [ ] ctx)
```

説明

データを、外部ソースから localData 属性で指定されたデータベース BLOB にインポートします。外部データ・ソースは、srcType、srcLocation および srcName 属性で指定されます。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する import( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
audObj.importData(ctx);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。

importFrom( )

構文

```
public void importFrom(byte[ ] [ ] ctx, String srcType,
  String srcLocation, String srcName)
```

説明

データを、外部ソースから localData 属性で指定されたデータベース BLOB にインポートします。外部データ・ソースは、srcType、srcLocation および srcName パラメータで指定されます。srcType、srcLocation および srcName 属性は、importFrom( ) メソッドに渡された srcType、srcLocation および srcName の各パラメータの値で更新されます。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

srcType
データのインポート元のソース・タイプ

srcLocation
データのインポート元のソースの位置

srcName
データのインポート元のソース名

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する importFrom( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
audObj.importFrom("file","AUDIODIR","testaud.dat");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- file: データのインポートに使用するソース・プラグインです。
- AUDIODIR: データのインポート元のデータベース・サーバー上でのファイルの位置です。
- testaud.dat: データのインポート元のファイルです。

isLocal( )

構文

```
public boolean isLocal( )
```

説明

オーディオ・データが、localData 属性で指定されたデータベース内の BLOB でローカルに格納されているかどうかを示します。

パラメータ

なし

戻り値

データベース内の BLOB でデータがローカルに格納されている場合は true、格納されていない場合は false を戻します。

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
if(audObj.isLocal( ))
     System.out.println("local attribute is set to true");
else
     System.out.println("local attribute is set to false");
```

loadDataFromByteArray( )

構文

```
public boolean loadDataFromByteArray(byte[ ] byteArr)
```

説明

データを、バイト配列から localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

byteArr
データのロード元のバイト配列

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中またはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

バイト配列の読込み中にエラーが発生した場合、この例外がスローされます。

例

```
byte[ ] data = new byte[32000];
FileInputStream fStream = new FileInputStream("testaud.dat");
fStream.read(data,0,32000);
boolean success = audObj.loadDataFromByteArray(data);
if(success)
     System.out.println("loadDataFromByteArray was successful");
else
     System.out.println("loadDataFromByteArray was unsuccessful");
```

パラメータは次のとおりです。

- data: データのロード元のローカル・バイト配列です。
- testaud.dat: 32,000 バイトのデータを含むローカル・ファイルです。

loadDataFromFile( )

構文

```
public boolean loadDataFromFile(String filename)
```

説明

データを、ファイルから localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

filename
データのロード元のファイル名

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中またはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

データ・ファイルの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.loadDataFromFile("testaud.dat");
```

パラメータは次のとおりです。

- testaud.dat: オーディオ・データを含むローカル・ファイルです。

loadDataFromInputStream( )

構文

```
public boolean loadDataFromInputStream(InputStream inpStream)
```

説明

データを、InputStream オブジェクトから localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

inpStream
データのロード元の InputStream オブジェクト

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセスまたはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

InputStream オブジェクトの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
FileInputStream fStream = new FileInputStream("testaud.dat");
audObj.loadDataFromInputStream(fStream);
```

パラメータは次のとおりです。

- testaud.dat: オーディオ・データを含むローカル・ファイルです。
- fStream: オーディオ・データを OrdAudio オブジェクトにロードするローカル InputStream オブジェクトです。

openSource( )

構文

```
public int openSource(byte[ ] userArg, byte[ ] [ ] ctx)
```

説明

データ・ソースをオープンします。

パラメータ

userArg
ユーザー定義のソース・プラグインで必要になる可能性があるソース・プラグイン追加情報

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

戻り値

ステータスを、整数で戻します。0（ゼロ）は成功を示し、0（ゼロ）以外の値はソース・プラグイン固有の障害コードを示します。

例外

java.sql.SQLException

データベース内で対応する openSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte[ ] userarg = new byte[64];
byte [ ] [ ] ctx = new byte[1][64];
int i = audObj.openSource(userarg,ctx);
if(i == 0)
     System.out.println("openSource successful");
else
     System.out.println("openSource unsuccessful");
```

パラメータは次のとおりです。

- userArg: 権限関連のパラメータです。
- ctx: ソース・プラグインのコンテキスト情報です。

processAudioCommand( )

構文

```
public byte[ ] processAudioCommand(byte[ ] [ ] ctx,
  String cmd, String args, byte[ ] [ ] result)
```

説明

データベース内でフォーマット・プラグインをコールして、コマンドを実行します。このメソッドは、ユーザー定義のフォーマット・プラグインのみで使用します。このメソッドをオラクル社が提供するフォーマット・プラグインで使用すると、例外が生成されます。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

cmd
フォーマット・プラグインで実行されるコマンド

args
コマンドの引数

result
[1][*n*] 形式のバイト配列。ここに、コマンドの実行結果が書き込まれます。

戻り値

コマンドの実行結果を戻します。

例外

java.sql.SQLException

データベース内で対応する processAudioCommand( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64]
String cmd;
String args;
byte [ ] [ ] result;
//assign a command value to cmd
//assign any arguments to args
byte[ ] commandResults = audObj.processAudioCommand(ctx,cmd,
     args,result);
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。
- cmd: 実行されるコマンドです。
- args: コマンドに必要なすべての引数です。
- result: コマンドの実行結果です。

processSourceCommand( )

構文

```
public byte[ ] processSourceCommand(byte[ ] [ ] ctx,
  String cmd, String args, byte[ ] [ ] result)
```

説明

データベース内でソース・プラグインをコールして、コマンドを実行します。このメソッドは、ユーザー定義のプラグインのみで使用します。このメソッドをオラクル社が提供するソース・プラグインで使用すると、例外が生成されます。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

cmd
ソース・プラグインによって実行されるコマンド

args
コマンドの引数

result
[1][*n*] 形式のバイト配列。ここに、コマンドの実行結果が書き込まれます。

戻り値

コマンドの実行結果を戻します。

例外

java.sql.SQLException

データベース内で対応する processSourceCommand( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
String cmd;
String args;
byte [ ] [ ] result;
//assign a command value to cmd
//assign any arguments to args
byte[ ] commandResults = audObj.processSourceCommand(ctx,cmd,
     args,result);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- cmd: 実行されるコマンドです。
- args: コマンドに必要なすべての引数です。
- result: コマンドの実行結果です。

readFromSource( )

構文

```
public int readFromSource(byte[ ] [ ] ctx,
  int startPos, int numBytes, byte[ ] [ ] buffer)
```

説明

データ・ソースからデータを読み込みます。このメソッドは、指定されたバイト数を、データ・ソースの指定された位置から開始して、データ・ソースからアプリケーション・バッファに読み込みます。

データ・ソースを読込み前にオープンする必要がないソース・プラグインもあります。ただし、アプリケーションが現行または今後のソース・プラグインで必ず動作するようにするには、このメソッドをコールする前に、openSource( ) メソッドをコールします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

startPos
データ・ソース内の開始位置

numBytes
データ・ソースから読み込むバイト数

buffer
[1][*n*] 形式のバイト配列。*n* は、numBytes 以上の値です。

戻り値

読み込んだバイト数を、整数で戻します。

例外

java.sql.SQLException

データベース内で対応する readFromSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
byte [ ] [ ] commentBuffer = new byte[12];
int i = audObj.readFromSource(ctx,0,12,commentBuffer);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- 0: コメント・フィールドからの読込み開始位置です。
- 12: 読込みバイト数です。
- commentBuffer: データの読込み先の位置です。

setAudioDuration( )

構文

```
public void setAudioDuration(int audioDuration)
```

説明

audioDuration 属性の値を設定します。

audioDuration 属性は、setProperties( ) メソッドによって自動的に特定のオーディオ・フォーマットに設定されます。setAudioDuration( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、オーディオ・データ自体は変更されません。

パラメータ

audioDuration
新しい属性値

戻り値

なし

例外

java.sql.SQLException

audioDuration 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setAudioDuration(16);
```

パラメータは次のとおりです。

- 16: audioDuration 属性に設定する値（秒単位）です。

setComments( )

構文

```
public void setComments(oracle.sql.CLOB comments)
```

説明

コメント属性の値を設定します。

*inter*Media によって、使用するコメント属性が予約されます。任意の値を設定できますが、Oracle *inter*Media Annotator または setProperties( ) メソッドによって上書きされる場合があります。

パラメータ

comments
新しい属性値

戻り値

なし

例外

java.sql.SQLException

コメント属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setComments(commentsData);
```

パラメータは次のとおりです。

- commentsData: コメント属性に設定するデータを含む CLOB です。

setCompressionType( )

構文

```
public void setCompressionType(String compressionType)
```

説明

compressionType 属性の値を設定します。

compressionType 属性は、setProperties( ) メソッドによって自動的に特定のオーディオ・フォーマットに設定されます。setCompressionType( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、オーディオ・データ自体は変更されません。

パラメータ

compressionType
新しい属性値

戻り値

なし

例外

java.sql.SQLException

compressionType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setCompressionType("8BITMONOAUDIO");
```

パラメータは次のとおりです。

- 8BITMONOAUDIO: compressionType 属性に設定する値です。

setDescription( )

構文

```
public void setDescription(String description)
```

説明

description 属性の値を設定します。

パラメータ

description
新しい属性値

戻り値

なし

例外

java.sql.SQLException
description 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setDescription("My audio file");
```

パラメータは次のとおりです。

- My audio file: description 属性に設定する値です。

setEncoding( )

構文

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

説明

encoding 属性の値を設定します。

encoding 属性は、setProperties( ) メソッドによって自動的に特定のオーディオ・フォーマットに設定されます。setEncoding( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、オーディオ・データ自体は変更されません。

パラメータ

encoding
新しい属性値

戻り値

なし

例外

java.sql.SQLException

encoding 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setEncoding("MULAW");
```

パラメータは次のとおりです。

- MULAW: encoding 属性に設定する値です。

setFormat( )

構文

```
public void setFormat(String format)
```

説明

format 属性の値を設定します。

format 属性は、オーディオ・データを操作するメソッドへのコールの処理に使用するフォーマット・プラグインを判断します。特に、setProperties( ) メソッドは、format 属性を使用して、オーディオ・データのプロパティを解析するためにコールするフォーマット・プラグインを判断します。setProperties( ) メソッドをコールする前の、format 属性の初期化の方法、および setProperties( ) メソッドをオラクル社が提供するデフォルトのプラグインで使用した format 属性の値の設定方法については、setProperties( ) メソッドを参照してください。setFormat( ) メソッドのコールで設定されるのは属性値のみで、オーディオ・データ自体は変更されません。

パラメータ

format
新しい属性値

戻り値

なし

例外

java.sql.SQLException

format 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setFormat("AUFF");
```

パラメータは次のとおりです。

- AUFF: format 属性に設定する値です。

setKnownAttributes( )

構文

```
public void setKnownAttributes(String format, String encoding,int
                               numberOfChannels, int samplingRate,
                               int sampleSize, String compressionType,
                               int audioDuration)
```

説明

OrdAudio Java オブジェクトの既知の属性の値を設定します。

属性（format、encoding、numberOfChannels、samplingRate、sampleSize、compressionType および audioDuration）は、setProperties( ) メソッドによって自動的に特定のオーディオ・フォーマットに設定されます。setKnownAttributes( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは指定された属性値のみで、オーディオ・データ自体は変更されません。

パラメータ

format
新しい属性値（文字列）

encoding
新しい属性値（文字列）

numberOfChannels
新しい属性値（整数）

samplingRate
新しい属性値（整数）

sampleSize
新しい属性値（整数）

compressionType
新しい属性値（文字列）

audioDuration
新しい属性値（整数）

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setKnownAttributes( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setKnownAttributes("AUFF","MULAW",1,8,8,"8BITMONOAUDIO",16);
```

パラメータは次のとおりです。

- AUFF: format 属性に設定する値です。
- MULAW: encoding 属性に設定する値です。
- 1: numberOfChannels 属性に設定する値です。
- 8: samplingRate 属性に設定する値（1 秒当たりのサンプル数）です。
- 8: sampleSize 属性に設定する値です。
- 8BITMONOAUDIO: compressionType 属性に設定する値です。
- 16: audioDuration 属性に設定する値（秒単位）です。

setLocal( )

構文

```
public void setLocal( )
```

説明

オーディオ・データが、localData 属性で指定されたデータベース内の BLOB でローカルに格納されているかどうかを示すために、ローカル属性の値を設定します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setLocal( );
```

setMimeType( )

構文

```
public void setMimeType(String mimeType)
```

説明

mimeType 属性の値を設定します。

mimeType 属性は、setProperties( ) メソッドによって自動的に特定のオーディオ・フォーマットに設定されます。setMimeType( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、オーディオ・データ自体は変更されません。

パラメータ

mimeType
新しい属性値

戻り値

なし

例外

java.sql.SQLException

mimeType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setMimeType("audio/basic");
```

パラメータは次のとおりです。

- audio/basic: 設定する MIME タイプです。

setNumberOfChannels( )

構文

```
public void setNumberOfChannels(int numberOfChannels)
```

説明

numberOfChannels 属性の値を設定します。

numberOfChannels 属性は、setProperties( ) メソッドによって自動的に特定のオーディオ・フォーマットに設定されます。setNumberOfChannels( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、オーディオ・データ自体は変更されません。

パラメータ

numberOfChannels
新しい属性値

戻り値

なし

例外

java.sql.SQLException

numberOfChannels 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setNumberOfChannels(1);
```

パラメータは次のとおりです。

- 1: numberOfChannels 属性に設定する値です。

setProperties(byte[ ][ ])

構文

```
public void setProperties(byte[ ] [ ] ctx)
```

説明

オーディオ・データのプロパティを解析し、OrdAudio Java オブジェクト内の属性の値を設定します。このメソッドは、format、mimeType、encoding、numberOfChannels、samplingRate、sampleSize、compressionType および audioDuration 属性の値を設定します。対応するプロパティが特定のオーディオ・フォーマットについて抽出できない場合、属性は null に設定されます。オーディオ・フォーマットが認識できない場合は、SQLException エラーがスローされます。

format 属性は、オーディオ・データのプロパティの解析に使用するフォーマット・プラグインを判断します。setProperties( ) メソッドへのコール時に format 属性が null の場合は、オーディオ・データのプロパティの解析、およびサポートされているオーディオ・フォーマットに対応する様々な属性（実際のオーディオ・データ・フォーマットなど）の埋込みに、オラクル社が提供するデフォルトのフォーマット・プラグインが使用されます。オラクル社が提供するフォーマット・プラグインでサポートされるオーディオ・フォーマットの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。データベース内の ORDAudio.init メソッドは、常に、format 属性の値を null に設定することに注意してください。format 属性が null でない場合は、setProperties( ) メソッドへのコール時に、format 属性で指定されたフォーマット・プラグインがコールされます。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
audObj.setProperties(ctx);
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。

setProperties(byte[ ][ ], boolean)

構文

```
public void setProperties(byte[ ] [ ] ctx, boolean setComments)
```

説明

オーディオ・データのプロパティを解析し、OrdAudio Java オブジェクト内の属性の値を設定し、オプションでコメント属性で指定された CLOB を移入します。このメソッドは、format、mimeType、encoding、numberOfChannels、samplingRate、sampleSize、compressionType および audioDuration 属性の値を設定します。対応するプロパティが特定のオーディオ・フォーマットについて抽出できない場合、属性は null に設定されます。setComments パラメータが true の場合、このメソッドは抽出された XML 形式のすべてのプロパティを、コメント属性で指定された CLOB に移入します。setComments パラメータが false の場合、コメント属性は変更されません。オーディオ・フォーマットが認識できない場合は、SQLException エラーがスローされます。

format 属性は、オーディオ・データのプロパティの解析に使用するフォーマット・プラグインを判断します。setProperties( ) メソッドへのコール時に format 属性が null の場合は、オーディオ・データのプロパティの解析、およびサポートされているオーディオ・フォーマットに対応する様々な属性（実際のオーディオ・データ・フォーマットなど）の埋込みに、オラクル社が提供するデフォルトのフォーマット・プラグインが使用されます。オラクル社が提供するフォーマット・プラグインでサポートされるオーディオ・フォーマットの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。データベース内の ORDAudio.init メソッドは、常に、format 属性の値を null に設定することに注意してください。format 属性が null でない場合は、setProperties( ) メソッドへのコール時に、format 属性で指定されたフォーマット・プラグインがコールされます。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

setComments
コメント属性で指定された CLOB を移入するかどうかを指定するブール値

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
audObj.setProperties(ctx,true);
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。
- true: コメント・フィールドが移入されることを示します。

setSampleSize( )

構文

```
public void setSampleSize(int sampleSize)
```

説明

sampleSize 属性の値を設定します。

sampleSize 属性は、setProperties( ) メソッドによって自動的に特定のオーディオ・フォーマットに設定されます。setSampleSize( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、オーディオ・データ自体は変更されません。

パラメータ

sampleSize
新しい属性値

戻り値

なし

例外

java.sql.SQLException

sampleSize 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setSampleSize(8);
```

パラメータは次のとおりです。

- 8: sampleSize 属性に設定する値です。

setSamplingRate( )

構文

```
public void setSamplingRate(int samplingRate)
```

説明

samplingRate 属性の値を設定します。

samplingRate 属性は、setProperties( ) メソッドによって自動的に特定のオーディオ・フォーマットに設定されます。setSamplingRate( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、オーディオ・データ自体は変更されません。

パラメータ

samplingRate
新しい属性値

戻り値

なし

例外

java.sql.SQLException

samplingRate 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setSamplingRate(8);
```

パラメータは次のとおりです。

- 8: samplingRate 属性に設定する値（1 秒当たりのサンプル数）です。

setSource( )

構文

```
public void setSource(String srcType, String srcLocation,
  String srcName)
```

説明

srcType、srcLocation および srcName 属性の値を設定します。

パラメータ

srcType
ソース・タイプ

srcLocation
ソースの位置

srcName
ソースの名前

戻り値

なし

例外

java.sql.SQLException

srcType、srcLocation または srcName 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setSource("file","AUDIODIR","audio.au");
```

パラメータは次のとおりです。

- file: ソース・タイプです。
- AUDIODIR: ソースの位置です。
- audio.au: ソースの名前です。

setUpdateTime( )

構文

```
public void setUpdateTime(java.sql.Timestamp currentTime)
```

説明

updateTime 属性の値を設定します。このメソッドは、updateTime 属性の値を指定された時刻に設定します。ただし、currentTime 属性が null に指定されている場合は、updateTime 属性の値をデータベース・サーバーの現在の SYSDATE 時刻に設定します。

パラメータ

currentTime
updateTime 属性の値をデータベース・サーバーの現在の SYSDATE 時刻に設定するために使用する更新時刻（null 値）

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setUpdateTime( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
audObj.setUpdateTime(null);
```

trimSource( )

構文

```
public int trimSource(byte[ ] [ ] ctx, int newLen)
```

説明

データを指定の長さに切り捨てます。

ただし、切捨て操作をサポートしないソース・プラグインもあります。たとえば、アプリケーションは localData 属性で指定された BLOB に格納されているデータの切捨てを行うことができますが、file および http のデータ・ソース・タイプは書込みアクセスをサポートしないため、このメソッドもサポートしません。また、書込みアクセスをサポートするソース・プラグインが切捨て操作をサポートしない場合もあります。

データ・ソースを変更前にオープンする必要がないソース・プラグインもあります。ただし、アプリケーションが現行または今後のソース・プラグインで必ず動作するようにするには、このメソッドをコールする前に、openSource( ) メソッドをコールします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

newLen
切捨て後のデータの長さ

戻り値

ステータスを、整数で戻します。0（ゼロ）は成功を示し、0（ゼロ）以外の値はソース・プラグイン固有の障害コードを示します。

例外

java.sql.SQLException

データベース内で対応する trimSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
int i = audObj.trimSource(ctx,10);
if (i == 0)
     System.out.println("trimSource successful");
else
     System.out.println("trimSource unsuccessful");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- 10: ソースの新しい長さです。

writeToSource( )

構文

```
public int writeToSource(byte[ ] [ ] ctx,
  int startPos, int numBytes, byte[ ] buffer)
```

説明

データをデータ・ソースに書き込みます。このメソッドは、指定されたバイト数を、データ・ソースの指定された位置から開始してアプリケーションのバッファからデータ・ソースに書き込みます。

ただし、書込み操作をサポートしないソース・プラグインもあります。たとえば、アプリケーションは localData 属性で指定された BLOB への書込みを行うことができますが、file および http のデータ・ソース・タイプは書込みアクセスをサポートしないため、このメソッドもサポートしません。また、書込みアクセスをサポートするソース・プラグインが、順次書込みアクセスのみサポートし、データ・ソース内の任意の位置への書込みアクセスはサポートしない場合もあります。

データ・ソースを書込み前にオープンする必要がないソース・プラグインもあります。ただし、アプリケーションが現行または今後のソース・プラグインで必ず動作するようにするには、このメソッドをコールする前に、openSource( ) メソッドをコールします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

startPos
データ・ソース内の開始位置

numBytes
データ・ソースに書き込むバイト数

buffer
書き込むデータを含むバイト配列

戻り値

書き込んだバイト数を、整数で戻します。

例外

java.sql.SQLException

データベース内で対応する writeToSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
byte[ ] data = new byte[20];
//populate data with 20 bytes of content
int i = audObj.writeToSource(ctx,1,20,data);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- 1: 書込みが開始されるコメント・フィールドの位置です。
- 20: 書込みバイト数です。
- data: 書き込むコンテンツです。

4

OrdDoc リファレンス情報

OrdDoc クラスは、Java アプリケーションで ORDSYS.ORDDoc データベース型のインスタンスを表すために使用されます。OrdDoc クラスには、OrdDoc Java オブジェクトに対して様々な操作を行う一連のメソッドおよび様々なオブジェクト属性を取得し、設定するための一連のメソッドがあります。

ほぼすべてのメソッドで、アプリケーション内の OrdDoc Java オブジェクトの属性が操作されます。読込みまたは書込みを目的としてメディア・データにアクセスするメソッドは例外です。例外のメソッドには、次のものがあります。

- localData 属性で指定されたデータベース BLOB を操作し、データベース BLOB 内に格納されたデータの読込みおよび書込みを行うメソッド
- srcType 属性が file の場合に srcLocation および srcName 属性で指定されたデータベース BFILE を操作し、データベース・サーバー内の指定されたファイルからデータを読み込むメソッド
- srcType 属性が http の場合に srcType、srcLocation および srcName 属性で指定された URL を操作し、指定された URL でリソースからデータを読み込むメソッド

アプリケーションによって、データベース内の OrdDoc Java オブジェクトまたはメディア・データが変更される場合は、これらの変更を永続的なものにするために、データベース内の ORDDoc SQL オブジェクトを更新する必要があります。

OrdDoc Java クラスの一部のメソッドは、データベース・ソース・プラグインまたはデータベース・フォーマット・プラグインに渡されて処理され、コンテキスト・パラメータに `byte [ ] [ ] ctx` を取ります。ソース・プラグインまたはフォーマット・プラグインで必要になる可能性のあるコンテキスト情報を保持するには、アプリケーションで 64 バイトの配列を割り当てる必要があります。たとえば、プラグインでは、一度のコールでコンテキスト情報を初期化し、その情報を後続のコールで使用できます。配列は、ソース・プラグインのコンテキストおよびフォーマット・プラグインのコンテキストに 1 つずつ必要です。ほとんどのプラグインは、64 バイトで十分ですが、ユーザー定義のプラグインによっては、追加領域が必要になる場合もあります。次の例では、プラグインのコンテキスト情報配列の割当て方法を示します。

```
byte [] [] ctx = new byte[1][64];
```

注意： 今回のリリースでは、オラクル社が提供するソース・プラグインまたはフォーマット・プラグインでは、コンテキストは保持されません。また、ユーザー定義のソース・プラグインまたはフォーマット・プラグインでも、コンテキストが保持されない場合もあります。ただし、前述した方法でコンテキスト・パラメータを指定すると、アプリケーションは現行または今後のソース・プラグインまたはフォーマット・プラグインで動作します。

プラグインの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

4.1 前提条件

*inter*Media メソッドを実行するために、次のインポート文を Java ファイルに組み込む必要があります。

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
import oracle.ord.im.*;
```

この章で示す例は、次の操作がすでに実行済であることを示しています。

- OrdDoc 型の列を含む表への接続が確立されている。
- ローカル OrdDoc オブジェクトとして docObj が作成され、データが移入されている。

接続の確立またはローカル・オブジェクトの移入の例については、2.2.2 項を参照してください。

4.2 リファレンス情報

この項では、OrdDoc オブジェクトを操作するメソッドについて説明します。

clearLocal( )

構文

```
public void clearLocal( )
```

説明

ローカル属性を消去して、メディア・データが外部に格納されていることを示します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

ローカル属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
docObj.clearLocal( )
```

closeSource( )

構文

```
public int closeSource(byte[ ] [ ] ctx)
```

説明

データ・ソースをクローズします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

戻り値

ステータスを、整数で戻します。0（ゼロ）は成功を示し、0（ゼロ）以外の値はソース・プラグイン固有の障害コードを示します。

例外

java.sql.SQLException

データベース内で対応する closeSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
int i = docObj.closeSource(ctx);
if(i == 0)
     System.out.println("Source close successful");
else
     System.out.println("Source close unsuccessful");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。

deleteContent( )

構文

```
public void deleteContent( )
```

説明

localData 属性で指定されたデータベース BLOB に格納されているデータを削除します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する deleteContent( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
docObj.deleteContent( );
```

export( )

構文

```
public void export (byte[ ] [ ] ctx, String srcType,
  String srcLocation, String srcName)
```

説明

localData 属性で指定された BLOB からデータをエクスポートします。このメソッドは、データベース内の対応する export( ) メソッドをコールして、srcType、srcLocation および srcName の各パラメータで指定された位置に、メディア・データをエクスポートします。

ただし、export( ) メソッドをサポートしないソース・プラグインもあります。file ソース・タイプのみがネイティブにサポートされています。

このメソッドは、Oracle データベース・サーバーのリリース 8.1.7 以上で実行している場合のみ機能します。

後半部分では、export( ) メソッドおよびオラクル社が提供する file ソース・プラグインの使用方法について説明します。ユーザー定義のプラグインの動作は異なります。

オラクル社が提供する file ソース・プラグインによって実装された export( ) メソッドは、localData 属性で指定されたデータベース BLOB 内に格納されているメディア・データのコピーのみ行い、変更は行いません。

メディア・データのエクスポート後も、すべてのメディア・プロパティ属性は変更されません。ただし、srcType、srcLocation および srcName 属性は、export( ) メソッドに渡された srcType、srcLocation および srcName の各パラメータの値で更新されます。export( ) メソッドへのコール後、データベース内部のメディア・データを管理する必要がない場合は、clearLocal( ) メソッドをコールして、メディア・データがデータベースの外部に格納されていることを示し、deleteContent( ) メソッドをコールして、データベース BLOB に格納されているメディア・データを削除します。

データベース内の export( ) メソッドは、ユーザーがアクセス権限を所有しているデータベース・ディレクトリ・オブジェクトに対してのみ書込みを行います。つまり、SQL の CREATE DIRECTORY 文を使用して作成したディレクトリ、または読込み権限を付与されたディレクトリにアクセスできます。CREATE DIRECTORY 文を実行するには、CREATE ANY DIRECTORY 権限が必要です。また、書込み可能なファイルを指定するには、DBMS_JAVA.GRANT_PERMISSION メソッドを使用する必要があります。

たとえば、次の SQL*Plus コマンドは、ユーザー MEDIAUSER にファイル filmclip1.mov の書込み権限を付与します。

```
CALL DBMS_JAVA.GRANT_PERMISSION(
    'MEDIAUSER',
     'java.io.FilePermission',
     '/media/movies/filmclip1.mov',
     'write');
```

前述の例は、単一ファイルへの書込み権限を許可する方法を示しています。この他にも、ワイルド・カードを使用した、複数のディレクトリおよびファイル名への書込み権限を許可する様々なパス指定があります。たとえば、スラッシュとアスタリスク（/*）（スラッシュは、オペレーティング・システム固有のファイル・セパレータ文字）で終わるパス指定は、指定されたディレクトリに含まれているすべてのファイルを示します。スラッシュとハイフン（/-）で終わるパス指定は、指定されたディレクトリおよびそのすべてのサブディレクトリに含まれているすべてのファイルを示します。特別なトークン <<ALL FILES>> で構成されたパス名は、すべてのファイルへのアクセス権限を許可します。

セキュリティおよびパフォーマンスの詳細は、『Oracle9*i* Java Developer's Guide』および Java API の java.io.FilePermission クラスを参照してください。必要な権限の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

srcType
コンテンツのエクスポート先のソース・タイプ

srcLocation
コンテンツのエクスポート先のソースの位置

srcName
コンテンツのエクスポート先のソース名

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する export( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
docObj.export(ctx,"file","DOCDIR","complete.xml");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- file: コンテンツのエクスポートに使用するソース・プラグインです。
- DOCDIR: コンテンツのエクスポート先の位置です。
- complete.xml: コンテンツのエクスポート先のファイルです。

getBFILE( )

構文

```
public oracle.sql.BFILE getBFILE( )
```

説明

srcType 属性が file の場合、データベースから BFILE ロケータを戻します。このメソッドは、データベース内の対応する getBFILE( ) メソッド（srcLocation および srcName 属性を使用して BFILE を作成）をコールします。

パラメータ

なし

戻り値

oracle.sql.BFILE ロケータを戻します。

例外

java.sql.SQLException

データベース内で対応する getBFILE( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
BFILE documentBFILE = docObj.getBFILE( );
```

getComments( )

構文

```
public oracle.sql.CLOB getComments( )
```

説明

コメント属性から CLOB ロケータを戻します。

パラメータ

なし

戻り値

コメント属性の値を、oracle.sql.CLOB ロケータで戻します。

例外

java.sql.SQLException

コメント属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
CLOB comments = docObj.getComments( )
```

getContent( )

構文

```
public oracle.sql.BLOB getContent( )
```

説明

localData 属性から BLOB ロケータを戻します。

パラメータ

なし

戻り値

oracle.sql.BLOB ロケータを戻します。

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
BLOB localContent = docObj.getContent( );
```

getContentInLob( )

構文

```
public oracle.sql.BLOB getContentInLob(byte[ ] [ ] ctx,
  String mimeType[ ], String format[ ])
```

説明

データベース内の一時 BLOB の localData 属性で指定された BLOB からデータを戻します。このメソッドは、データベース内に一時 BLOB を作成し、localData 属性で指定された BLOB からデータを読み込み、一部 BLOB にデータを書き込み、一時 BLOB ロケータをコール元に戻します。

> **注意：** アプリケーションでは、一時 BLOB 内に含まれている情報にアクセスした後、一時 BLOB を解放する必要があります。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

mimeType
mimeType 属性が要素 0 として書き込まれる文字列配列（長さは 1 要素）

format
format 属性が要素 0 として書き込まれる文字列配列（長さは 1 要素）

戻り値

一時 oracle.sql.BLOB ロケータのメディア・データを戻します。

例外

java.sql.SQLException

一時 BLOB の作成中、またはデータベース内で対応する getContentInLob( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
String mimetype[ ] = new String[1];
String format[ ] = new String[1];
BLOB localContent = docObj.getContentInLob(ctx,mimetype,format);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- mimeType: 最初の値に LOB データの MIME タイプを含む文字列の配列です。この値は、サーバーによって生成されます。
- format: 最初の値に LOB データのフォーマットを含む文字列の配列です。この値は、サーバーによって生成されます。

getContentLength( )

構文

```
public int getContentLength( )
```

説明

contentLength 属性の値を戻します。

パラメータ

なし

戻り値

contentLength 属性の値を、整数で戻します。

例外

java.sql.SQLException

contentLength 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int contentLength = docObj.getContentLength( );
```

getDataInByteArray( )

構文

```
public byte[ ] getDataInByteArray( )
```

説明

localData 属性で指定されたデータベース BLOB からのデータを含むバイト配列を戻します。

パラメータ

なし

戻り値

データを含むバイト配列を戻します。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

BLOB からのデータの読込み中にエラーが発生した場合、この例外がスローされます。

java.lang.OutOfMemoryError

データの保持のために十分なメモリーを割当てできない場合、この例外がスローされます。

例

```
byte[ ] byteArr = docObj.getDataInByteArray( );
```

getDataInFile( )

構文

```
public boolean getDataInFile(String filename)
```

説明

localData 属性で指定されたデータベース BLOB からローカル・ファイルに、データを書き込みます。

パラメータ

filename
データの書込み先のファイル名

戻り値

データが正常にファイルに書き込まれた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

BLOB からのデータの読込み中または出力ファイルへのデータの書込み中にエラーが発生した場合、この例外がスローされます。

例

```
boolean load = docObj.getDataInFile("output1.dat");
if(load)
     System.out.println("getDataInFile completed successfully");
else
     System.out.println("Error in getDataInFile");
```

パラメータは次のとおりです。

- output1.dat: データのロード先のファイルです。

getDataInStream( )

構文

```
public InputStream getDataInStream( )
```

説明

localData 属性で指定されたデータベース BLOB のデータの読込み元となる InputStream オブジェクトを戻します。

パラメータ

なし

戻り値

データの読込み元から InputStream オブジェクトを戻します。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
InputStream inpStream = docObj.getDataInStream( );
```

getFactory( )

構文

```
public static oracle.sql.CustomDatumFactory getFactory( )
```

説明

getCustomDatum( ) メソッドが使用する OrdDoc CustomDatumFactory インタフェースを戻します。OracleResultSet または OracleCallableStatement オブジェクトから OrdDoc オブジェクトを取り出す場合は、getFactory( ) メソッドを getCustomDatum( ) メソッドの factory パラメータとして指定します。

パラメータ

なし

戻り値

CustomDatumFactory インタフェースの OrdDoc 実装を戻します。

例外

なし

例

```
OrdDoc media = (OrdDoc)rset.getCustomDatum( 1, OrdDoc.getFactory() );
```

getFormat( )

構文

```
public String getFormat( )
```

説明

format 属性の値を戻します。

パラメータ

なし

戻り値

format 属性の値を、文字列で戻します。

例外

java.sql.SQLException

format 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String format = docObj.getFormat( );
```

getMimeType( )

構文

```
public String getMimeType( )
```

説明

mimeType 属性の値を戻します。

パラメータ

なし

戻り値

mimeType 属性の値を、文字列で戻します。

例外

java.sql.SQLException

mimeType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String mimeType = docObj.getMimeType( );
```

getSource( )

構文

```
public String getSource( )
```

説明

srcType://srcLocation/srcName という形式でソース情報を戻します。

パラメータ

なし

戻り値

ソース情報を、文字列で戻します。

例外

java.sql.SQLException

データベース内で対応する getSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
String source = docObj.getSource( );
```

getSourceLocation( )

構文

```
public String getSourceLocation( )
```

説明

srcLocation 属性の値を戻します。

パラメータ

なし

戻り値

srcLocation 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcLocation 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String location = docObj.getSourceLocation( );
```

getSourceName( )

構文

```
public String getSourceName( )
```

説明

srcName 属性の値を戻します。

パラメータ

なし

戻り値

srcName 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcName 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String name = docObj.getSourceName( );
```

getSourceType( )

構文

```
public String getSourceType( )
```

説明

srcType 属性の値を戻します。

パラメータ

なし

戻り値

srcType 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String type = docObj.getSourceType( );
```

getUpdateTime( )

構文

```
public java.sql.Timestamp getUpdateTime( )
```

説明

updateTime 属性の値を戻します。

パラメータ

なし

戻り値

updateTime 属性の値を java.sql.Timestamp オブジェクトで戻します。

例外

java.sql.SQLException

upateTime 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
Timestamp time = docObj.getUpdateTime( );
```

importData( )

構文

```
public void importData(byte[ ] [ ] ctx, boolean setProp)
```

説明

データを、外部ソースから localData 属性で指定されたデータベース BLOB にインポートします。また、オプションで setProperties( ) メソッドをコールします。setProp パラメータが true の場合、このメソッドはデータベース内の setProperties( ) メソッドをコールして、プロパティ属性を設定し、コメント属性で指定された CLOB を移入します。外部データ・ソースは、srcType、srcLocation および srcName 属性で指定されます。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

setProp
setProperties( ) メソッドをコールするかどうかを指定するブール値

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する import( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
docObj.importData(ctx);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。

importFrom( )

構文

```
public void importFrom(byte[ ] [ ] ctx, String srcType,
  String srcLocation, String srcName, boolean setProp)
```

説明

データを、外部ソースから localData 属性で指定されたデータベース BLOB にインポートします。また、オプションで setProperties( ) メソッドをコールします。setProp パラメータが true の場合、このメソッドはデータベース内の setProperties( ) メソッドをコールして、プロパティ属性を設定し、コメント属性で指定された CLOB を移入します。外部データ・ソースは、srcType、srcLocation および srcName パラメータで指定されます。srcType、srcLocation および srcName 属性は、importFrom( ) メソッドに渡された srcType、srcLocation および srcName の各パラメータの値で更新されます。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

srcType
データのインポート元のソース・タイプ

srcLocation
データのインポート元のソースの位置

srcName
データのインポート元のソース名

setProp
setProperties( ) メソッドをコールするかどうかを指定するブール値

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する importFrom( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
docObj.importFrom("file","DOCDIR","testdoc.dat");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- file: データのインポートに使用するソース・プラグインです。
- DOCDIR: データのインポート元のデータベース・サーバー上でのファイルの位置です。
- testdoc.dat: データのインポート元のファイルです。

isLocal( )

構文

```
public boolean isLocal( )
```

説明

メディア・データが、localData 属性で指定されたデータベース内の BLOB でローカルに格納されているかどうかを示します。

パラメータ

なし

戻り値

データベース内の BLOB でデータがローカルに格納されている場合は true、格納されていない場合は false を戻します。

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
if(docObj.isLocal( ))
     System.out.println("local attribute is set to true");
else
     System.out.println("local attribute is set to false");
```

loadDataFromByteArray( )

構文

```
public boolean loadDataFromByteArray(byte[ ] byteArr)
```

説明

データを、バイト配列から localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

byteArr
データのロード元のローカル・バイト配列の名前

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中またはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

バイト配列の読込み中にエラーが発生した場合、この例外がスローされます。

例

```
byte[ ] data = new byte[32000];
FileInputStream fStream = new FileInputStream("testdoc.dat");
fStream.read(data,0,32000);
boolean success = docObj.loadDataFromByteArray(data);
if(success)
     System.out.println("loadDataFromByteArray was successful");
else
     System.out.println("loadDataFromByteArray was unsuccessful");
```

パラメータは次のとおりです。

- data: データのロード元のローカル・バイト配列です。
- testdoc.dat: 32,000 バイトのデータを含むローカル・ファイルです。

loadDataFromFile( )

構文

```
public boolean loadDataFromFile(String filename)
```

説明

データを、ファイルから localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

filename
データのロード元のローカル・ファイル名

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中またはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

データ・ファイルの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
docObj.loadDataFromFile("testdoc.dat");
```

パラメータは次のとおりです。

- testdoc.dat: メディア・データを含むローカル・ファイルです。

loadDataFromInputStream( )

構文

```
public boolean loadDataFromInputStream(InputStream inpStream)
```

説明

データを、InputStream オブジェクトから localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

inpStream
データのロード元の InputStream オブジェクトの名前

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中またはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

InputStream オブジェクトの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
FileInputStream fStream = new FileInputStream("testdoc.dat");
docObj.loadDataFromInputStream(fStream);
```

パラメータは次のとおりです。

- testdoc.dat: メディア・データを含むローカル・ファイルです。
- fStream: メディア・データを OrdDoc オブジェクトにロードするローカル InputStream オブジェクトです。

openSource( )

構文

```
public int openSource(byte[ ] userArg, byte[ ] [ ] ctx)
```

説明

データ・ソースをオープンします。

パラメータ

userArg
ユーザー定義のソース・プラグインで必要になる可能性があるソース・プラグイン追加情報

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

戻り値

ステータスを、整数で戻します。0（ゼロ）は成功を示し、0（ゼロ）以外の値はソース・プラグイン固有の障害コードを示します。

例外

java.lang.Exception

データベース内で対応する openSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte[ ] userarg = new byte[64];
byte [ ] [ ] ctx = new byte[1][64];
int i = docObj.openSource(userarg,ctx);
if(i == 0)
     System.out.println("openSource successful");
else
     System.out.println("openSource unsuccessful");
```

パラメータは次のとおりです。

- userArg: 権限関連のパラメータです。
- ctx: ソース・プラグインのコンテキスト情報です。

processSourceCommand( )

構文

```
public byte[ ] processSourceCommand(byte[ ] [ ] ctx,
  String cmd, String args, byte[ ] [ ] result)
```

説明

データベース内でソース・プラグインをコールして、コマンドを実行します。このメソッドは、ユーザー定義のプラグインのみで使用します。このメソッドをオラクル社が提供するソース・プラグインで使用すると、例外が生成されます。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

cmd
ソース・プラグインによって実行されるコマンド

args
コマンドの引数

result
[1][*n*] 形式のバイト配列。ここにコマンドの実行結果が書き込まれます。

戻り値

コマンドの実行結果を戻します。

例外

java.sql.SQLException

データベース内で対応する processSourceCommand( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
String cmd;
String args;
byte [ ] [ ] result;
//assign a command value to cmd
//assign any arguments to args
byte[ ] commandResults = docObj.processSourceCommand(ctx,cmd,
     args,result);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- cmd: 実行されるコマンドです。
- args: コマンドに必要なすべての引数です。
- result: コマンドの実行結果です。

readFromSource( )

構文

```
public int readFromSource(byte[ ] [ ] ctx,
  int startPos, int numBytes, byte[ ] [ ] buffer)
```

説明

データ・ソースからデータを読み込みます。このメソッドは、指定されたバイト数を、データ・ソースの指定された位置から開始して、データ・ソースからアプリケーション・バッファに読み込みます。

データ・ソースを読込み前にオープンする必要がないソース・プラグインもあります。ただし、アプリケーションが現行または今後のソース・プラグインで必ず動作するようにするには、このメソッドをコールする前に、openSource( ) メソッドをコールします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

startPos
データ・ソース内の開始位置

numBytes
データ・ソースから読み込むバイト数

buffer
[1][*n*] 形式のバイト配列。*n* は、numBytes 以上の値です。

戻り値

読み込んだバイト数を、整数で戻します。

例外

java.sql.SQLException

データベース内で対応する readFromSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
byte [ ] [ ] buffer = new byte[12][1];
int i = docObj.readFromSource(ctx,0,12,buffer);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- 0: localData フィールドからの読込み開始位置です。
- 12: 読込みバイト数です。
- buffer: データの読込み先のバッファです。データは、buffer[0] に格納されます。

setComments( )

構文

```
public void setComments(oracle.sql.CLOB comments)
```

説明

コメント属性の値を設定します。

*inter*Media によって、使用するコメント属性が予約されます。任意の値を設定できますが、Oracle *inter*Media Annotator または setProperties( ) メソッドによって上書きされる場合があります。

パラメータ

comments
新しい属性値

戻り値

なし

例外

java.sql.SQLException

コメント属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
docObj.setComments(commentsData);
```

パラメータは次のとおりです。

- commentsData: コメント属性に設定するデータを含む CLOB です。

setContentLength( )

構文

```
public void setContentLength(int contentLength)
```

説明

contentLength 属性の値を設定します。

contentLength 属性は、setProperties( ) メソッドによって自動的に特定のメディア・フォーマットに設定されます。setContentLength( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、メディア・データ自体は変更されません。

パラメータ

contentLength
新しい属性値

戻り値

なし

例外

java.sql.SQLException

contentLength 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

なし

setFormat( )

構文

```
public void setFormat(String format)
```

説明

format 属性の値を設定します。

format 属性は、メディア・データを操作するメソッドへのコールの処理に使用するフォーマット・プラグインを判断します。特に、setProperties( ) メソッドは、format 属性を使用して、メディア・データのプロパティを解析するためにコールするフォーマット・プラグインを判断します。setProperties( ) メソッドをコールする前の、format 属性の初期化の方法、および setProperties( ) メソッドをオラクル社が提供するデフォルトのプラグインで使用した format 属性の値の設定方法については、setProperties( ) メソッドを参照してください。setFormat( ) メソッドのコールで設定されるのは属性値のみで、メディア・データ自体は変更されません。

パラメータ

format
新しい属性値

戻り値

なし

例外

java.sql.SQLException

format 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
docObj.setFormat("AUFF");
```

パラメータは次のとおりです。

- AUFF: format 属性に設定する値です。

setLocal( )

構文

```
public void setLocal( )
```

説明

メディア・データが、localData 属性で指定されたデータベース内の BLOB でローカルに格納されているかどうかを示すために、ローカル属性の値を設定します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
docObj.setLocal( );
```

setMimeType( )

構文

```
public void setMimeType(String mimeType)
```

説明

mimeType 属性の値を設定します。

mimeType 属性は、setProperties( ) メソッドによって自動的に特定のメディア・フォーマットに設定されます。setMimeType( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、メディア・データ自体は変更されません。

パラメータ

mimeType
新しい属性値

戻り値

なし

例外

java.sql.SQLException

mimeType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
docObj.setMimeType("application/pdf");
```

パラメータは次のとおりです。

- application/pdf: 設定する MIME タイプです。

setProperties( )

構文

```
public void setProperties(byte[ ] [ ] ctx, boolean setComments)
```

説明

メディア・データのプロパティを解析し、OrdDoc Java オブジェクト内の属性の値を設定し、オプションでコメント属性で指定された CLOB を移入します。このメソッドは、format、mimeType および contentLength の値を設定します。対応するプロパティが特定のメディア・フォーマットについて抽出できない場合、属性は null に設定されます。setComments パラメータが true の場合、このメソッドは抽出された XML 形式のすべてのプロパティを、コメント属性で指定された CLOB に移入します。setComments パラメータが false の場合、コメント属性は変更されません。メディア・フォーマットが認識できない場合は、SQLException エラーがスローされます。

format 属性は、メディア・データのプロパティの解析に使用するフォーマット・プラグインを判断します。setProperties( ) メソッドへのコール時に、format 属性が null の場合は、メディア・データのプロパティの解析、およびサポートされているメディア・フォーマットに対応する様々な属性（実際のメディア・データ・フォーマットなど）の埋込みに、オラクル社が提供するデフォルトのフォーマット・プラグインが使用されます。オラクル社が提供するフォーマット・プラグインでサポートされるメディア・フォーマットの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。データベース内の ORDDoc.init メソッドは、常に、format 属性の値を null に設定することに注意してください。format 属性が null でない場合は、setProperties( ) メソッドへのコール時に、format 属性で指定されたフォーマット・プラグインがコールされます。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

setComments
コメント属性で指定された CLOB を移入するかどうかを指定するブール値

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
docObj.setProperties(ctx,true);
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。
- true: コメント・フィールドが移入されることを示します。

setSource( )

構文

```
public void setSource(String srcType, String srcLocation,
  String srcName)
```

説明

srcType、srcLocation および srcName 属性の値を設定します。

パラメータ

srcType
ソース・タイプ

srcLocation
ソースの位置

srcName
ソースの名前

戻り値

なし

例外

java.sql.SQLException

srcType、srcLocation または srcName 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
docObj.setSource("file","DOCDIR","sample.pdf");
```

パラメータは次のとおりです。

- file: ソース・タイプです。
- DOCDIR: ソースの位置です。
- sample.pdf: ソースの名前です。

setUpdateTime( )

構文

```
public void setUpdateTime(java.sql.Timestamp currentTime)
```

説明

updateTime 属性の値を設定します。このメソッドは、updateTime 属性の値を指定された時刻に設定します。ただし、currentTime 属性が null に指定されている場合は、updateTime 属性の値をデータベース・サーバーの現在の SYSDATE 時刻に設定します。

パラメータ

currentTime
updateTime 属性の値をデータベース・サーバーの現在の SYSDATE 時刻に設定するために使用する更新時刻（null 値）

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setUpdateTime( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
docObj.setUpdateTime(null);
```

trimSource( )

構文

```
public int trimSource(byte[ ] [ ] ctx, int newLen)
```

説明

データを指定の長さに切り捨てます。

ただし、切捨て操作をサポートしないソース・プラグインもあります。たとえば、アプリケーションは localData 属性で指定された BLOB に格納されているデータの切捨てを行うことができますが、file および http のデータ・ソース・タイプは書込みアクセスをサポートしないため、このメソッドもサポートしません。また、書込みアクセスをサポートするソース・プラグインが切捨て操作をサポートしない場合もあります。

データ・ソースを変更前にオープンする必要がないソース・プラグインもあります。ただし、アプリケーションが現行または今後のソース・プラグインで必ず動作するようにするには、このメソッドをコールする前に、openSource( ) メソッドをコールします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

newLen
切捨て後のデータの長さ

戻り値

ステータスを、整数で戻します。0（ゼロ）は成功を示し、0（ゼロ）以外の値はソース・プラグイン固有の障害コードを示します。

例外

java.sql.SQLException

データベース内で対応する trimSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
int i = docObj.trimSource(ctx,10);
if (i == 0)
     System.out.println("trimSource successful");
else
     System.out.println("trimSource unsuccessful");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- 10: ソースの新しい長さです。

writeToSource( )

構文

```
public int writeToSource(byte[ ] [ ] ctx,
  int startPos, int numBytes, byte[ ] buffer)
```

説明

データをデータ・ソースに書き込みます。このメソッドは、指定されたバイト数を、データ・ソースの指定された位置から開始してアプリケーションのバッファからデータ・ソースに書き込みます。

ただし、書込み操作をサポートしないソース・プラグインもあります。たとえば、アプリケーションは localData 属性で指定された BLOB への書込みを行うことができますが、file および http のデータ・ソース・タイプは書込みアクセスをサポートしないため、このメソッドもサポートしません。また、書込みアクセスをサポートするソース・プラグインが、順次書込みアクセスのみサポートし、データ・ソース内の任意の位置への書込みアクセスはサポートしない場合もあります。

データ・ソースを書込み前にオープンする必要がないソース・プラグインもあります。ただし、アプリケーションが現行または今後のソース・プラグインで必ず動作するようにするには、このメソッドをコールする前に、openSource( ) メソッドをコールします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

startPos
データ・ソース内の開始位置

numBytes
データ・ソースに書き込むバイト数

buffer
書き込むデータを含むバイト配列

戻り値

書き込んだバイト数を、整数で戻します。

例外

java.sql.SQLException

データベース内で対応する writeToSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
byte[ ] data = new byte[20];
//populate data with 20 bytes of content
int i = docObj.writeToSource(ctx,1,20,data);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- 1: 書込みが開始される localData フィールドの位置です。
- 20: 書込みバイト数です。
- data: 書き込むコンテンツです。

5
OrdImage リファレンス情報

OrdImage クラスは、Java アプリケーションで ORDSYS.ORDImage データベース型のインスタンスを表すために使用されます。OrdImage クラスには、OrdImage Java オブジェクトに対して様々な操作を行う一連のメソッドおよび様々なオブジェクト属性を取得し、設定するための一連のメソッドがあります。

ほぼすべてのメソッドで、アプリケーション内の OrdImage Java オブジェクトの属性が操作されます。読込みまたは書込みを目的としてイメージ・データにアクセスするメソッドは例外です。例外のメソッドには、次のものがあります。

- localData 属性で指定されたデータベース BLOB を操作し、データベース BLOB 内に格納されたデータの読込みおよび書込みを行うメソッド
- srcType 属性が file の場合に srcLocation および srcName 属性で指定されたデータベース BFILE を操作し、データベース・サーバー内の指定されたファイルからデータを読み込むメソッド
- srcType 属性が http の場合に srcType、srcLocation および srcName 属性で指定された URL を操作し、指定された URL でリソースからデータを読み込むメソッド

アプリケーションによって、データベース内の OrdImage Java オブジェクトまたはイメージ・データが変更される場合は、これらの変更を永続的なものにするために、データベース内の ORDImage SQL オブジェクトを更新する必要があります。

OrdImage レベルで起動されたメソッドは、データベース・ソース・プラグインまたはデータベース・フォーマット・プラグインに渡されて処理され、コンテキスト・パラメータに `byte[ ] [ ]ctx` を取ります。クライアント・システムが、データベース・サーバーに接続している場合、パラメータ領域（参照例では 64 バイトの領域）はクライアントによって作成されますが、コンテキスト・パラメータのコンテンツはサーバーによって生成されます。コンテキスト・パラメータは、コンテキスト情報を処理するためにクライアントからサーバーに渡されます。

OrdImage Java クラスの一部のメソッドは、データベース・ソース・プラグインまたはデータベース・フォーマット・プラグインに渡されて処理され、コンテキスト・パラメータに byte [ ] [ ] ctx を取ります。ソース・プラグインまたはフォーマット・プラグインで必要になる可能性のあるコンテキスト情報を保持するには、アプリケーションで 64 バイトの配列を割り当てる必要があります。たとえば、プラグインでは、一度のコールでコンテキスト情報を初期化し、その情報を後続のコールで使用できます。配列は、ソース・プラグインのコンテキストおよびフォーマット・プラグインのコンテキストに 1 つずつ必要です。ほとんどのプラグインは、64 バイトで十分ですが、ユーザー定義のプラグインによっては、追加領域が必要になる場合もあります。次の例では、プラグインのコンテキスト情報配列の割当て方法を示します。

```
byte [] [] ctx = new byte[1][64];
```

注意： 今回のリリースでは、オラクル社が提供するソース・プラグインまたはフォーマット・プラグインでは、コンテキストは保持されません。また、ユーザー定義のソース・プラグインまたはフォーマット・プラグインでも、コンテキストが保持されない場合もあります。ただし、前述した方法でコンテキスト・パラメータを指定すると、アプリケーションは現行または今後のソース・プラグインまたはフォーマット・プラグインで動作します。

プラグインの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

5.1 前提条件

*inter*Media メソッドを実行するために、次のインポート文を Java ファイルに組み込む必要があります。

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
import oracle.ord.im.*;
```

この章で示す例は、次の操作がすでに実行済であることを示しています。

- OrdImage 型の列を含む表への接続が確立されている。
- ローカル OrdImage オブジェクトとして imgObj が作成され、データが移入されている。

接続の確立またはローカル・オブジェクトの移入の例については、2.3.2 項を参照してください。

5.2 リファレンス情報

この項では、OrdImage オブジェクトを操作するメソッドについて説明します。

checkProperties( )

構文

```
public boolean checkProperties( )
```

説明

イメージ・データのプロパティが、OrdImage Java オブジェクトの属性と一貫性があるかどうかを確認します。

パラメータ

なし

戻り値

イメージ・データのプロパティが、OrdImage Java オブジェクトの属性と一貫性がある場合は true、一貫性がない場合は false を戻します。

例外

java.sql.SQLException

データベース内で対応する checkProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
if(imgObj.checkProperties( ))
     System.out.println("checkProperties successful");
```

clearLocal( )

構文

```
public void clearLocal( )
```

説明

ローカル属性を消去して、イメージ・データが外部に格納されていることを示します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

ローカル属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
imgObj.clearLocal( );
```

copy( )

構文

```
public void copy(OrdImage dest)
```

説明

OrdImage Java オブジェクトをコピーします。このメソッドは、データベース内の対応する copy( ) メソッドをコールします。copy( ) メソッドは、localData 属性で指定される BLOB を除き、現行の OrdImage オブジェクトのすべての属性を、宛先 OrdImage オブジェクトにコピーします。イメージ・データは、データベース内でローカルに格納されている場合、現行の OrdImage オブジェクトの localData 属性で指定された BLOB から、宛先オブジェクトの localData 属性で指定された BLOB にコピーされます。

パラメータ

dest
データのコピー先の宛先 OrdImage オブジェクト

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する copy( ) メソッドのコール中にエラーが発生した場合、この例外がスローされます。

例

```
//create and populate an object named imgObj2
imgObj.copy(imgObj2);
```

パラメータは次のとおりです。

- imgObj2:imgObj からイメージ・データを受け取る OrdImage オブジェクトです。

deleteContent( )

構文

```
public void deleteContent( )
```

説明

localData 属性で指定されたデータベース BLOB に格納されているデータを削除します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する deleteContent( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
imgObj.deleteContent( );
```

export( )

構文

```
public void export (byte[ ] [ ], ctx String srcType,
  String srcLocation, String srcName)
```

説明

localData 属性で指定された BLOB からデータをエクスポートします。このメソッドは、データベース内の対応する export( ) メソッドをコールして、srcType、srcLocation および srcName の各パラメータで指定された位置に、イメージ・データをエクスポートします。

ただし、export( ) メソッドをサポートしないソース・プラグインもあります。たとえば、file ソース・タイプは、export( ) メソッドをサポートする唯一のオラクル社が提供するソース・タイプです。

このメソッドは、Oracle データベース・サーバーのリリース 8.1.7 以上で実行している場合のみ機能します。

後半部分では、export( ) メソッドおよびオラクル社が提供する file ソース・プラグインの使用方法について説明します。ユーザー定義のプラグインの動作は異なります。

オラクル社が提供する file ソース・プラグインによって実装された export( ) メソッドは、localData 属性で指定されたデータベース BLOB 内に格納されているメディア・データのコピーのみ行い、変更は行いません。

イメージ・データのエクスポート後も、すべてのイメージ・プロパティ属性は変更されません。ただし、srcType、srcLocation および srcName 属性は、export( ) メソッドに渡された srcType、srcLocation および srcName の各パラメータの値で更新されます。export( ) メソッドへのコール後、データベース内部のイメージ・データを管理する必要がない場合は、clearLocal( ) メソッドをコールして、イメージ・データがデータベースの外部に格納されていることを示し、deleteContent( ) メソッドをコールして、データベース BLOB に格納されているイメージ・データを削除します。

データベース内の export( ) メソッドは、ユーザーがアクセス権限を所有しているデータベース・ディレクトリ・オブジェクトに対してのみ書込みを行います。つまり、SQL の CREATE DIRECTORY 文を使用して作成したディレクトリ、または読込み権限を付与されたディレクトリにアクセスできます。また、CREATE DIRECTORY 文を実行するには、CREATE ANY DIRECTORY 権限が必要です。書込み可能なファイルを指定するには、DBMS_JAVA.GRANT_PERMISSION メソッドを使用する必要があります。

たとえば、次の SQL*Plus コマンドは、ユーザー MEDIAUSER にファイル scenery1.jpg の書込み権限を付与します。

```
CALL DBMS_JAVA.GRANT_PERMISSION(
    'MEDIAUSER',
'java.io.FilePermission',
'/images/outdoors/scenery1.jpg',
'write');
```

前述の例は、単一ファイルへの書込み権限を許可する方法を示しています。この他にも、ワイルド・カードを使用した、複数のディレクトリおよびファイル名への書込み権限を許可する様々なパス指定があります。たとえば、スラッシュとアスタリスク（/*）（スラッシュは、オペレーティング・システム固有のファイル・セパレータ文字）で終わるパス指定は、指定されたディレクトリに含まれているすべてのファイルを示します。スラッシュとハイフン（/-）で終わるパス指定は、指定されたディレクトリおよびそのすべてのサブディレクトリに含まれているすべてのファイルを示します。特別なトークン <<ALL FILES>> で構成されたパス名は、すべてのファイルへのアクセス権限を許可します。

セキュリティおよびパフォーマンスの詳細は、『Oracle9*i* Java Developer's Guide』および Java API の java.io.FilePermission クラスを参照してください。必要な権限の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

srcType
コンテンツのエクスポート先のソース・タイプ

srcLocation
コンテンツのエクスポート先のソースの位置

srcName
コンテンツのエクスポート先のソース名

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する export( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
imgObj.export(ctx,"file","IMAGEDIR","image.gif");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- file: コンテンツのエクスポートに使用するソース・プラグインです。
- IMAGEDIR: コンテンツのエクスポート先のデータベース・サーバー上での位置です。
- image.gif: コンテンツのエクスポート先のファイルです。

getBFILE( )

構文

```
public oracle.sql.BFILE getBFILE( )
```

説明

srcType 属性が file の場合、データベースから BFILE ロケータを戻します。このメソッドは、データベース内の対応する getBFILE( ) メソッド（srcLocation および srcName 属性を使用して BFILE を作成）をコールします。

パラメータ

なし

戻り値

oracle.sql.BFILE ロケータを戻します。

例外

java.sql.SQLException

データベース内で対応する getBFILE( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
BFILE imageBFILE = imgObj.getBFILE( );
```

getCompressionFormat( )

構文

```
public String getCompressionFormat( )
```

説明

compressionFormat 属性の値を戻します。

パラメータ

なし

戻り値

compressionFormat 属性の値を、文字列で戻します。

例外

java.sql.SQLException

compressionFormat 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String compression = imgObj.getCompressionFormat( );
```

getContent( )

構文

```
public oracle.sql.BLOB getContent( )
```

説明

localData 属性から BLOB ロケータを戻します。

パラメータ

なし

戻り値

oracle.sql.BLOB ロケータを戻します。

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
BLOB localContent = imgObj.getContent( );
```

getContentFormat( )

構文

```
public String getContentFormat( )
```

説明

contentFormat 属性の値を戻します。

パラメータ

なし

戻り値

contentFormat 属性の値を、文字列で戻します。

例外

java.sql.SQLException

contentFormat 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String format = imgObj.getContentFormat( );
```

getContentLength( )

構文

```
public int getContentLength( )
```

説明

contentLength 属性の値を戻します。

パラメータ

なし

戻り値

contentLength の値を、整数で戻します。

例外

java.sql.SQLException

contentLength 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int length = imgObj.getContentLength( );
```

getDataInByteArray( )

構文

```
public byte[ ] getDataInByteArray( )
```

説明

localData 属性で指定されたデータベース BLOB からのデータを含むバイト配列を戻します。

パラメータ

なし

戻り値

データを含むバイト配列を戻します。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

BLOB からのデータの読込み中にエラーが発生した場合、この例外がスローされます。

java.lang.OutOfMemoryError

データの保持のために十分なメモリーを割当てできない場合、この例外がスローされます。

例

このメソッドの例は、2.3.2.8 項を参照してください。

getDataInFile( )

構文

```
public boolean getDataInFile(String filename)
```

説明

localData 属性で指定されたデータベース BLOB からローカル・ファイルに、データを書き込みます。

パラメータ

filename
データの書込み先のファイル名

戻り値

データが正常にファイルに書き込まれた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

BLOB からのデータの読込み中または出力ファイルへのデータの書込み中にエラーが発生した場合、この例外がスローされます。

例

このメソッドの例は、2.3.2.6 項を参照してください。

getDataInStream( )

構文

```
public InputStream getDataInStream( )
```

説明

localData 属性で指定されたデータベース BLOB のデータの読込み元となる InputStream オブジェクトを戻します。

パラメータ

なし

戻り値

データの読込み元から InputStream オブジェクトを戻します。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

このメソッドの例は、2.3.2.7 項を参照してください。

getFactory( )

構文

```
public static oracle.sql.CustomDatumFactory getFactory( )
```

説明

getCustomDatum( ) メソッドが使用する OrdImage CustomDatumFactory インタフェースを戻します。OracleResultSet または OracleCallableStatement オブジェクトから OrdImage オブジェクトを取り出す場合は、getFactory( ) メソッドを getCustomDatum( ) メソッドの factory パラメータとして指定します。

パラメータ

なし

戻り値

CustomDatumFactory インタフェースの OrdImage 実装を戻します。

例外

なし

例

```
OrdImage img = (OrdImage)rset.getCustomDatum( 1, OrdImage.getFactory() );
```

getFormat( )

構文

```
public String getFormat( )
```

説明

fileFormat 属性の値を戻します。

パラメータ

なし

戻り値

fileFormat 属性の値を、文字列で戻します。

例外

java.sql.SQLException

fileFormat 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String format = imgObj.getFormat( );
```

getHeight( )

構文

```
public int getHeight( )
```

説明

height 属性の値を戻します。

パラメータ

なし

戻り値

height 属性の値を、整数で戻します。

例外

java.sql.SQLException

height 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int height = imgObj.getHeight( );
```

getMimeType( )

構文

```
public String getMimeType( )
```

説明

mimeType 属性の値を戻します。

パラメータ

なし

戻り値

mimeType 属性の値を、文字列で戻します。

例外

java.sql.SQLException

mimeType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String mime = imgObj.getMimeType( );
```

getSource( )

構文

```
public String getSource( )
```

説明

srcType://srcLocation/srcName という形式でソース情報を戻します。

パラメータ

なし

戻り値

ソース情報を、文字列で戻します。

例外

java.sql.SQLException

データベース内で対応する getSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
String sourceName = imgObj.getSource( );
```

getSourceLocation( )

構文

```
public String getSourceLocation( )
```

説明

srcLocation 属性の値を戻します。

パラメータ

なし

戻り値

srcLocation 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcLocation 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String location = imgObj.getSourceLocation( );
```

getSourceName( )

構文

```
public String getSourceName( )
```

説明

srcName 属性の値を戻します。

パラメータ

なし

戻り値

srcName 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcName 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String name = imgObj.getSourceName( );
```

getSourceType( )

構文

```
public String getSourceType( )
```

説明

srcType 属性の値を戻します。

パラメータ

なし

戻り値

srcType 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String type = imgObj.getSourceType( );
```

getUpdateTime( )

構文

```
public java.sql.Timestamp getUpdateTime( )
```

説明

updateTime 属性の値を戻します。

パラメータ

なし

戻り値

updateTime 属性の値を java.sql.Timestamp オブジェクトで戻します。

例外

java.sql.SQLException

upateTime 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
Timestamp time = imgObj.getUpdateTime( );
```

getWidth( )

構文

```
public int getWidth( )
```

説明

width 属性の値を戻します。

パラメータ

なし

戻り値

width 属性の値を、整数で戻します。

例外

java.sql.SQLException

width 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int width = imgObj.getWidth( );
```

importData( )

構文

```
public void importData(byte[ ] [ ] ctx)
```

説明

データを、外部ソースから localData 属性で指定されたデータベース BLOB にインポートします。外部データ・ソースは、srcType、srcLocation および srcName 属性で指定されます。イメージ・データがインポートされると、このメソッドは、デフォルトでデータベース内の setProperties( ) メソッドを自動的にコールし、プロパティ属性を設定します。*inter*Media がサポートしていないフォーマットの外部イメージをインポートする場合は、setFormat( ) メソッドをコールして fileFormat を「other」で始まる文字列に設定し、setProperties( ) メソッドの自動コールを使用禁止にします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する import( ) メソッドまたは setProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
imgObj.importData(ctx)
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。

importFrom( )

構文

```
public void importFrom(byte[ ] [ ] ctx,
  String srcType, String srcLocation, String srcName)
```

説明

データを、外部ソースから localData 属性で指定されたデータベース BLOB にインポートします。外部データ・ソースは、srcType、srcLocation および srcName パラメータで指定されます。srcType、srcLocation および srcName 属性は、importFrom( ) メソッドに渡された srcType、srcLocation および srcName の各パラメータの値で更新されます。イメージ・データがインポートされると、このメソッドは、デフォルトでデータベース内の setProperties( ) メソッドを自動的にコールし、プロパティ属性を設定します。*inter*Media がサポートしていないフォーマットの外部イメージをインポートする場合は、setFormat( ) メソッドをコールして fileFormat を「other」に設定し、setProperties( ) メソッドの自動コールを使用禁止にします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

srcType
データのインポート元のソース・タイプ

srcLocation
データのインポート元のソースの位置

srcName
データのインポート元のソース名

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する importFrom( ) メソッドまたは setProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
imgObj.importFrom(ctx,"file","IMAGEDIR","testimg.dat");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- file: データのインポートに使用するソース・プラグインです。
- IMAGEDIR: データのインポート元のファイルの位置です。
- testimg.dat: データのインポート元のファイルです。

isLocal( )

構文

```
public boolean isLocal( )
```

説明

イメージ・データが、localData 属性で指定されたデータベース内の BLOB でローカルに格納されているかどうかを示します。

パラメータ

なし

戻り値

データベース内の BLOB でデータがローカルに格納されている場合は true、格納されていない場合は false を戻します。

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
if(imgObj.isLocal( ))
     System.out.println("local attribute is true");
else
     System.out.println("local attribute is false");
```

loadDataFromByteArray( )

構文

```
public boolean loadDataFromByteArray(byte[ ] byteArr)
```

説明

データを、バイト配列から localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

byteArr
データのロード元のバイト配列の名前

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中またはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

バイト配列の読込み中にエラーが発生した場合、この例外がスローされます。

例

このメソッドの例は、2.3.2.8 項を参照してください。

loadDataFromFile( )

構文

```
public boolean loadDataFromFile(String filename)
```

説明

データを、ファイルから localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

filename
データのロード元のファイル名

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中またはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

データ・ファイルの読込み中にエラーが発生した場合、この例外がスローされます。

例

このメソッドの例は、2.3.2.6 項を参照してください。

loadDataFromInputStream( )

構文

```
public boolean loadDataFromInputStream(InputStream inpStream)
```

説明

データを、InputStream オブジェクトから localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

inpStream
データのロード元の InputStream オブジェクトの名前

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false を戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中またはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

InputStream オブジェクトの読込み中にエラーが発生した場合、この例外がスローされます。

例

このメソッドの例は、2.3.2.7 項を参照してください。

process( )

構文

```
public void process(String cmd)
```

説明

localData 属性で指定されたデータベース BLOB 内のイメージ・データに対して、イメージ処理操作を 1 回以上実行します。このメソッドは、データベース内の対応する process( ) メソッドをコールして、cmd パラメータで指定されたイメージ処理操作を実行します。

イメージに対して実行できるイメージ処理操作の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

パラメータ

cmd
イメージに対して実行するイメージ処理操作のリストを指定する文字列

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する process( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
imgObj.process("fileFormat=JFIF");
```

パラメータは次のとおりです。

- fileFormat=JFIF: 実行されるコマンドです。

processCopy( )

構文

```
public void processCopy(String cmd, OrdImage dest)
```

説明

イメージ・データを宛先オブジェクトにコピーし、そのイメージ・データに対してイメージ処理操作を 1 回以上実行します。ソース・イメージ・データがデータベースの外部に格納されている場合、このデータは宛先オブジェクト OrdImage の localData 属性で指定されたデータベース BLOB にインポートされます。それ以外の場合、イメージ・データは、現行の OrdImage オブジェクトの localData 属性で指定された BLOB から、宛先オブジェクトの localData 属性で指定された BLOB にコピーされます。

パラメータ

cmd
イメージに対して実行するイメージ処理操作のリストを指定する文字列

dest
宛先 OrdImage オブジェクト

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する processCopy( ) メソッドのコール中にエラーが発生した場合、この例外がスローされます。

例

```
//create and populate an OrdImage object named imgObj2
imgObj.processCopy("maxScale=32 32, fileFormat= GIFF", imgObj2);
```

パラメータは次のとおりです。

- maxScale=32 32, fileFormat= GIFF: 実行されるコマンドです。
- imgObj2: コマンドの実行結果を受け取るオブジェクトです。

setCompressionFormat( )

構文

```
public void setCompressionFormat(String CompressionFormat)
```

説明

compressionFormat 属性の値を設定します。

compressionFormat 属性は、setProperties( ) メソッドによって自動的に特定のメディア・フォーマットに設定されます。setCompressionFormat( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、メディア・データ自体は変更されません。

パラメータ

CompressionFormat
新しい属性値

戻り値

なし

例外

java.sql.SQLException

compressionFormat 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

なし

setContentFormat( )

構文

```
public void setContentFormat(String ContentFormat)
```

説明

contentFormat 属性の値を設定します。

contentFormat 属性は、setProperties( ) メソッドによって自動的に特定のメディア・フォーマットに設定されます。setContentFormat( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、メディア・データ自体は変更されません。

パラメータ

ContentFormat
新しい属性値

戻り値

なし

例外

java.sql.SQLException

contentFormat 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

なし

setContentLength( )

構文

```
public void setContentLength(int ContentLength)
```

説明

contentLength 属性の値を設定します。

contentLength 属性は、setProperties( ) メソッドによって自動的に特定のメディア・フォーマットに設定されます。setContentLength( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、メディア・データ自体は変更されません。

パラメータ

ContentLength
新しい属性値

戻り値

なし

例外

java.sql.SQLException

contentLength 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

なし

setFormat( )

構文

```
public void setFormat(String Format)
```

説明

fileFormat 属性の値を設定します。

fileFormat 属性は、setProperties( ) メソッドによって自動的に特定のメディア・フォーマットに設定されます。setFormat( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、メディア・データ自体は変更されません。importData( ) メソッドおよび importFrom( ) メソッドによって setProperties( ) メソッドの自動コールを使用禁止にするには、fileFormat 属性を「other」で始まる文字列に設定してください。

パラメータ

Format
新しい属性値

戻り値

なし

例外

java.sql.SQLException

fileFormat 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

なし

setHeight( )

構文

```
public void setHeight(int Height)
```

説明

height 属性の値を設定します。

height 属性は、setProperties( ) メソッドによって自動的に特定のイメージ・フォーマットに設定されます。setHeight( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、イメージ・データ自体は変更されません。

パラメータ

Height
新しい属性値

戻り値

なし

例外

java.sql.SQLException

height 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
imgObj.setHeight(24);
```

パラメータは次のとおりです。

- 24: height 属性に設定する値（ピクセル単位）です。

setLocal( )

構文

```
public void setLocal( )
```

説明

イメージ・データが、localData 属性で指定されたデータベース内の BLOB でローカルに格納されているかどうかを示すために、ローカル属性の値を設定します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
imgObj.setLocal( );
```

setMimeType( )

構文

```
public void setMimeType(String mimeType)
```

説明

mimeType 属性の値を設定します。

mimeType 属性は、setProperties( ) メソッドによって自動的に特定のメディア・フォーマットに設定されます。setMimeType( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、メディア・データ自体は変更されません。

パラメータ

mimeType
新しい属性値

戻り値

なし

例外

java.sql.SQLException

mimeType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
imgObj.setMimeType("image/bmp");
```

パラメータは次のとおりです。

- image/bmp: 設定する MIME タイプです。

setProperties( )

構文

```
public void setProperties( )
```

説明

イメージ・データのプロパティを解析し、OrdImage Java オブジェクト内の属性の値を設定します。このメソッドは、height、width、contentLength、fileFormat、contentFormat、compressionFormat および mimeType 属性の値を設定します。対応するプロパティが特定のイメージ・フォーマットについて抽出できない場合、属性は null に設定されます。イメージ・フォーマットが認識できない場合は、SQLException エラーがスローされます。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
imgObj.setProperties( );
```

setProperties(String)

構文

```
public void setProperties(String description)
```

説明

外部イメージの特性を、適切な属性フィールドに書き込みます。このメソッドは、イメージ・プロパティを定義する一連の特性に基づいて、OrdImage オブジェクトの様々な属性の値を設定します。この情報によって、*inter*Media は特定の外部イメージ・フォーマットを処理できます。外部イメージのイメージ特性の設定の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

パラメータ

description
外部イメージに設定するイメージ特性を指定する文字列

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
String properties = "width=123 height=321 compressionformat=NONE
     userString=DJM dataOffset=128 scanlineOrder=INVERSE
     pixelOrder=REVERSE interleaving=BIL numberOfBands=1
     defaultChannelSelection=1";
imgObj.setProperties(properties);
```

パラメータは次のとおりです。

- properties: 設定するプロパティを含む文字列です。

setSource( )

構文

```
public void setSource(String srcType, String srcLocation,
  String srcName)
```

説明

srcType、srcLocation および srcName 属性の値を設定します。

パラメータ

srcType
ソース・タイプ

srcLocation
ソースの位置

srcName
ソースの名前

戻り値

なし

例外

java.sql.SQLException

srcType、srcLocation または srcName 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
imgObj.setSource("file","IMAGEDIR","jdoe.gif");
```

パラメータは次のとおりです。

- file: ソース・タイプです。
- IMAGEDIR: ソースの位置です。
- jdoe.gif: ソースの名前です。

setUpdateTime( )

構文

```
public void setUpdateTime(java.sql.Timestamp currentTime)
```

説明

updateTime 属性の値を設定します。このメソッドは、updateTime 属性の値を指定された時刻に設定します。ただし、currentTime 属性が null に指定されている場合は、updateTime 属性の値をデータベース・サーバーの現在の SYSDATE 時刻に設定します。

パラメータ

currentTime
updateTime 属性の値をデータベース・サーバーの現在の SYSDATE 時刻に設定するために使用する更新時刻（null 値）

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setUpdateTime( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
imgObj.setUpdateTime(null);
```

setWidth( )

構文

```
public void setWidth(int Width)
```

説明

width 属性の値を設定します。

width 属性は、setProperties( ) メソッドによって自動的に特定のイメージ・フォーマットに設定されます。setWidth( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、イメージ・データ自体は変更されません。

パラメータ

Width
新しい属性値

戻り値

なし

例外

java.sql.SQLException

width 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
imgObj.setWidth(24);
```

パラメータは次のとおりです。

- 24: width 属性に設定する値（ピクセル単位）です。

6

OrdImageSignature リファレンス情報

OrdImageSignature クラスは、Java アプリケーションで ORDSYS.ORDImageSignature データベース型のインスタンスを表すために使用されます。OrdImageSignature クラスには、2 つのイメージ・シグネチャを比較するための static メソッドおよびイメージ・シグネチャを生成するためのメソッドがあります。

アプリケーションによって OrdImageSignature オブジェクトが変更される場合は、これらの変更を永続的なものにするために、SQL 更新操作を実行する必要があります。

注意： 基礎となる表のイメージ・シグネチャ索引とともにイメージ・マッチングを使用してパフォーマンスを向上させるには、IMGSimilar および IMGScore の SQL 操作を使用してください。

イメージ・シグネチャ索引の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

6.1 前提条件

*inter*Media メソッドを実行するために、次のインポート文を Java ファイルに組み込む必要があります。

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
import oracle.ord.im.*;
```

この章で示す例は、次の操作がすでに実行済であることを示しています。

- OrdImageSignature 型の列を含む表への接続が確立されている。
- ローカル OrdImageSignature オブジェクトとして matchObj が作成され、データが移入されている。

6.2 リファレンス情報

この項では、OrdImageSignature オブジェクトを操作するメソッドについて説明します。

evaluateScore( )

構文

```
public static float evaluateScore(OrdImageSignature signature1,
  OrdImageSignature signature2, String attrWeights)
```

説明

2 つのイメージ・シグネチャを比較し、イメージ・シグネチャ間の差異の程度を示すスコアを戻します。このメソッドは、1 つ以上の視覚属性に指定された重みを使用して、signature1 および signature2 のイメージ・シグネチャを比較し、0.0 ～ 100.0 の範囲のスコアを戻します。値が小さいほど、一致度が高いことを示します。

color、shape、texture および location のうちの 1 つ以上の視覚属性に対して、0.0 ～ 1.0 の重みを指定します。

color、shape または texture のうち 1 つ以上の属性に、0.0 より大きい値を指定する必要があります。location 属性は、イメージの色、形またはテクスチャの分布の重要度を示しています。処理時に、これらの値は全体で 1.0 になるように正規化されます。次に例を示します。

```
color=0.7,shape=0.3
```

> **注意：** ORDImageSignature evaluateScore( ) メソッドが操作するのは、データベース表の索引ではなく、2 つのイメージ・シグネチャです。そのため、このメソッドでは、基礎となる表のイメージ・シグネチャ索引とともにイメージ・マッチングを使用して、パフォーマンスを向上させることはできません。イメージ・マッチング索引を使用するには、IMGSimilar および IMGScore の SQL 操作を使用してください。
>
> イメージ・シグネチャ索引の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

パラメータ

signature1
1 つ目の OrdImageSignature

signature2
signature1 と比較する OrdImageSignature

attrWeights
1 つ以上の視覚属性のリストおよび各属性に適用する重みを指定する文字列

戻り値

スコアを、浮動小数点値で戻します。

例外

java.sql.SQLException

データベース内で対応する evaluateScore( ) メソッドのコール中にエラーが発生した場合、この例外がスローされます。

例

```
float score = matchObj.evaluateScore(signature1, signature2, "color=1.0");
```

パラメータは次のとおりです。

- signature1: 1 つ目のシグネチャです。
- signature2: 2 つ目のシグネチャです。signature1 と比較されます。
- color=1.0: 色視覚属性に適用する重み値です。

generateSignature( )

構文

```
public void generateSignature(OrdImage img)
```

説明

指定されたイメージのイメージ・シグネチャを生成します。

パラメータ

img
シグネチャの生成元となる OrdImage オブジェクト

戻り値

なし

例外

SQLException

データベース内で対応する generateSignature( ) メソッドのコール中にエラーが発生した場合、この例外がスローされます。

例

```
matchObj.generateSignature(imgObj);
```

パラメータは次のとおりです。

- imgObj: シグネチャが生成されるイメージ・オブジェクトです。

getFactory( )

構文

```
public static oracle.sql.CustomDatumFactory getFactory( )
```

説明

getCustomDatum( ) メソッドが使用する OrdImageSignature CustomDatumFactory インタフェースを戻します。OracleResultSet または OracleCallableStatement オブジェクトから OrdImageSignature オブジェクトを取り出す場合は、getFactory( ) メソッドを getCustomDatum( ) メソッドの factory パラメータとして指定します。

パラメータ

なし

戻り値

CustomDatumFactory インタフェースの OrdImageSignature 実装を戻します。

例外

なし

例

```
OrdImageSignature sig = (OrdImageSignature)rset.getCustomDatum( 1,
OrdImageSignature.getFactory() );
```

isSimilar( )

構文

```
public static int isSimilar(OrdImageSignature signature1,
  OrdImageSignature signature2, String attrWeights, float threshold)
```

説明

2 つのイメージ・シグネチャを比較し、イメージ・シグネチャ間の差異の程度が指定されたしきい値の範囲内かどうかを示すステータスを戻します。このメソッドは、1 つ以上の視覚属性に指定された重みを使用して、signature1 および signature2 のイメージ・シグネチャを比較し、0.0 ～ 100.0 の範囲のスコアを戻します。値が小さいほど、一致度が高いことを示します。スコアが指定されたしきい値以下の場合は、イメージが一致していると判断し、1 を戻します。それ以外の場合は、0 を戻します。

color、shape、texture および location のうちの 1 つ以上の視覚属性に対して、0.0 ～ 1.0 の重みを指定します。

color、shape または texture のうち 1 つ以上の属性に、0.0 より大きい値を指定する必要があります。location 属性は、イメージの色、形またはテクスチャの分布の重要度を示しています。処理時に、これらの値は全体で 1.0 になるように正規化されます。次に例を示します。

```
color=0.7,shape=0.3
```

> **注意：** ORDImageSignature isSimilar( ) メソッドが操作するのは、データベース表の索引ではなく、2 つのイメージ・シグネチャです。そのため、このメソッドでは、基礎となる表のイメージ・シグネチャ索引とともにイメージ・マッチングを使用して、パフォーマンスを向上させることはできません。イメージ・マッチング索引を使用するには、IMGSimilar および IMGScore の SQL 操作を使用してください。
>
> イメージ・シグネチャ索引の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

パラメータ

signature1
1 つ目の OrdImageSignature

signature2
signature1 と比較する OrdImageSignature

attrWeights
1 つ以上の視覚属性のリストおよび各属性に適用する重みを指定する文字列

threshold
2 つのイメージが一致していると判断するために必要な一致度を指定する浮動小数点値

戻り値

イメージが一致している場合は、整数値 1 を戻します。それ以外の場合は、0 を戻します。

例外

java.sql.SQLException

データベース内で対応する isSimilar( ) メソッドのコール中にエラーが発生した場合、この例外がスローされます。

例

```
int i = matchObj.isSimilar(signature1, signature2, "color=0.5,shape=0.5", 10);
```

パラメータは次のとおりです。

- signature1: 1 つ目のシグネチャです。
- signature2: 2 つ目のシグネチャです。signature1 と比較されます。
- color=0.5, shape=0.5: 各視覚属性に適用する重み値です。
- 10: スコアがこの値以下の場合、一致していると判断されます。

7

JAI 入力ストリームおよび出力ストリームのリファレンス情報

この章では、Java Advanced Imaging（JAI）で使用可能な、*inter*Media Java Classes が BLOB および BFILE データへのインタフェースとして提供する、3 つのタイプのストリーム・オブジェクトについて説明します。

- BfileInputStream。ストリームに対応付けられた Oracle BFILE からデータを読み込むことができます。
- BlobInputStream。ストリームに対応付けられた Oracle BLOB からデータを読み込むことができます。
- BlobOutputStream。ストリームに対応付けられた Oracle BLOB にバッファのデータを書き込むことができます。

7.1 前提条件

JAI ストリーム・オブジェクトを使用するために、次のインポート文を Java ファイルに組み込む必要があります。

```
import oracle.sql.BLOB;
import oracle.sql.BFILE;
import oracle.ord.media.jai.io.*;
```

BfileInputStream オブジェクト

この項では、BFILE からデータを読み込むための JAI へのインタフェースを提供する、BfileInputStream オブジェクトに対応付けられたメソッドについてのリファレンス情報を示します。このオブジェクトは、com.sun.media.jai.codec.SeekableStream および java.io.InputStream のサブクラスで、java.io.DataInput を実装しています。

この項で示す例は、次の操作がすでに実行済であることを前提にしています。

- 次のインポート文が組み込まれている。

```
import javax.media.jai.JAI;
import java.awt.image.RenderedImage;
```

- ローカル BfileInputStream オブジェクトとして inStream が作成されている。

BfileInputStream(BFILE)

構文

```
public BfileInputStream(oracle.sql.BFILE bfile)
```

説明

指定された BFILE から読込みを行う BfileInputStream オブジェクトを作成します。コンストラクタは、BFILE に定義された最大チャンク・サイズを使用します。BFILE は、このコンストラクタの実行後に、オープンします。

パラメータ

bfile
データの読込み元の BFILE

戻り値

なし

例外

java.io.IOException

java.sql.SQLException

例

```
BfileInputStream inStream = new BfileInputStream(bfile);
RenderedImage image = JAI.create("stream",inStream);
```

BfileInputStream(BFILE, int)

構文

```
public BfileInputStream(oracle.sql.BFILE bfile, int chunkSize)
```

説明

指定された BFILE から読込みを行う BfileInputStream オブジェクトを作成します。コンストラクタは、指定されたチャンク・サイズを使用します。BFILE は、このコンストラクタの実行後に、オープンします。

パラメータ

bfile
データの読込み元の BFILE

chunkSize
BFILE から一度に読み込むデータの最大サイズ

戻り値

なし

例外

java.io.IOException

java.sql.SQLException

例

```
BfileInputStream inStream = new BfileInputStream(bfile,4096);
RenderedImage image = JAI.create("stream",inStream);
```

パラメータは次のとおりです。

- 4096: 一度に読み込む最大バイト数です。

canSeekBackwards( )

構文

```
public boolean canSeekBackwards( )
```

説明

ストリームが、逆順読込み可能かどうかを確認します。BfileInputStream オブジェクトは逆順読込みが可能であるため、このメソッドは常に true を戻します。

パラメータ

なし

戻り値

true を戻します。

例外

なし

例

なし

close( )

構文

```
public void close( )
```

説明

BfileInputStream をクローズして、使用中のリソースを解放します。BFILE は、ストリームがクローズした後に自動的にクローズします。

パラメータ

なし

戻り値

なし

例外

java.io.IOException

例

```
inStream.close( )
```

getBFILE( )

構文

```
public oracle.sql.BFILE getBFILE( )
```

説明

BfileInputStream に対応付けられた BFILE を戻します。

パラメータ

なし

戻り値

BfileInputStream に対応付けられた BFILE を戻します。

例外

なし

例

```
BFILE imageBFILE = inStream.getBFILE( );
```

getFilePointer( )

構文

```
public long getFilePointer( )
```

説明

次の読込みを行うために、BFILE の先頭からのオフセットを戻します。

パラメータ

なし

戻り値

次の読込みを行うために、BFILE の先頭からのオフセットを、バイト単位で戻します。

例外

java.io.IOException

例

```
long offset = inStream.getFilePointer( );
```

mark( )

構文

```
public void mark(int readLimit)
```

説明

BfileInputStream 内の現在の位置をマークします。reset( ) メソッドをコールすると、BfileInputStream 内の最後にマークされた位置が戻されます。

パラメータ

readLimit
この引数は無視されます。

戻り値

なし

例外

なし

例

```
inStream.mark(1);
```

markSupported( )

構文

```
public boolean markSupported( )
```

説明

BfileInputStream が、マーク付けをサポートしているかどうかを確認します。BfileInputStream オブジェクトはマーク付けをサポートしているため、このメソッドは常に true を戻します。

パラメータ

なし

戻り値

true を戻します。

例外

なし

例

なし

read( )

構文

```
public int read( )
```

説明

BfileInputStream に対応付けられた BFILE からシングル・バイトを読み込みます。

パラメータ

なし

戻り値

読み込んだデータのバイト数を戻すか、または BFILE の終わりに達した場合は -1 を戻します。

例外

java.io.IOException

例

```
int i = inStream.read( );
```

read(byte[ ])

構文

```
public int read(byte[ ] buffer)
```

説明

指定されたバッファに BFILE のデータを読み込みます。

パラメータ

buffer
データの読込み先バッファ

戻り値

バッファに読み込んだバイト数を戻すか、またはデータを読み込む前に BFILE の終わりに達した場合は -1 を戻します。値は、バッファ長を超えることはありません。

例外

java.io.IOException

例

```
byte[ ] buffer = new byte[4000];
int i = inStream.read(buffer);
```

パラメータは次のとおりです。

- buffer: データの読込み先バッファです。

read(byte[ ], int, int)

構文

```
public int read(byte[ ]buffer, int off, int len)
```

説明

BFILE のデータを、指定されたバッファの指定されたオフセットから、指定されたデータ長（最大で）分読み込みます。

パラメータ

buffer
データの読込み先バッファ

off
バッファの先頭からのオフセット。この位置からデータがバイト単位で書き込まれます。

len
バッファに読み込む最大バイト数

戻り値

読み込んだバイト数を戻すか、またはデータを読み込む前に BFILE の終わりに達した場合は -1 を戻します。値は、バッファ長を超えることはありません。

例外

java.io.IOException

例

```
byte[ ] buffer = new byte[4000];
int i = inStream.read(buffer,75,50);
```

パラメータは次のとおりです。

- buffer: データの読込み先バッファです。
- 75: バッファの先頭からのオフセットで、この位置から読込みを開始します。
- 50: バッファに読み込むバイト数です。

remaining( )

構文

```
public long remaining( )
```

説明

BFILE 内の、まだ読み込んでいないバイト数を戻します。

パラメータ

なし

戻り値

BFILE 内の、読み込んでいないバイト数を戻します。

例外

なし

例

```
long remain = inStream.remaining( );
```

reset( )

構文

```
public void reset( )
```

説明

ストリームを最後の有効マークの位置に戻します。

パラメータ

なし

戻り値

なし

例外

java.io.IOException

例

```
inStream.reset( );
```

seek( )

構文

```
public void seek(long pos)
```

説明

次の読込みを行うために、BFILE の先頭からのオフセットを設定します。

パラメータ

pos
BFILE の先頭からのオフセット。この位置から次の読込みを行います。

戻り値

なし

例外

java.io.IOException

例

```
inStream.seek(75);
```

パラメータは次のとおりです。

- 75: BFILE の先頭からのオフセットで、この位置から次の読込みを行います。

skip( )

構文

```
public long skip(long n)
```

説明

BFILE 内の指定されたバイト数分のスキップを試行します。

スキップするバイト数は、指定された数より小さい場合があります。たとえば、ファイルの終わりに達した場合、数は小さくなります。

パラメータ

n
スキップするバイト数

戻り値

実際にスキップしたバイト数を戻します。

例外

java.io.IOException

例

```
long skipped = inStream.skip(100);
```

パラメータは次のとおりです。

- 100: スキップするバイト数です。

BlobInputStream オブジェクト

この項では、BLOB からデータを読み込むための JAI へのインタフェースを提供する、BlobInputStream オブジェクトに対応付けられたメソッドについてのリファレンス情報を示します。このオブジェクトは、com.sun.media.jai.codec.SeekableStream および java.io.InputStream のサブクラスで、java.io.DataInput を実装しています。

この項で示す例は、次の操作がすでに実行済であることを前提にしています。

- 次のインポート文が組み込まれている。

```
import javax.media.jai.JAI;
import java.awt.image.RenderedImage;
```

- ローカル BlobInputStream オブジェクトとして inStream が作成されている。

BlobInputStream(BLOB)

構文

```
public BlobInputStream(oracle.sql.BLOB blob)
```

説明

指定された BLOB から読み込む BlobInputStream オブジェクトを作成します。コンストラクタは、データベースによって決定される最適なチャンク・サイズを使用します。

パラメータ

blob
データの読込み元の BLOB

戻り値

なし

例外

java.io.IOException

java.sql.SQLException

例

```
BlobInputStream inStream = new BlobInputStream(blob);
RenderedImage image = JAI.create("stream",inStream);
```

BlobInputStream(BLOB, int)

構文

```
public BlobInputStream(oracle.sql.BLOB blob, int chunkSize)
```

説明

指定された BLOB から読み込む BlobInputStream オブジェクトを作成します。コンストラクタは、指定されたチャンク・サイズを使用します。

パラメータ

blob
データの読込み元の BLOB

chunkSize
BLOB から一度に読み込むデータの最大サイズ

戻り値

なし

例外

java.io.IOException

java.sql.SQLException

例

```
BlobInputStream inStream = new BlobInputStream(blob,4096);
RenderedImage image = JAI.create("stream",inStream);
```

パラメータは次のとおりです。

- 4096: 一度に読み込む最大バイト数です。

canSeekBackwards( )

構文

```
public boolean canSeekBackwards( )
```

説明

ストリームが、逆順読込み可能かどうかを確認します。BlobInputStream オブジェクトは逆順読込みが可能であるため、このメソッドは常に true を戻します。

パラメータ

なし

戻り値

true を戻します。

例外

なし

例

なし

close( )

構文

```
public void close( )
```

説明

BlobInputStream をクローズして、使用中のリソースを解放します。

パラメータ

なし

戻り値

なし

例外

java.io.IOException

例

```
inStream.close( )
```

getBLOB( )

構文

```
public oracle.sql.BLOB getBLOB( )
```

説明

BlobInputStream に対応付けられた BLOB を戻します。

パラメータ

なし

戻り値

BlobInputStream に対応付けられた BLOB を戻します。

例外

なし

例

```
BLOB imageBLOB = inStream.getBLOB( );
```

getFilePointer( )

構文

```
public long getFilePointer( )
```

説明

次の読込みを行うために、BLOB の先頭からのオフセットを戻します。

パラメータ

なし

戻り値

次の読込みを行うために、BLOB の先頭からのオフセットを、バイト単位で戻します。

例外

java.io.IOException

例

```
long offset = inStream.getFilePointer( );
```

mark( )

構文

```
public void mark(int readLimit)
```

説明

BlobInputStream 内の現在の位置をマークします。reset( ) メソッドをコールすると、BlobInputStream 内の最後にマークされた位置が戻されます。

パラメータ

readLimit
この引数は無視されます。

戻り値

なし

例外

なし

例

```
inStream.mark(1);
```

markSupported( )

構文

```
public boolean markSupported( )
```

説明

BlobInputStream が、マーク付けをサポートしているかどうかを確認します。BlobInputStream オブジェクトはマーク付けをサポートしているため、このメソッドは常に true を戻します。

パラメータ

なし

戻り値

true を戻します。

例外

なし

例

なし

read( )

構文

```
public int read( )
```

説明

BlobInputStream に対応付けられた BLOB からシングル・バイトを読み込みます。

パラメータ

なし

戻り値

読み込んだデータのバイト数を戻すか、または BLOB の終わりに達した場合は -1 を戻します。

例外

java.io.IOException

例

```
int i = inStream.read( );
```

read(byte[ ])

構文

```
public int read(byte[ ] buffer)
```

説明

指定されたバッファに BLOB のデータを読み込みます。

パラメータ

buffer
データの読込み先バッファ

戻り値

バッファに読み込んだバイト数を戻すか、またはデータを読み込む前に BLOB の終わりに達した場合は -1 を戻します。値は、バッファ長を超えることはありません。

例外

java.io.IOException

例

```
byte[ ] buffer = new byte[4000];
int i = inStream.read(buffer);
```

パラメータは次のとおりです。

- buffer: データの読込み先バッファです。

read(byte[ ], int, int)

構文

```
public int read(byte[ ]buffer, int off, int len)
```

説明

BLOB のデータを、指定されたバッファの指定されたオフセットから、指定されたデータ長（最大で）分読み込みます。

パラメータ

buffer
データの読込み先バッファ

off
バッファの先頭からのオフセット。この位置からデータがバイト単位で書き込まれます。

len
バッファに書き込まれる最大バイト数

戻り値

バッファに読み込んだバイト数を戻すか、または BLOB の終わりに達した場合は -1 を戻します。値は、バッファ長を超えることはありません。

例外

java.io.IOException

例

```
byte[ ] buffer = new byte[4000];
int i = inStream.read(buffer,75,50);
```

パラメータは次のとおりです。

- buffer: データの読込み先バッファです。
- 75: バッファの先頭からのオフセットで、この位置からデータが書き込まれます。
- 50: バッファに読み込むバイト数です。

remaining( )

構文

```
public long remaining( )
```

説明

BLOB 内の、まだ読み込んでいないバイト数を戻します。

パラメータ

なし

戻り値

BLOB 内の、読み込んでいないバイト数を戻します。

例外

なし

例

```
long remain = inStream.remaining( );
```

reset( )

構文

```
public void reset( )
```

説明

ストリームを最後の有効マークの位置に戻します。

パラメータ

なし

戻り値

なし

例外

java.io.IOException

例

```
inStream.reset( );
```

seek( )

構文

```
public void seek(long pos)
```

説明

次の読込みを行うために、BLOB の先頭からのオフセットを設定します。

パラメータ

pos
BLOB の先頭からのオフセット。この位置から次の読込みを行います。

戻り値

なし

例外

java.io.IOException

例

```
inStream.seek(75);
```

パラメータは次のとおりです。

- 75: BLOB の先頭からのオフセットで、この位置から次の読込みを行います。

skip( )

構文

```
public long skip(long n)
```

説明

BLOB 内の指定されたバイト数分のスキップを試行します。

スキップするバイト数は、指定された数より小さい場合があります。たとえば、ファイルの終わりに達した場合、数は小さくなります。

パラメータ

n
スキップするバイト数

戻り値

実際にスキップしたバイト数を戻します。

例外

java.io.IOException

例

```
long skipped = inStream.skip(100);
```

パラメータは次のとおりです。

- 100: スキップするバイト数です。

BlobOutputStream オブジェクト

この項では、BLOB にデータを書き込むための JAI へのインタフェースを提供する、BlobOutputStream オブジェクトに対応付けられたメソッドについてのリファレンス情報を示します。このオブジェクトは、java.io.OutputStream のサブクラスです。

この項で示す例は、次の操作がすでに実行済であることを前提にしています。

- 次のインポート文が組み込まれている。

```
import javax.media.jai.JAI;
import java.awt.image.RenderedImage;
```

- ローカル BlobOutputStream オブジェクトとして outStream が作成されている。

BlobOutputStream(BLOB)

構文

```
public BlobOutputStream(oracle.sql.BLOB blob)
```

説明

データベースによって決定される最適なチャンク・サイズを使用して、指定された BLOB に書き込む BlobOutputStream オブジェクトを作成します。このタイプのオブジェクトを作成すると、BLOB 内のデータが長さ 0 に暗黙的に切り捨てられます。

パラメータ

blob
データの書込み先の BLOB

戻り値

なし

例外

java.io.IOException

java.sql.SQLException

例

```
RenderedImage = JAI.create("fileload","sample.jpg");
BlobOutputStream outStream = new BlobOutputStream(blob);
JAI.create("encode",image,"bmp")
```

BlobOutputStream(BLOB, int)

構文

```
public BlobOutputStream(oracle.sql.BLOB blob, int chunkSize)
```

説明

指定された整数を最大チャンク・サイズとして使用して、指定された BLOB に書き込む BlobOutputStream オブジェクトを作成します。このタイプのオブジェクトを作成すると、BLOB 内のデータが長さ 0 に暗黙的に切り捨てられます。

パラメータ

blob
データの書込み先の BLOB

chunkSize
BLOB に一度に書き込むデータの最大サイズ

戻り値

なし

例外

java.io.IOException

java.sql.SQLException

例

```
RenderedImage = JAI.create("fileload","sample.jpg");
BlobOutputStream outStream = new BlobOutputStream(blob,4096);
JAI.create("encode",image,"bmp")
```

パラメータは次のとおりです。

- 4096: 一度に書き込むデータの最大サイズです。

close( )

構文

```
public void close( )
```

説明

出力ストリームをクローズして、ストリームに対応付けられたシステム・リソースを解放します。このメソッドは、自動的に flush( ) をコールして BLOB にバッファリング・バイトを書き込んだ後、ストリームをクローズします。

パラメータ

なし

戻り値

なし

例外

java.io.IOException

例

```
outStream.close( );
```

flush( )

構文

```
public void flush( )
```

説明

出力ストリームをフラッシュして、バッファリング出力バイトを BLOB に強制的に書き込みます。

パラメータ

なし

戻り値

なし

例外

java.io.IOException

例

```
outStream.flush( );
```

getFilePointer( )

構文

```
public long getFilePointer( )
```

説明

次の書込みを行うために、BLOB の先頭からのオフセットを戻します。

パラメータ

なし

戻り値

次の書込みを行うために、BLOB の先頭からのオフセットを、バイト単位で戻します。

例外

java.io.IOException

例

```
long offset = outStream.getFilePointer( );
```

length( )

構文

```
public long length( )
```

説明

現在の出力ストリーム長を戻します。

パラメータ

なし

戻り値

現在の出力ストリーム長を戻します。

例外

java.io.IOException

例

```
long length = outStream.length( );
```

seek( )

構文

```
public void seek(long pos)
```

説明

次の書込みを行うために、ストリームの先頭からのファイル・ポインタのオフセットを設定します。

オフセットは、ストリームの終端を超えて設定される場合があります。オフセットをストリームの終わりを超えて設定しても、ストリーム長は変更されません。ストリーム長が変更されるのは、オフセットがストリームの終わりを超えて設定された後に書き込まれた場合のみです。

パラメータ

pos
ストリームの先頭から測定されたオフセットの位置（バイト単位）。この位置にファイル・ポインタを設定します。

戻り値

なし

例外

java.io.IOException

例

```
outStream.seek(4096);
```

write(byte[ ])

構文

```
public void write(byte[ ] buffer)
```

説明

指定されたバイト配列内のすべてのバイトを BLOB に書き込みます。

パラメータ

buffer
BLOB に書き込むバイト配列

戻り値

なし

例外

java.io.IOException

例

```
//create a byte array named buffer and populate it with data
outStream.write(buffer);
```

パラメータは次のとおりです。

- buffer: BLOB に書き込むバイト配列です。

write(byte[ ], int, int)

構文

```
public void write(byte[ ] buffer, int off, int len)
```

説明

指定されたバイト配列の指定されたバイト数を BLOB に書き込みます。

パラメータ

buffer
BLOB に書き込むデータを含むバッファ

off
バッファ内の開始オフセット

len
BLOB に書き込むバイト数

戻り値

なし

例外

java.io.IOException

例

```
//create a byte array named buffer and populate it with data
outStream.write(buffer,75,50);
```

パラメータは次のとおりです。

- buffer: BLOB に書き込むバイト配列です。
- 75: バッファの先頭からのオフセットで、この位置からデータを読み込みます。
- 50: 書込みバイト数です。

write(int)

構文

```
public void write(int b)
```

説明

指定されたバイトを BLOB に書き込みます。

パラメータ

b
BLOB に書き込むバイト。下位バイトのみを書き込みます。上位 24 ビットは無視されます。

戻り値

なし

例外

java.io.IOException

例

```
outStream.write(50);
```

パラメータは次のとおりです。

- 50: 下位バイトが BLOB に書き込まれる整数です。

8

OrdVideo リファレンス情報

OrdVideo クラスは、Java アプリケーションで ORDSYS.ORDVideo データベース型のインスタンスを表すために使用されます。OrdVideo クラスには、OrdVideo Java オブジェクトに対して様々な操作を行う一連のメソッドおよび様々なオブジェクト属性を取得し、設定するための一連のメソッドがあります。

ほぼすべてのメソッドで、アプリケーション内の OrdVideo Java オブジェクトの属性が操作されます。読込みまたは書込みを目的としてビデオ・データにアクセスするメソッドは例外です。例外のメソッドには、次のものがあります。

- localData 属性で指定されたデータベース BLOB を操作し、データベース BLOB 内に格納されたデータの読込みおよび書込みを行うメソッド
- srcType 属性が file の場合に srcLocation および srcName 属性で指定されたデータベース BFILE を操作し、データベース・サーバー内の指定されたファイルからデータを読み込むメソッド
- srcType 属性が http の場合に srcType、srcLocation および srcName 属性で指定された URL を操作し、指定された URL でリソースからデータを読み込むメソッド

アプリケーションによって、データベース内の OrdVideo Java オブジェクトまたはビデオ・データが変更される場合は、これらの変更を永続的なものにするために、データベース内の ORDVideo SQL オブジェクトを更新する必要があります。

OrdVideo Java クラスの一部のメソッドは、データベース・ソース・プラグインまたはデータベース・フォーマット・プラグインに渡されて処理され、コンテキスト・パラメータに `byte [ ] [ ] ctx` を取ります。ソース・プラグインまたはフォーマット・プラグインで必要になる可能性のあるコンテキスト情報を保持するには、アプリケーションで 64 バイトの配列を割り当てる必要があります。たとえば、プラグインでは、一度のコールでコンテキスト情報を初期化し、その情報を後続のコールで使用できます。配列は、ソース・プラグインのコンテキストおよびフォーマット・プラグインのコンテキストに 1 つずつ必要です。ほとんどのプラグインは、64 バイトで十分ですが、ユーザー定義のプラグインによっては、追加領域が必要になる場合もあります。次の例では、プラグインのコンテキスト情報配列の割当て方法を示します。

```
byte [] [] ctx = new byte[1][64];
```

注意： 今回のリリースでは、オラクル社が提供するソース・プラグインまたはフォーマット・プラグインでは、コンテキストは保持されません。また、ユーザー定義のソース・プラグインまたはフォーマット・プラグインでも、コンテキストが保持されない場合もあります。前述した方法でコンテキスト・パラメータを指定しておくと、アプリケーションは、現行または将来のソース・プラグインまたはフォーマット・プラグインで動作します。

プラグインの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

8.1 前提条件

*inter*Media メソッドを実行するために、次のインポート文を Java ファイルに組み込む必要があります。

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
import oracle.ord.im.*;
```

この章で示す例は、次の操作がすでに実行済であることを前提にしています。

- OrdVideo 型の列を含む表への接続が確立されている。
- ローカル OrdVideo オブジェクトとして vidObj が作成され、データが移入されている。

接続の確立およびローカル・オブジェクトの移入の例については、2.4.2 項を参照してください。

8.2 リファレンス情報

この項では、OrdVideo オブジェクトを操作するメソッドについて説明します。

checkProperties( )

構文

```
public boolean checkProperties(byte[ ] [ ] ctx)
```

説明

ビデオ・データのプロパティが、OrdVideo Java オブジェクトの属性と一貫性があるかどうかを確認します。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

戻り値

ビデオ・データのプロパティが、OrdVideo Java オブジェクトの属性と一貫性がある場合は true、一貫性がない場合は false を戻します。

例外

java.sql.SQLException

データベース内で対応する checkProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
if(vidObj.checkProperties(ctx))
     System.out.println("checkProperties successful");
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。

clearLocal( )

構文

```
public void clearLocal( )
```

説明

ローカル属性を消去して、ビデオ・データが外部に格納されていることを示します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

ローカル属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.clearLocal( );
```

closeSource( )

構文

```
public int closeSource(byte[ ] [ ] ctx)
```

説明

データ・ソースをクローズします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

戻り値

ステータスを、整数で戻します。0（ゼロ）は成功を示し、0（ゼロ）以外の値はソース・プラグイン固有の障害コードを示します。

例外

java.sql.SQLException

データベース内で対応する closeSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
int i = vidObj.closeSource(ctx);
if(i == 0)
     System.out.println("closeSource successful");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。

deleteContent( )

構文

```
public void deleteContent( )
```

説明

localData 属性によって指定されたデータベース BLOB に格納されているデータを削除します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する deleteContent( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.deleteContent( );
```

export( )

構文

```
public void export (byte[ ] [ ] ctx, String srcType,
  String srcLocation, String srcName)
```

説明

localData 属性で指定された BLOB からデータをエクスポートします。このメソッドは、データベース内の対応する export( ) メソッドをコールして、srcType、srcLocation および srcName の各パラメータで指定された位置に、ビデオ・データをエクスポートします。

ただし、export( ) メソッドをサポートしないソース・プラグインもあります。file ソース・タイプのみがネイティブにサポートされています。

このメソッドは、Oracle データベース・サーバーのリリース 8.1.7 以上で実行している場合のみ機能します。

後半部分では、export( ) メソッドおよびオラクル社が提供する file ソース・プラグインの使用方法について説明します。ユーザー定義のプラグインの動作は異なります。

file ソース・プラグインによって実装された export( ) メソッドは、localData 属性で指定されたデータベース BLOB 内に格納されているビデオ・データのコピーのみ行い、変更は行いません。

ビデオ・データのエクスポート後も、すべてのビデオ・プロパティ属性は変更されません。ただし、srcType、srcLocation および srcName 属性は、export( ) メソッドに渡された srcType、srcLocation および srcName の各パラメータの値で更新されます。export( ) メソッドへのコール後、データベース内部のビデオ・データを管理する必要がない場合は、clearLocal( ) メソッドをコールして、ビデオ・データがデータベースの外部に格納されていることを示し、deleteContent( ) メソッドをコールして、データベース BLOB に格納されているビデオ・データを削除します。

データベース内の export( ) メソッドは、ユーザーがアクセス権限を所有しているデータベース・ディレクトリ・オブジェクトに対してのみ書込みを行います。つまり、SQL の CREATE DIRECTORY 文を使用して作成したディレクトリ、または読込み権限を付与されたディレクトリにアクセスできます。CREATE DIRECTORY 文を実行するには、CREATE ANY DIRECTORY 権限が必要です。また、書込み可能なファイルを指定するには、DBMS_JAVA.GRANT_PERMISSION メソッドを使用する必要があります。

たとえば、次の SQL*Plus コマンドは、ユーザー MEDIAUSER にファイル movie1.mov の書込み権限を付与します。

```
CALL DBMS_JAVA.GRANT.PERMISSION(
    'MEDIAUSER',
    'java.io.FilePermission',
    '/videos/filmclips/movie1.mov',
    'write');
```

前述の例は、単一ファイルへの書込み権限を許可する方法を示しています。この他にも、ワイルド・カードを使用した、複数のディレクトリおよびファイル名への書込み権限を許可する様々なパス指定があります。たとえば、スラッシュとアスタリスク（/*）（スラッシュは、オペレーティング・システム固有のファイル・セパレータ文字）で終わるパス指定は、指定されたディレクトリに含まれているすべてのファイルを示します。スラッシュとハイフン（/-）で終わるパス指定は、指定されたディレクトリおよびそのすべてのサブディレクトリに含まれているすべてのファイルを示します。特別なトークン <<ALL FILES>> で構成されたパス名は、すべてのファイルへのアクセス権限を許可します。

セキュリティとパフォーマンスの詳細は、『Oracle9*i* Java Developer's Guide』および Java API の java.io.FilePermission クラスを参照してください。必要な権限の詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

srcType
コンテンツのエクスポート先のソース・タイプ

srcLocation
コンテンツのエクスポート先のソースの位置

srcName
コンテンツのエクスポート先のソース名

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する export( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
vidObj.export(ctx,"file","VIDEODIR","complete.dat");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- file: コンテンツのエクスポートに使用するソース・プラグインです。
- VIDEODIR: コンテンツのエクスポート先の位置です。
- complete.dat: コンテンツのエクスポート先のファイルです。

getAllAttributes( )

構文

```
public oracle.sql.CLOB getAllAttributes(byte[ ] [ ] ctx)
```

説明

一時 CLOB 内のビデオ・プロパティの値を、フォーマット・プラグインで定義された形式で戻します。ネイティブにサポートされているフォーマットの場合、これらの情報は、カンマで区切られた属性のリストとして表示されます。属性のリストは、attributeName=attributeValue という形式で示されます。このリストには、format、mimeType、width、height、frameResolution、frameRate、videoDuration、numberOfFrames、compressionType、numberOfColors および bitRate 属性が含まれます。ユーザーが定義したフォーマットの場合、これらの情報は、フォーマット・プラグインで定義された形式で表示されます。

> **注意：** アプリケーションでは、一時 CLOB 内に含まれている情報を読み込んだ後、一時 CLOB を解放する必要があります。

パラメータ

ctx[ ]
フォーマット・プラグインのコンテキスト情報

戻り値

属性の値を、一時 oracle.sql.CLOB で戻します。

例外

java.sql.SQLException

データベース内で対応する getAllAttributes( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
CLOB attributes = vidObj.getAllAttributes(ctx);
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。

getAttribute( )

構文

```
public String getAttribute(byte[ ] [ ] ctx, String name)
```

説明

要求されたビデオ・プロパティの値を戻します。このメソッドは、ユーザー定義のフォーマット・プラグインで、OrdVideo Java オブジェクトの属性として使用できないビデオ・プロパティの値を戻すために使用します。このメソッドは、オラクル社が提供するフォーマット・プラグインでは実装されません。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

name
プロパティまたは属性の名前

戻り値

属性の値を、文字列で戻します。

例外

java.sql.SQLException

データベース内で対応する getAttribute( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
String attribute = vidObj.getAttribute(ctx, video_duration);
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。
- video_duration: 取得する属性の値です。

getBFILE( )

構文

```
public oracle.sql.BFILE getBFILE( )
```

説明

srcType 属性が file の場合、データベースから BFILE ロケータを戻します。このメソッドは、データベース内の対応する getBFILE( ) メソッド（srcLocation および srcName の属性を使用して BFILE を作成）をコールします。

パラメータ

なし

戻り値

oracle.sql.BFILE ロケータを戻します。

例外

java.sql.SQLException

データベース内で対応する getBFILE( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
BFILE videoBFILE = vidObj.getBFILE( );
```

getBitRate( )

構文

```
public int getBitRate( )
```

説明

bitRate 属性の値を戻します。

パラメータ

なし

戻り値

bitRate 属性の値を、整数で戻します。

例外

java.sql.SQLException

bitRate 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int bitRate = vidObj.getBitRate( );
```

getComments( )

構文

```
public oracle.sql.CLOB getComments( )
```

説明

コメント属性から CLOB ロケータを戻します。

パラメータ

なし

戻り値

コメント属性の値を、oracle.sql.CLOB ロケータで戻します。

例外

java.sql.SQLException

コメント属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
CLOB comments = vidObj.getComments( );
```

getCompressionType( )

構文

```
public String getCompressionType( )
```

説明

compressionType 属性の値を戻します。

パラメータ

なし

戻り値

compressionType 属性の値を、文字列で戻します。

例外

java.sql.SQLException

compressionType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String compressionType = vidObj.getCompressionType( );
```

getContent( )

構文

```
public oracle.sql.BLOB getContent( )
```

説明

localData 属性の BLOB ロケータを戻します。

パラメータ

なし

戻り値

oracle.sql.BLOB ロケータを戻します。

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
BLOB localContent = vidObj.getContent( );
```

getContentInLob( )

構文

```
public oracle.sql.BLOB getContentInLob(byte[ ] [ ] ctx,
  String mimeType[ ], String format[ ])
```

説明

データベース内の一時 BLOB の localData 属性で指定された BLOB からデータを戻します。このメソッドは、データベース内に一時 BLOB を作成し、localData 属性で指定された BLOB からデータを読み込み、一部 BLOB にデータを書き込み、一時 BLOB ロケータをコール元に戻します。

> **注意：** アプリケーションでは、一時 BLOB 内に含まれている情報にアクセスした後、一時 BLOB を解放する必要があります。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

mimeType
mimeType 属性が要素 0 として書き込まれる文字列配列（長さは 1 要素）

format
format 属性が要素 0 として書き込まれる文字列配列（長さは 1 要素）

戻り値

一時 oracle.sql.BLOB ロケータのビデオデータを戻します。

例外

java.sql.SQLException

一時 BLOB の作成中、またはデータベース内で対応する getContentInLob( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
String mimetype[ ] = new String[1];
String format[ ] = new String[1];
BLOB localContent = vidObj.getContentInLob(ctx,mimetype,format);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- mimeType: 最初の値に MIME タイプを含む文字列の配列です。この値は、サーバーによって生成されます。
- format: 最初の値にデータ・フォーマットを含む文字列の配列です。この値は、サーバーによって生成されます。

getContentLength( )

構文

```
public int getContentLength( )
```

説明

ビデオ・データ長を戻します。このメソッドは、データベース内の対応する getContentLength( ) メソッドをコールします。

このメソッドをサポートしないソース・タイプもあります。たとえば、http ソース・タイプではサポートされません。

パラメータ

なし

戻り値

contentLength 属性の値を、整数で戻します。

例外

java.sql.SQLException

データベース内で対応する getContentLength( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
int contentLength = vidObj.getContentLength( );
```

getContentLength(byte[ ][ ])

構文

```
public int getContentLength(byte[ ] [ ] ctx)
```

説明

ソース・プラグインのコンテキスト情報を使用して、ビデオ・データ長を戻します。このメソッドは、データベース内の対応する getContentLength( ) メソッドをコールします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

戻り値

contentLength 属性の値を、整数で戻します。

例外

java.sql.SQLException

データベース内で対応する getContentLength( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
int contentLength = vidObj.getContentLength(ctx);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。

getDataInByteArray( )

構文

```
public byte[ ] getDataInByteArray( )
```

説明

localData 属性で指定されたデータベース BLOB からデータを含むバイト配列を戻します。

パラメータ

なし

戻り値

データを含むバイト配列を戻します。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

BLOB からのデータ読込み中にエラーが発生した場合、この例外がスローされます。

java.lang.OutOfMemoryError

データの保持のために十分なメモリーを割当てできない場合、この例外がスローされます。

例

```
byte[ ] byteArr = vidObj.getDataInByteArray( );
```

getDataInFile( )

構文

```
public boolean getDataInFile(String filename)
```

説明

localData 属性で指定されたデータベース BLOB からローカル・ファイルに、データを書き込みます。

パラメータ

filename
データの書込み先のファイル名

戻り値

データが正常にファイルに書き込まれた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false は戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

BLOB からのデータ読込み中または出力ファイルへのデータ書込み中にエラーが発生した場合、この例外がスローされます。

例

```
boolean load = vidObj.getDataInFile("output1.dat");
if(load)
     System.out.println("getDataInFile completed successfully");
else
     System.out.println("Error in getDataInFile");
```

パラメータは次のとおりです。

- output1.dat: データのロード先のファイルです。

getDataInStream( )

構文

```
public InputStream getDataInStream( )
```

説明

localData 属性で指定されたデータベース BLOB のデータの読込み元となる InputStream オブジェクトを戻します。

パラメータ

なし

戻り値

データの読込み元から InputStream オブジェクトを戻します。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
InputStream inpStream = vidObj.getDataInStream( );
```

getDescription( )

構文

```
public String getDescription( )
```

説明

description 属性の値を戻します。

パラメータ

なし

戻り値

description 属性の値を、文字列で戻します。

例外

java.sql.SQLException

description 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String description = vidObj.getDescription( );
```

getFactory( )

構文

```
public static oracle.sql.CustomDatumFactory getFactory( )
```

説明

getCustomDatum( ) メソッドが使用する OrdAudio CustomDatumFactory インタフェースを戻します。OracleResultSet または OracleCallableStatement オブジェクトから OrdAudio オブジェクトを取り出す場合は、getFactory( ) メソッドを getCustomDatum( ) メソッドの factory パラメータとして指定します。

パラメータ

なし

戻り値

CustomDatumFactory インタフェースの OrdVideo 実装を戻します。

例外

なし

例

```
OrdVideo vid = (OrdVideo)rset.getCustomDatum( 1, OrdVideo.getFactory() );
```

getFormat( )

構文

```
public String getFormat( )
```

説明

format 属性の値を戻します。

パラメータ

なし

戻り値

format 属性の値を、文字列で戻します。

例外

java.sql.SQLException

format 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String format = vidObj.getFormat( );
```

getFrameRate( )

構文

```
public int getFrameRate( )
```

説明

frameRate 属性の値を戻します。

パラメータ

なし

戻り値

frameRate 属性の値を、整数で戻します。

例外

java.sql.SQLException

frameRate 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int frameRate = vidObj.getFrameRate( );
```

getFrameResolution( )

構文

```
public int getFrameResolution( )
```

説明

frameResolution 属性の値を戻します。

パラメータ

なし

戻り値

frameResolution 属性の値を、整数で戻します。

例外

java.sql.SQLException

frameResolution 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int frameResolution = vidObj.getFrameResolution( );
```

getHeight( )

構文

```
public int getHeight( )
```

説明

height 属性の値を戻します。

パラメータ

なし

戻り値

height 属性の値を、整数で戻します。

例外

java.sql.SQLException

height 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int height = vidObj.getHeight( );
```

getMimeType( )

構文

```
public String getMimeType( )
```

説明

mimeType 属性の値を戻します。

パラメータ

なし

戻り値

mimeType 属性の値を、文字列で戻します。

例外

java.sql.SQLException

mimeType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String mimeType = vidObj.getMimeType( );
```

getNumberOfColors( )

構文

```
public int getNumberOfColors( )
```

説明

numberOfColors 属性の値を戻します。

パラメータ

なし

戻り値

numberOfColors 属性の値を、整数で戻します。

例外

java.sql.SQLException

numberOfColors 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int numberOfColors = vidObj.getNumberOfColors( );
```

getNumberOfFrames( )

構文

```
public int getNumberOfFrames( )
```

説明

numberOfFrames 属性の値を戻します。

パラメータ

なし

戻り値

numberOfFrames 属性の値を、整数で戻します。

例外

java.sql.SQLException

numberOfFrames 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int numberOfFrames = vidObj.getNumberOfFrames( );
```

getSource( )

構文

```
getSource( )
```

説明

srcType://srcLocation/srcName という形式でソース情報を戻します。

パラメータ

なし

戻り値

ソース情報を、文字列で戻します。

例外

java.sql.SQLException

データベース内で対応する getSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
String source = viObj.getSource( );
```

getSourceLocation( )

構文

```
public String getSourceLocation( )
```

説明

srcLocation 属性の値を戻します。

パラメータ

なし

戻り値

srcLocation 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcLocation 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String location = vidObj.getSourceLocation( );
```

getSourceName( )

構文

```
public String getSourceName( )
```

説明

srcName 属性の値を戻します。

パラメータ

なし

戻り値

srcName 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcName 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String name = vidObj.getSourceName( );
```

getSourceType( )

構文

```
public String getSourceType( )
```

説明

srcType 属性の値を戻します。

パラメータ

なし

戻り値

srcType 属性の値を、文字列で戻します。

例外

java.sql.SQLException

srcType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
String type = vidObj.getSourceType( );
```

getUpdateTime( )

構文

```
public java.sql.Timestamp getUpdateTime( )
```

説明

updateTime 属性の値を戻します。

パラメータ

なし

戻り値

updateTime 属性の値を、java.sql.Timestamp オブジェクトで戻します。

例外

java.sql.SQLException

updateTime 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
Timestamp time = vidObj.getUpdateTime( );
```

getVideoDuration( )

構文

```
public int getVideoDuration( )
```

説明

videoDuration 属性の値を戻します。

パラメータ

なし

戻り値

videoDuration 属性の値を、整数で戻します。

例外

java.sql.SQLException

videoDuration 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int videoDuration = vidObj.getVideoDuration( );
```

getWidth( )

構文

```
public int getWidth( )
```

説明

width 属性の値を戻します。

パラメータ

なし

戻り値

width 属性の値を、整数で戻します。

例外

java.sql.SQLException

width 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
int width = vidObj.getWidth( );
```

importData( )

構文

```
public void importData(byte[ ] [ ] ctx)
```

説明

データを、外部ソースから localData 属性で指定されたデータベース BLOB にインポートします。外部データ・ソースは、srcType、srcLocation および srcName の属性で指定されます。

パラメータ

ctx
ソース・プラグインのコンテキスト情報

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する import( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
vidObj.importData(ctx);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。

importFrom( )

構文

```
public void importFrom(byte[ ] [ ] ctx,
  String srcType, String srcLocation, String srcName)
```

説明

データを、外部ソースから localData 属性で指定されたデータベース BLOB にインポートします。外部データ・ソースは、srcType、srcLocation および srcName のパラメータで指定されます。srcType、srcLocation および srcName 属性は、importFrom( ) メソッドに渡された srcType、srcLocation および srcName パラメータの値により更新されます。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

srcType
データのインポート元のソース・タイプ

srcLocation
データのインポート元のソースの位置

srcName
データのインポート元のソース名

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する importFrom( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
vidObj.importFrom(ctx,"file","VIDEODIR","testvid.dat");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- file: データのインポートに使用するソース・プラグインです。
- VIDEODIR: データのインポート元のファイルの位置です。
- testvid.dat: データのインポート元のファイルです。

isLocal( )

構文

```
public boolean isLocal( )
```

説明

ビデオ・データが、localData 属性によって指定されたデータベース内の BLOB でローカル格納されているかどうかを示します。

パラメータ

なし

戻り値

データベース内の BLOB でデータがローカルに格納されている場合は true、格納されていない場合は false を戻します。

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
if(vidObj.isLocal( ))
     System.out.println("local attribute is set to true");
else
     System.out.println("local attribute is set to false");
```

loadDataFromByteArray( )

構文

```
public boolean loadDataFromByteArray(byte[ ] byteArr)
```

説明

データを、バイト配列から localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

byteArr
データのロード元のバイト配列

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false は戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中またはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

バイト配列の読込み中にエラーが発生した場合、この例外がスローされます。

例

```
byte[ ] data = new byte[32000];
FileInputStream fStream = new FileInputStream("testvid.dat");
fStream.read(data,0,32000);
boolean success = vidObj.loadDataFromByteArray(data);
if(success)
     System.out.println("loadDataFromByteArray was successful");
else
     System.out.println("loadDataFromByteArray was unsuccessful");
```

パラメータは次のとおりです。

- data: データのロード元のローカル・バイト配列です。
- testvid.dat: 32,000 バイトのデータを含むローカル・ファイルです。

loadDataFromFile( )

構文

```
public boolean loadDataFromFile(String filename)
```

説明

データを、ファイルから localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

filename
データのロード元のファイル名

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false は戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中またはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

データ・ファイルの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.loadDataFromFile("testvid.dat");
```

パラメータは次のとおりです。

- testvid.dat: ビデオ・データを含むローカル・ファイルです。

loadDataFromInputStream( )

構文

```
public boolean loadDataFromInputStream(InputStream inpStream)
```

説明

データを、InputStream オブジェクトから localData 属性で指定されたデータベース BLOB にロードします。このメソッドは、データをロードする前に、deleteContent( ) メソッドをコールして BLOB 内の既存のデータを削除します。また、setLocal( ) メソッドをコールして、ローカル・フラグを設定し、setUpdateTime( ) メソッドをコールして、updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定もします。

パラメータ

inpStream
データのロード元の InputStream オブジェクト

戻り値

データが正常にロードされた場合は、true を戻します。エラーが発生した場合は、例外が生成されます。このメソッドは false は戻しません。

例外

java.sql.SQLException

オブジェクト属性へのアクセス中またはデータベース内でのメソッドの実行中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

InputStream オブジェクトの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
FileInputStream fStream = new FileInputStream("testvid.dat");
vidObj.loadDataFromInputStream(fStream);
```

パラメータは次のとおりです。

- testvid.dat: ビデオ・データを含むローカル・ファイルです。
- fStream: ビデオ・データを OrdVideo オブジェクトにロードするローカル InputStream オブジェクトです。

openSource( )

構文

```
public int openSource(byte[ ] userArg, byte[ ] [ ] ctx)
```

説明

データ・ソースをオープンします。

パラメータ

userArg
ユーザー定義のソース・プラグインで場合に応じて必要となる追加ソース・プラグイン情報

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

戻り値

ステータスを、整数で戻します。0（ゼロ）は成功を示し、0（ゼロ）以外の値はソース・プラグイン固有の障害コードを示します。

例外

java.lang.Exception

データベース内で対応する openSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte[ ] userarg = new byte[4000];
byte [ ] [ ] ctx = new byte[1][64];
int i = vidObj.openSource(userarg,ctx);
if(i == 0)
     System.out.println("openSource successful");
else
     System.out.println("openSource unsuccessful");
```

パラメータは次のとおりです。

- userArg: 権限関連のパラメータです。
- ctx: ソース・プラグインのコンテキスト情報です。

processSourceCommand( )

構文

```
public byte[ ] processSourceCommand(byte[ ] [ ] ctx,
  String cmd, String args, byte[ ] [ ] result)
```

説明

データベース内でソース・プラグインをコールして、コマンドを実行します。このメソッドは、ユーザー定義のプラグインのみで使用します。このメソッドをオラクル社が提供するソース・プラグインで使用すると、例外が生成されます。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

cmd
ソース・プラグインによって実行されるコマンド

args
コマンド引数

result
[1][*n*] 形式のバイト配列。ここにコマンドの実行結果が書き込まれます。

戻り値

コマンドの実行結果を戻します。

例外

java.sql.SQLException

データベース内で対応する processSourceCommand( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
String cmd;
String args;
byte [ ] [ ] result;
//assign a command value to cmd
//assign any arguments to args
byte[ ] commandResults = vidObj.processSourceCommand(ctx,cmd,
     args,result);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインの情報です。
- cmd: 実行されるコマンドです。
- args: コマンドに必要なすべての引数です。
- result: コマンドの実行結果です。

processVideoCommand( )

構文

```
public byte[ ] processVideoCommand(byte[ ] [ ] ctx,
  String cmd, String args, byte[ ] [ ] result)
```

説明

データベース内でフォーマット・プラグインをコールして、コマンドを実行します。このメソッドは、ユーザー定義のフォーマット・プラグインのみで使用します。このメソッドをオラクル社が提供するフォーマット・プラグインで使用すると、例外が生成されます。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

cmd
フォーマット・プラグインによって実行されるコマンド

args
コマンド引数

result
[1][n] 形式のバイト配列。ここにコマンドの実行結果が書き込まれます。

戻り値

コマンドの実行結果を戻します。

例外

java.sql.SQLException

データベース内で対応する processVideoCommand( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64]
String cmd;
String args;
byte [ ] [ ] result;
//assign a command value to cmd
//assign any arguments to args
byte[ ] commandResults = vidObj.processVideoCommand(ctx,cmd,
     args,result);
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインの情報です。
- cmd: 実行されるコマンドです。
- args: コマンドに必要なすべての引数です。
- result: コマンドの実行結果です。

readFromSource( )

構文

```
public int readFromSource(byte[ ] [ ] ctx,
  int startPos, int numBytes, byte[ ] [ ] buffer)
```

説明

データ・ソースからデータを読み込みます。このメソッドは、指定されたバイト数を、データ・ソースの指定された位置から開始して、データ・ソースからアプリケーション・バッファに読み込みます。

データ・ソースを読込み前にオープンする必要がないソース・プラグインもあります。ただし、アプリケーションが現行または今後のソース・プラグインで必ず動作するようにするには、このメソッドをコールする前に、openSource( ) メソッドをコールします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

startPos
データ・ソース内の開始位置

numBytes
データ・ソースから読み込まれるバイト数

buffer
[1][n] 形式のバイト配列。n は、numBytes 以上の値です。

戻り値

読み込んだバイト数を、整数で戻します。

例外

java.sql.SQLException

データベース内で対応する readFromSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
byte [ ] [ ] buffer = new byte[12];
int i = vidObj.readFromSource(ctx,0,12,buffer);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- 0: localData フィールドからの読込み開始位置です。
- 12: 読込みバイト数です。
- buffer: データの読込み先の位置です。

setBitRate( )

構文

```
public void setBitRate(int bitRate)
```

説明

bitRate 属性の値を設定します。

bitRate 属性は、setProperties( ) メソッドによって自動的に特定のビデオ・フォーマットに設定されます。setBitRate( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

bitRate
新しい属性値

戻り値

なし

例外

java.sql.SQLException

bitRate 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setBitRate(1500);
```

パラメータは次のとおりです。

- 1500: bitRate 属性に設定する値（1 秒当たりのビット数）

setComments( )

構文

```
public void setComments(oracle.sql.CLOB comments)
```

説明

コメント属性の値を設定します。

*inter*Media によって、使用するコメント属性が予約されます。任意の値を設定できますが、Oracle *inter*Media Annotator または setProperties( ) メソッドによって上書きされる場合があります。

パラメータ

comments
新しい属性値

戻り値

なし

例外

java.sql.SQLException

コメント属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setComments(commentsData);
```

パラメータは次のとおりです。

- commentsData: コメント属性に設定するデータを含む CLOB です。

setCompressionType( )

構文

```
public void setCompressionType(String compressionType)
```

説明

compressionType 属性の値を設定します。

compressionType 属性は、setProperties( ) メソッドによって自動的に特定のビデオ・フォーマットに設定されます。setCompressionType( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

compressionType
新しい属性値

戻り値

なし

例外

java.sql.SQLException

compressionType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setCompressionType("Cinepak");
```

パラメータは次のとおりです。

- Cinepak: compressionType 属性に設定する値です。

setDescription( )

構文

```
public void setDescription(String description)
```

説明

description 属性の値を設定します。

パラメータ

description
新しい属性値

戻り値

なし

例外

java.sql.SQLException

description 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setDescription("My video file");
```

パラメータは次のとおりです。

- My video file: description 属性に設定する値です。

setFormat( )

構文

```
public void setFormat(String format)
```

説明

format 属性の値を設定します。

format 属性は、ビデオ・データを操作するメソッドへのコールの処理に使用するフォーマット・プラグインを判断します。特に、setProperties( ) メソッドは、format 属性を使用して、ビデオ・データのプロパティを解析するためにコールするフォーマット・プラグインを判断します。setProperties( ) メソッドをコールする前の、format 属性の初期化の方法、および setProperties( ) メソッドをオラクル社が提供するデフォルトのプラグインで使用した format 属性の値の設定方法については、setProperties( ) メソッドを参照してください。setFormat( ) メソッドのコールで設定されるのは属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

format
新しい属性値

戻り値

なし

例外

java.sql.SQLException

format 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setFormat("MOOV");
```

パラメータは次のとおりです。

- MOOV: format 属性に設定する値です。

setFrameRate( )

構文

```
public void setFrameRate(int frameRate)
```

説明

frameRate 属性の値を設定します。

frameRate 属性は、setProperties( ) メソッドによって自動的に特定のビデオ・フォーマットに設定されます。setFrameRate( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

frameRate
新しい属性値

戻り値

なし

例外

java.sql.SQLException

frameRate 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setFrameRate(5);
```

パラメータは次のとおりです。

- 5: frameRate 属性に設定する値（1 秒当たりのフレーム数）

setFrameResolution( )

構文

```
public void setFrameResolution(int frameResolution)
```

説明

frameResolution 属性の値を設定します。

frameResolution 属性は、setProperties( ) メソッドによって自動的に特定のビデオ・フォーマットに設定されます。setFrameResolution( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

frameResolution
新しい属性値

戻り値

なし

例外

java.sql.SQLException

frameResolution 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setFrameResolution(4);
```

パラメータは次のとおりです。

- 4: frameResolution 属性に設定する値です。

setHeight( )

構文

```
public void setHeight(int height)
```

説明

height 属性の値を設定します。

height 属性は、setProperties( ) メソッドによって自動的に特定のビデオ・フォーマットに設定されます。setHeight( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

height
新しい属性値

戻り値

なし

例外

java.sql.SQLException

height 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setHeight(24);
```

パラメータは次のとおりです。

- 24: height 属性に設定する値（ピクセル単位）です。

setKnownAttributes( )

構文

```
public void setKnownAttributes(String format, int width, int height,
                               int frameResolution, int frameRate,
                               int videoDuration, int numberOfFrames,
                               String compressionType, int numberOfColors,
                               int bitRate)
```

説明

OrdVideo Java オブジェクトの既知の属性値を設定します。

属性（format、width、height、frameResolution、frameRate、videoDuration、numberOfFrames、compressionType、numberOfColors および bitRate）は、setProperties( ) メソッドによって自動的に特定のビデオ・フォーマットに設定されます。setKnownAttributes( ) メソッドは、setProperties( ) を使用していない場合にのみ使用します。このメソッドで設定されるのは指定された属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

format
新しい属性値（文字列）

width
新しい属性値（整数）

height
新しい属性値（整数）

frameResolution
新しい属性値（整数）

frameRate
新しい属性値（整数）

videoDuration
新しい属性値（整数）

numberOfFrames
新しい属性値（整数）

compressionType
新しい属性値（文字列）

numberOfColors
新しい属性値（整数）

bitRate
新しい属性値（整数）

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setKnownAttributes( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setKnownAttributes("MOOV",1,2,4,5,20,8,"Cinepak",256,1500);
```

パラメータは次のとおりです。

- MOOV: format 属性に設定する値です。
- 1: width 属性に設定する値（ピクセル単位）です。
- 2: height 属性に設定する値（ピクセル単位）です。
- 4: frameResolution 属性に設定する値です。
- 5: frameRate 属性に設定する値（1 秒当たりのフレーム数）です。
- 20: videoDuration 属性に設定する値です。
- 8: numberOf Frames 属性に設定する値です。
- Cinepak: compressionType 属性に設定する値です。
- 256: numberOfColors 属性に設定する値です。
- 1500: bitRate 属性に設定する値（1 秒当たりのビット数）です。

setLocal( )

構文

```
public void setLocal( )
```

説明

ビデオ・データが、localData 属性で指定されたデータベース内の BLOB でローカルに格納されているかどうかを示すために、ローカル属性の値を設定します。

パラメータ

なし

戻り値

なし

例外

java.sql.SQLException

localData 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setLocal( );
```

setMimeType( )

構文

```
public void setMimeType(String mimeType)
```

説明

mimeType 属性の値を設定します。

mimeType 属性は、setProperties( ) メソッドによって自動的に特定のビデオ・フォーマットに設定されます。setMimeType( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

mimeType
新しい属性値

戻り値

なし

例外

java.sql.SQLException

mimeType 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setMimeType("video/avi");
```

パラメータは次のとおりです。

- video/avi: 設定する MIME タイプです。

setNumberOfColors( )

構文

```
public void setNumberOfColors(int numberOfColors)
```

説明

numberOfColors 属性の値を設定します。

numberOfColors 属性は、setProperties( ) メソッドによって自動的に特定のビデオ・フォーマットに設定されます。setNumberOfColors( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

numberOfColors
新しい属性値

戻り値

なし

例外

java.sql.SQLException

numberOfColors 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setNumberOfColors(256);
```

パラメータは次のとおりです。

- 256: numberOfColors 属性に設定する値です。

setNumberOfFrames( )

構文

```
public void setNumberOfFrames(int numberOfFrames)
```

説明

numberOfFrames 属性の値を設定します。

numberOfFrames 属性は、setProperties( ) メソッドによって自動的に特定のビデオ・フォーマットに設定されます。setNumberOfFrames( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

numberOfFrames
新しい属性値

戻り値

なし

例外

java.sql.SQLException

numberOfFrames 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setNumberOfFrames(8);
```

パラメータは次のとおりです。

- 8: numberOfFrames 属性に設定する値です。

setProperties(byte[ ][ ])

構文

```
public void setProperties(byte[ ] [ ] ctx)
```

説明

ビデオ・データのプロパティを解析し、OrdVideo Java オブジェクト内の属性の値を設定します。このメソッドは、format、mimeType、width、height、frameResolution、frameRate、videoDuration、numberOfFrames、compressionType、numberOfColors および bitRate 属性の値を設定します。対応するプロパティが特定のビデオ・フォーマットについて抽出できない場合、属性は null に設定されます。ビデオ・フォーマットが認識できない場合は、SQLException エラーがスローされます。

format 属性は、ビデオ・データのプロパティの解析に使用するフォーマット・プラグインを判断します。setProperties( ) メソッドへのコール時に format 属性が null の場合は、ビデオ・データのプロパティの解析、およびサポートされているビデオ・フォーマットに対応する様々な属性（実際のビデオ・データ・フォーマットなど）の埋込みに、オラクル社が提供するデフォルトのフォーマット・プラグインが使用されます。オラクル社が提供するフォーマット・プラグインによってサポートされるビデオ・フォーマットの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。データベース内の ORDVideo.init メソッドは、常に、format 属性の値を null に設定することに注意してください。format 属性が null でない場合は、setProperties( ) メソッドへのコール時に、format 属性で指定されたフォーマット・プラグインがコールされます。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
vidObj.setProperties(ctx);
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。

setProperties(byte[ ][ ], boolean)

構文

```
public void setProperties(byte[ ] [ ] ctx, boolean setComments)
```

説明

ビデオ・データのプロパティを解析し、OrdVideo Java オブジェクト内の属性の値を設定し、オプションでコメント属性で指定された CLOB を移入します。このメソッドは、format、mimeType、width、height、frameResolution、frameRate、videoDuration、numberOfFrames、compressionType、numberOfColors および bitRate 属性の値を設定します。対応するプロパティが特定のビデオ・フォーマットについて抽出できない場合、属性は null に設定されます。setComments パラメータが true の場合、このメソッドは抽出された XML 形式のすべてのプロパティを、コメント属性で指定された CLOB に移入します。setComments パラメータが false の場合、コメント属性は変更されません。ビデオ・フォーマットが認識できない場合は、SQLException エラーがスローされます。

format 属性は、ビデオ・データのプロパティの解析に使用するフォーマット・プラグインを判断します。setProperties( ) メソッドへのコール時に、format 属性が null の場合は、ビデオ・データのプロパティの解析、およびサポートされているビデオ・フォーマットに対応する様々な属性（実際のビデオ・データ・フォーマットなど）の埋込みに、オラクル社が提供するデフォルトのフォーマット・プラグインが使用されます。オラクル社が提供するフォーマット・プラグインによってサポートされるビデオ・フォーマットの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。データベース内の ORDVideo.init メソッドは、常に、format 属性の値を null に設定することに注意してください。format 属性が null でない場合は、setProperties( ) メソッドへのコール時に、format 属性で指定されたフォーマット・プラグインがコールされます。

パラメータ

ctx
フォーマット・プラグインのコンテキスト情報

setComments
コメント属性によって指定された CLOB を移入するかどうかを指定するブール値

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setProperties( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
vidObj.setProperties(ctx,true);
```

パラメータは次のとおりです。

- ctx: フォーマット・プラグインのコンテキスト情報です。
- true: コメント・フィールドが移入されることを示します。

setSource( )

構文

```
public void setSource(String srcType, String srcLocation,
  String srcName)
```

説明

srcType、srcLocation および srcName の属性値を設定します。

パラメータ

srcType
ソース・タイプ

srcLocation
ソースの位置

srcName
ソースの名前

戻り値

なし

例外

java.sql.SQLException

srcType、srcLocation または srcName 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setSource("LOCAL","VIDEODIR","video.dat");
```

パラメータは次のとおりです。

- LOCAL: ソース・タイプです。
- VIDEODIR: ソースの位置です。
- video.dat: ソースの名前です。

setUpdateTime( )

構文

```
public void setUpdateTime(java.sql.Timestamp currentTime)
```

説明

updateTime 属性の値を設定します。このメソッドは、updateTime 属性の値を指定された時刻に設定します。ただし、currentTime 属性が null に指定されている場合は、updateTime 属性の値をデータベース・サーバーの現在の SYSDATE 時刻に設定します。

パラメータ

currentTime
updateTime 属性をデータベース・サーバーの現在の SYSDATE 時刻に設定するために使用する更新時刻（null 値）

戻り値

なし

例外

java.sql.SQLException

データベース内で対応する setUpdateTime( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setUpdateTime(null);
```

setVideoDuration( )

構文

```
public void setVideoDuration(int videoDuration)
```

説明

videoDuration 属性の値を設定します。

videoDuration 属性は、setProperties( ) メソッドによって自動的に特定のビデオ・フォーマットに設定されます。setVideoDuration( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

videoDuration
新しい属性値

戻り値

なし

例外

java.sql.SQLException

videoDuration 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setVideoDuration(20);
```

パラメータは次のとおりです。

- 20: videoDuration 属性に設定する値です。

setWidth( )

構文

```
public void setWidth(int width)
```

説明

width 属性の値を設定します。

width 属性は、setProperties( ) メソッドによって自動的に特定のビデオ・フォーマットに設定されます。setWidth( ) メソッドは、setProperties( ) メソッドを使用していない場合にのみ使用します。このメソッドで設定されるのは属性値のみで、ビデオ・データ自体は変更されません。

パラメータ

width
新しい属性値

戻り値

なし

例外

java.sql.SQLException

width 属性へのアクセス中にエラーが発生した場合、この例外がスローされます。

例

```
vidObj.setWidth(24);
```

パラメータは次のとおりです。

- 24: width 属性に設定する値（ピクセル単位）です。

trimSource( )

構文

```
public int trimSource(byte[ ] [ ] ctx, int newLen)
```

説明

データを指定の長さに切り捨てます。

ただし、切捨て操作をサポートしないソース・プラグインもあります。たとえば、アプリケーションは localData 属性で指定された BLOB に格納されているデータの切捨てを行うことができますが、file および http のデータ・ソース・タイプは書込みアクセスをサポートしないため、このメソッドもサポートしません。また、書込みアクセスをサポートするソース・プラグインが切捨て操作をサポートしない場合もあります。

データ・ソースを変更前にオープンする必要がないソース・プラグインもあります。ただし、アプリケーションが現行または今後のソース・プラグインで必ず動作するようにするには、このメソッドをコールする前に openSource( ) メソッドをコールします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

newLen
切捨て後のデータの長さ

戻り値

ステータスを、整数で戻します。0（ゼロ）は成功を示し、0（ゼロ）以外の値はソース・プラグイン固有の障害コードを示します。

例外

java.sql.SQLException

データベース内で対応する trimSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx = new byte[1][64];
int i = vidObj.trimSource(ctx,10);
if (i == 0)
     System.out.println("trimSource successful");
else
     System.out.println("trimSource unsuccessful");
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- 10: ソースの新しい長さです。

writeToSource( )

構文

```
public int writeToSource(byte[ ] [ ] ctx,
  int startPos, int numBytes, byte[ ] buffer)
```

説明

データをデータ・ソースに書き込みます。このメソッドは、指定されたバイト数を、データ・ソースの指定された位置から開始してアプリケーションのバッファからデータ・ソースに書き込みます。

ただし、書込み操作をサポートしないソース・プラグインもあります。たとえば、アプリケーションは localData 属性で指定された BLOB への書込みを行うことができますが、file および http のデータ・ソース・タイプは書込みアクセスをサポートしないため、このメソッドもサポートしません。書込みアクセスをサポートするソース・プラグインが、順次書込みアクセスのみサポートし、データ・ソース内の任意の位置への書込みアクセスはサポートしない場合もあります。

データ・ソースを書込み前にオープンする必要がないソース・プラグインもあります。ただし、アプリケーションが現行または今後のソース・プラグインで必ず動作するようにするには、このメソッドをコールする前に openSource( ) メソッドをコールします。

パラメータ

ctx
ソース・プラグインのコンテキスト情報。詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

startPos
データ・ソース内の開始位置

numBytes
データ・ソースに書き込まれるバイト数

buffer
書き込まれるデータを含むバイト配列

戻り値

書き込まれるバイト数を、整数で戻します。

例外

java.sql.SQLException

データベースの対応する writeToSource( ) メソッドの実行中にエラーが発生した場合、この例外がスローされます。

例

```
byte [ ] [ ] ctx= new byte[1][64];
byte[ ] data = new byte[20];
//populate data with 20 bytes of content
int i = vidObj.writeToSource(ctx,1,20,data);
```

パラメータは次のとおりです。

- ctx: ソース・プラグインのコンテキスト情報です。
- 1: 書込みが開始されるコメント・フィールドの位置です。
- 20: 書込みバイト数です。
- data: 書き込むコンテンツです。

9

Java Classes for Servlets and JSP のリファレンス情報

Oracle *inter*Media Java Classes for Servlets and JavaServer Pages（JSP）を使用すると、Oracle データベースに対するメディア・データの検索およびアップロードが簡単になります。

OrdHttpResponseHandler クラスを使用すると、Oracle データベースからのメディア・データの検索、および Java サーブレットからブラウザまたはその他の HTTP クライアントへのメディア・データの配信が簡単になります。OrdHttpJspResponseHandler クラスは、JSP 用の同じ機能を提供します。

HTML 形式を使用したフォームベースのファイルのアップロードでは、multipart/form-data フォーマットを使用して、POST 要求のフォーム・データおよびアップロード・ファイルをエンコードします。OrdHttpUploadFormData クラスを使用すると、POST データを解析し、通常の形式のフィールドまたはアップロード・ファイルのコンテンツを Java サーブレットまたは JSP にアクセス可能にすることによって、このような要求の処理が簡単になります。OrdHttpUploadFile クラスを使用すると、操作が簡単な API で、アプリケーションからコールしてイメージ、オーディオおよびビデオのデータをデータベースにロードできるため、アップロード・ファイルの処理が簡単になります。

9.1 前提条件

*inter*Media メソッドを実行するには、次のインポート文を Java ファイルに組み込む必要があります。

```
import java.sql.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
import oracle.ord.im.*;
```

OrdHttpResponseHandler リファレンス情報

ここでは、OrdHttpResponseHandler クラスのメソッドに関するリファレンス情報を説明します。

OrdHttpResponseHandler クラスを使用すると、Oracle データベースからのメディア・データの検索、および Java サーブレットからブラウザまたはその他の HTTP クライアントへのメディア・データの配信が簡単になります。

OrdImage オブジェクトなどの *inter*Media Java オブジェクトは、JDBC 文またはオブジェクト取得元の結果セットには依存しません。ただし、オブジェクトで使用され、SQL 文が実行されたか、結果セットの取得元となった JDBC 接続には依存します。このため、データベースから *inter*Media Java オブジェクトを取得すると、メディア・データをブラウザに配信するまで、アプリケーションでは JDBC 接続を解放できません。

このクラスは、次のフィールドを含みます。

- public static final int DEFAULT_BUFFER_SIZE

  OrdHttpResponseHandler クラスでは、データベースから LOB データを検索し、クライアントに配信するために、サイズ 32768 のデフォルト・バッファが使用されます。この値は、setBufferSize( ) メソッドで上書きできます。

次の例では、OrdHttpResponseHandler クラスを使用して、データベースからイメージを検索し、ブラウザに配信する方法を示します。

```
PreparedStatement stmt = conn.prepareStatement("select photo from photo_album
     where id = ?");
stmt.setString(1, request.getParameter("photo_id"));
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (rset.next( )){
     OrdImage media = (OrdImage)rset.getCustomDatum(1, OrdImage.getFactory( ));
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler(request,
          response);
     handler.sendImage(media);
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
rset.close( );
stmt.close( );
```

ISO 8859-1（Latin-1）以外のキャラクタ・セットを使用する場合の注意事項

OrdDoc オブジェクトから、ISO 8859-1（Latin-1）以外のキャラクタ・セット（charset）が使用されたテキストベースのドキュメントを検索し、そのドキュメントをブラウザに配信するには、アプリケーションで、HTTP Content-Type ヘッダーにキャラクタ・セット名を指定する必要があります。

OrdDoc オブジェクトの MIME タイプ属性にキャラクタ・セット指定が含まれている場合は、アプリケーションで sendDoc( ) メソッドをコールするのみで、そのドキュメントを検索してブラウザに配信できます。たとえば、日本語で記述された HTML ページが、MIME タイプ `text/html; charset=Shift_JIS` で、OrdDoc オブジェクトに格納されているとします。この場合、sendDoc( ) メソッドをコールすることによって、適切な Content-Type ヘッダーが送信され、これによって、ブラウザでそのページを正しく表示することができます。

ただし、OrdDoc オブジェクトの MIME タイプ属性にキャラクタ・セット指定が含まれていない場合は、sendResponse( ) メソッドの 1 つをコールして、MIME タイプを明示的に指定する必要があります。たとえば、日本語で記述された HTML ページが、MIME タイプ `text/html` で OrdDoc オブジェクトに格納され、キャラクタ・セット名が別の列に指定されている場合、アプリケーションで、sendResponse( ) メソッドをコールする前に、MIME タイプにキャラクタ・セット指定を追加する必要があります。次に例を示します。

```
OraclePreparedStatement stmt = (OraclePreparedStatement)conn.prepareStatement(
     "select doc, charset from documents where id = ?");
stmt.setString(1, request.getParameter("id");
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (rset.next( )){
     OrdDoc doc = (OrdDoc)rset.getCustomDatum(1, OrdDoc.getFactory( ));
     String charset = rset.getString(2);
     String mimeType = doc.getMimeType( ) + "; charset=" + charset;
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler(request,
          response);
     handler.sendResponse(mimeType, doc.getContentLength( ), doc.getContent( ),
          doc.getUpdateTime( ));
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
rset.close( );
stmt.close( );
```

OrdHttpResponseHandler( )

構文

```
public OrdHttpResponseHandler( )
```

説明

マルチメディア検索要求への応答を処理する OrdHttpResponseHandler オブジェクトを作成します。次に、アプリケーションで、setServletRequest( ) メソッドをコールして HttpServletResponse オブジェクトを指定する必要があります。また、オプションで setServletRequest( ) メソッドをコールして HttpServletRequest オブジェクトも指定できます。

パラメータ

なし

戻り値

なし

例外

なし

例

このメソッドの例は、「setServletRequest( )」を参照してください。

OrdHttpResponseHandler(HttpServletRequest,HttpServletResponse)

構文

```
public OrdHttpResponseHandler(javax.servlet.http.HttpServletRequest
                              request,javax.servlet.http.
                              HttpServletResponse response)
```

説明

マルチメディア検索要求への応答を処理する OrdHttpResponseHandler オブジェクトを作成し、応答ハンドラに対して HttpServletRequest オブジェクトおよび HttpServletResponse オブジェクトを指定します。

パラメータ

request
HttpServletRequest 型のオブジェクト

response
HttpServletResponse 型のオブジェクト

戻り値

なし

例外

なし

例

このメソッドの例は、「sendAudio( )」を参照してください。

sendAudio( )

構文

```
public void sendAudio(oracle.ord.im.OrdAudio media)
```

説明

OrdAudio オブジェクトからオーディオ・クリップを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。

パラメータ

media
oracle.ord.im.OrdAudio 型のオブジェクト

戻り値

なし

例外

java.lang.IllegalStateException

HttpServletRequest または HttpServletResponse が指定されていない場合、この例外がスローされます。

OrdHttpResponseException

ソース・タイプが認識されない場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("select media from sounds where id = ?");
stmt.setString(1, request.getParameter("id"));
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (rset.next( )){
     OrdAudio media = (OrdAudio)rset.getCustomDatum(1, OrdAudio.getFactory( ));
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler
          (request, response);
     handler.sendAudio(media);
     return;
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
```

パラメータは次のとおりです。

- media: oracle.ord.im.OrdAudio 型のオブジェクトです。

sendDoc( )

構文

```
public void sendDoc(oracle.ord.im.OrdDoc media)
```

説明

OrdDoc オブジェクトからメディア・データを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。

パラメータ

media
oracle.ord.im.OrdDoc 型のオブジェクト

戻り値

なし

例外

java.lang.IllegalStateException

HttpServletRequest または HttpServletResponse が指定されていない場合、この例外がスローされます。

OrdHttpResponseException

ソース・タイプが認識されない場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("select media from documents where id = ?");
stmt.setString(1, request.getParameter("id"));
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (rset.next( )){
     OrdDoc media = (OrdDoc)rset.getCustomDatum(1, OrdDoc.getFactory( ));
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler
          (request, response);
     handler.sendDoc(media);
     return;
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
```

パラメータは次のとおりです。

- media: oracle.ord.im.OrdDoc 型のオブジェクトです。

sendImage( )

構文

```
public void sendImage(oracle.ord.im.OrdImage media)
```

説明

OrdImage オブジェクトからイメージを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。

パラメータ

media
oracle.ord.im.OrdAudio 型のオブジェクト

戻り値

なし

例外

java.lang.IllegalStateException

HttpServletRequest または HttpServletResponse が指定されていない場合、この例外がスローされます。

OrdHttpResponseException

ソース・タイプが認識されない場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("select media from photos where id = ?");
stmt.setString(1, request.getParameter("id"));
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (rset.next( )){
     OrdImage media = (OrdImage)rset.getCustomDatum(1, OrdImage.getFactory( ));
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler
          (request, response);
     handler.sendImage(media);
     return;
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
```

パラメータは次のとおりです。

- media: oracle.ord.im.OrdImage 型のオブジェクトです。

sendResponse(String,int,BFILE,Timestamp)

構文

```
public void sendResponse(String contentType, int length,
                         oracle.sql.BFILE bfile, java.sql.Timestamp
                         lastModified)
```

説明

HTTP Response ヘッダーを作成して、データベースから BFILE のコンテンツを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。

パラメータ

contentType
コンテンツの MIME タイプを指定する文字列

length
データ長を指定する整数

bfile
メディア・データの検索元の oracle.sql.BFILE

lastModified
データが最後に更新された日付および時刻を指定する java.sql.Timestamp オブジェクト（更新日付および時刻が使用できない場合は null）

戻り値

なし

例外

java.lang.IllegalStateException

HttpServletRequest または HttpServletResponse が指定されていない場合、この例外がスローされます。

java.lang.IllegalArgumentException

長さが負の値の場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
OraclePreparedStatement stmt = (OraclePreparedStatement)conn.prepareStatement
     ("select mimetype, len, doc, updatetime from docfiles where id = ?");
stmt.setString( 1, request.getParameter("id"));
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (rset.next( )){
     String mimeType = rset.getString(1);
     int len = rset.getInt(2);
     BFILE bfile = rset.getBFILE(3);
     Timestamp updateTime = rset.getTimestamp(4);
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler
          (request, response);
     handler.sendResponse(mimeType, len, bfile, updateTime);
     return;
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
```

sendResponse(String,int,BLOB,Timestamp)

構文

```
public void sendResponse(String contentType, int length,
                         oracle.sql.BLOB blob, java.sql.Timestamp
                         lastModified)
```

説明

HTTP Response ヘッダーを作成して、データベースから BLOB のコンテンツを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。

パラメータ

contentType
コンテンツの MIME タイプを指定する文字列

length
データ長を指定する整数

blob
メディア・データの検索元の oracle.sql.BLOB

lastModified
データが最後に更新された日付および時刻を指定する java.sql.Timestamp オブジェクト（更新日付および時刻が使用できない場合は null）

戻り値

なし

例外

java.lang.IllegalStateException

HttpServletRequest または HttpServletResponse が指定されていない場合、この例外がスローされます。

java.lang.IllegalArgumentException

長さが負の値の場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
OraclePreparedStatement stmt = (OraclePreparedStatement)conn.prepareStatement
     ("select mimetype, len, doc, updatetime from docblobs where id = ?");
stmt.setString(1, request.getParameter("id"));
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (rset.next( )){
     String mimeType = rset.getString(1);
     int len = rset.getInt(2);
     BLOB blob = rset.getBLOB(3);
     Timestamp updateTime = rset.getTimestamp(4);
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler
          (request, response);
     handler.sendResponse(mimeType, len, blob, updateTime);
     return;
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
```

sendResponse(String,int,InputStream,Timestamp)

構文

```
public void sendResponse(String contentType, int length,
                         java.io.InputStream in, java.sql.Timestamp
                         lastModified)
```

説明

HTTP Response ヘッダーを作成して、InputStream オブジェクトのコンテンツを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。

パラメータ

contentType
コンテンツの MIME タイプを指定する文字列

length
データ長を指定する整数

in
メディア・データの検索元の InputStream オブジェクト

lastModified
データが最後に更新された日付および時刻を指定する java.sql.Timestamp オブジェクト（更新日付および時刻が使用できない場合は null）

戻り値

なし

例外

java.lang.IllegalStateException

HttpServletRequest または HttpServletResponse が指定されていない場合、この例外がスローされます。

java.lang.IllegalArgumentException

長さが負の値の場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
OraclePreparedStatement stmt = (OraclePreparedStatement)conn.prepareStatement
     ("select mimetype, len, doc, updatetime from docblobs where id = ?");
stmt.setString(1, request.getParameter("id"));
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if ( rset.next( ) ){
     String mimeType = rset.getString(1);
     int len = rset.getInt(2);
     BLOB blob = rset.getBLOB(3);
     Timestamp updateTime = rset.getTimestamp(4);
     InputStream blobInputStream = blob.getBinaryStream( );
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler
          (request, response);
     handler.sendResponse(mimeType, len, blobInputStream, updateTime);
     return;
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
```

sendResponseBody(int,BFILE)

構文

```
public void sendResponseBody(int length, oracle.sql.BFILE bfile)
```

説明

データベースから BFILE のコンテンツを検索し、Response Body としてブラウザへ配信します。コール元は、HTTP ヘッダーを作成する必要があります。

パラメータ

length
データ長を指定する整数

bfile
コンテンツの検索元の oracle.sql.BFILE

戻り値

なし

例外

java.lang.IllegalStateException

HttpServletRequest が指定されていない場合、この例外がスローされます。

java.lang.IllegalArgumentException

長さが負の値の場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
OraclePreparedStatement stmt = (OraclePreparedStatement)conn.prepareStatement
     ("select mimetype, len, doc from docfiles where id = ?");
stmt.setString(1, request.getParameter("id"));
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (rset.next( )){
     String mimeType = rset.getString(1);
     int len = rset.getInt(2);
     BFILE bfile = rset.getBFILE(3);
     response.setContentLength(len);
     response.setContentType(mimeType);
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler( );
     handler.setServletResponse(response);
     handler.sendResponseBody(len, bfile);
     return;
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
```

sendResponseBody(int,BLOB)

構文

```
public void sendResponseBody(int length, oracle.sql.BLOB blob)
```

説明

データベースから BLOB のコンテンツを検索し、Response Body としてブラウザへ配信します。コール元は、HTTP ヘッダーを作成する必要があります。

パラメータ

length
データ長を指定する整数

blob
コンテンツの検索元の oracle.sql.BLOB

戻り値

なし

例外

java.lang.IllegalStateException

HttpServletRequest が指定されていない場合、この例外がスローされます。

java.lang.IllegalArgumentException

長さが負の値の場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
OraclePreparedStatement stmt = (OraclePreparedStatement)conn.prepareStatement
     ("select mimetype, len, doc from docblobs where id = ?");
stmt.setString( 1, request.getParameter("id"));
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (rset.next( )){
     String mimeType = rset.getString(1);
     int len = rset.getInt(2);
     BLOB blob = rset.getBLOB(3);
     response.setContentLength(len);
     response.setContentType(mimeType);
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler( );
     handler.setServletResponse(response);
     handler.sendResponseBody(len, blob);
     return;
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
```

sendResponseBody(int,InputStream)

構文

```
public void sendResponseBody(int length, java.io.InputStream in)
```

説明

InputStream オブジェクトのコンテンツを検索し、ブラウザへ配信します。コール元は、HTTP ヘッダーを作成する必要があります。

パラメータ

length
データ長を指定する整数

in
メディア・データの検索元の InputStream オブジェクト

戻り値

なし

例外

java.lang.IllegalStateException

HttpServletRequest が指定されていない場合、この例外がスローされます。

java.lang.IllegalArgumentException

長さが負の値の場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
OraclePreparedStatement stmt = (OraclePreparedStatement)conn.prepareStatement
     ("select mimetype, len, doc from docblobs where id = ?");
stmt.setString(1, request.getParameter("id"));
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (rset.next( )){
     String mimeType = rset.getString(1);
     int len = rset.getInt(2);
     BLOB blob = rset.getBLOB(3);
     response.setContentLength(len);
     response.setContentType(mimeType);
     InputStream blobInputStream = blob.getBinaryStream( );
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler( );
     handler.setServletResponse(response);
     handler.sendResponseBody(len, blobInputStream);
     return;
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
```

sendVideo( )

構文

```
public void sendVideo(oracle.ord.im.OrdVideo media)
```

説明

OrdVideo オブジェクトからビデオ・クリップを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。

パラメータ

media
oracle.ord.im.OrdVideo 型のオブジェクト

戻り値

なし

例外

java.lang.IllegalStateException

HttpServletRequest または HttpServletResponse が指定されていない場合、この例外がスローされます。

OrdHttpResponseException

ソース・タイプが認識されない場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

```
OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("select video from movies where id = ?");
stmt.setString(1, request.getParameter("id"));
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (rset.next( )){
     OrdVideo media = (OrdVideo)rset.getCustomDatum(1, OrdVideo.getFactory( ));
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler
          (request, response);
     handler.sendVideo(media);
     return;
}
else{
     response.setStatus( response.SC_NOT_FOUND );
}
```

パラメータは次のとおりです。

- media: oracle.ord.im.OrdVideo 型のオブジェクトです。

setBufferSize( )

構文

```
public void setBufferSize(int bufferSize)
```

説明

LOB の読込みおよび応答の書込み操作に使用するバッファ・サイズを設定します。

パラメータ

bufferSize
データ長を指定する整数

戻り値

なし

例外

java.lang.IllegalArgumentException

長さが負の値または 0（ゼロ）の場合、この例外がスローされます。

例

```
OrdHttpResponseHandler handler = new OrdHttpResponseHandler(request, response);
handler.setBufferSize(16000);
```

setServletRequest( )

構文

```
public void setServletRequest(javax.servlet.http.HttpServletRequest
                               request)
```

説明

応答ハンドラに対して ttpServletRequest オブジェクトを指定します。コンストラクタで HttpServletRequest オブジェクトを指定しなかった場合、sendResponseBody メソッド以外の send メソッドをコールするには、このメソッドをコールする必要があります。sendResponseBody メソッドのみをコールする場合は、このメソッドをコールする必要はありません。

パラメータ

request
HttpServletRequest 型のオブジェクト

戻り値

なし

例外

なし

例

```
OrdHttpResponseHandler handler = new OrdHttpResponseHandler( );
handler.setServletRequest(request);
handler.setServletResponse(response);
```

setServletResponse( )

構文

```
public void setServletResponse
  (javax.servlet.http.HttpServletResponseresponse)
```

説明

応答ハンドラに対する HttpServletResponse オブジェクトを指定します。コンストラクタで HttpServletResponse オブジェクトを指定しなかった場合、send メソッドのコール前にこのメソッドをコールする必要があります。

パラメータ

response
HttpServletResponse 型のオブジェクト

戻り値

なし

例外

なし

例

このメソッドの例は、「setServletRequest( )」を参照してください。

OrdHttpJspResponseHandler リファレンス情報

ここでは、OrdHttpJspResponseHandler クラスのメソッドに関するリファレンス情報を説明します。

OrdHttpJspResponseHandler クラスを使用すると、Oracle データベースからのメディア・データの検索、および JSP からブラウザまたは別の HTTP クライアントへのメディア・データの配信が簡単になります。

このクラスは、OrdHttpResponseHandler クラスから DEFAULT_BUFFER_SIZE フィールドを継承します。

JSP エンジンに関する重要な注意事項

JSP エンジンは、サーブレットのバイナリ出力ストリームへのアクセスをサポートする必要はありません。そのため、OrdHttpJspResponseHandler クラスを使用したメディア・データの配信をサポートしない JSP エンジンもあります。

*inter*Media 型を使用してデータベースに格納されるすべてのメディア・データ（OrdDoc 型を使用して格納されるテキスト・ドキュメントを含む）は、バイナリ LOB データ型を使用して格納されます。データベースに内部的に格納されるメディア・データは、BLOB を使用して格納されます。データベース外部のオペレーティング・システム・ファイルに格納されるメディア・データは、BFILE を使用して格納されます。そのため、すべてのメディア・データは、ServletOutputStream クラスを使用するサーブレット・バイナリ出力ストリームを介してブラウザに配信されます。

OrdHttpJspResponseHandler クラスのすべての send メソッドは、jsp:forward タグの初期処理をミラー化します。具体的には、バイナリ出力ストリームを取得する前に、JspWriter.clear メソッドをコールしてページの出力バッファを消去します。ただし、JSP エンジンは、JSP 内からの ServletResponse.getOutputStream メソッドへのコールのサポートに必須ではありません。このコールをサポートしない JSP エンジンでは、通常、getOutputStream メソッドで IllegalStateException エラーが発生します。ただし、実際の動作は実装固有です。

JSP エンジンが、JSP 内からのバイナリ出力ストリームへのアクセスをサポートしていない場合は、サーブレットを使用してメディア・データを配信する必要があります。たとえば、次のいずれかの操作を行います。

- jsp:forward タグを使用して、マルチメディア検索要求をサーブレットに転送します。
- マルチメディア検索 URL を作成して、サーブレットからデータを直接検索します。

return 文に関する重要な注意事項

JSP からメディア・データを配信する場合は、OrdHttpJspResponseHandler クラスの send メソッドをコールした後に、常に、return 文が必要です。return 文は、メディア・データの後で JSP の出力ストリームに他のデータが書き込まれないようにするために必要です。

`if ( true ) { ... return; }` 構成を使用すると、コンパイル済ページの終わりで、JSP エンジンが生成した追加コードによって発生する「statement not reachable」エラーを回避できる可能性があります。このように、いくつかの JSP エンジンが生成したコードを厳密にミラー化して、`<jsp:forward ... >` ディレクティブを処理する構成については、後述の例を参照してください。

> **注意：** OrdImage オブジェクトなどの *inter*Media Java オブジェクトは、JDBC 文またはオブジェクト取得元の結果セットには依存しません。ただし、オブジェクトで使用され、SQL 文が実行されたか、結果セットの取得元となった JDBC 接続には依存します。このため、データベースから *inter*Media Java オブジェクトを取得すると、メディア・データをブラウザに配信するまで、アプリケーションでは JDBC 接続を解放できません。

このクラスのすべての send メソッドは、イメージを配信する前に、JspWriter.clear メソッドをコールしてページの出力バッファを消去します。そのため、ページは、デフォルトのバッファ出力モデルを使用する必要があります。

次に、OrdHttpJspResponseHandler クラスを使用して、データベースからイメージを検索し、ブラウザに配信する例を示します。return 文を使用すると、最後の終了タグ（`%>`）の後続の改行文字が、イメージの一部としてブラウザに送信されることはありません。

`if ( true ) { ... return; }` 構成は、コンパイル済ページの終わりで、JSP エンジンが生成した追加文のためにこの例で発生する「statement not reachable」エラーの回避のために使用されます。

```
<%@ page language="java" %>
<%@ page import="OrdSamplePhotoAlbumBean" %>
<%@ page import="oracle.ord.im.OrdHttpJspResponseHandler" %>

<jsp:useBean id="photos" scope="page"
             class="OrdSamplePhotoAlbumBean"/>
<jsp:useBean id="handler" scope="page"
             class="oracle.ord.im.OrdHttpJspResponseHandler"/>

<%
     // Select the entry from the table using the ID request parameter,
     // then fetch the row.

     photos.selectRowById(request.getParameter("id"));
     if (!photos.fetch( )){
```

```
            response.setStatus(response.SC_NOT_FOUND);
            return;
        }

        // Set the page context for the retrieve request, then retrieve
        // the image from the database and deliver it to the browser. The
        // getImage( ) method returns an object of type oracle.ord.im.OrdImage.

        if (true){
            handler.setPageContext(pageContext);
            handler.sendImage(photos.getImage( ));
            return;
        }
%>
```

OrdHttpJspResponseHandler( )

構文

```
public OrdHttpJspResponseHandler( )
```

説明

マルチメディア検索要求への応答を処理する OrdHttpJspResponseHandler オブジェクトを作成します。その後、アプリケーションは、setPageContext( ) メソッドをコールして PageContext オブジェクトを指定する必要があります。

パラメータ

なし

戻り値

なし

例外

なし

例

OrdHttpJspResponseHandler クラスが JavaBean として使用された場合、デフォルトのコンストラクタは、通常は暗黙的に起動されます。コンストラクタの暗黙的な使用例については、「setPageContext( )」を参照してください。

OrdHttpJspResponseHandler(PageContext)

構文

```
public OrdHttpJspResponseHandler(javax.servlet.jsp.PageContext
                                 pageContext)
```

説明

マルチメディア検索要求への応答を処理する OrdHttpJspResponseHandler オブジェクトを作成し、応答ハンドラに対して PageContext オブジェクトを指定します。

パラメータ

pageContext
PageContext 型のオブジェクト

戻り値

なし

例外

なし

例

```
OrdHttpJspResponseHandler handler = new OrdHttpJspResponseHandler(pageContext);
```

sendAudio( )

構文

```
public void sendAudio(oracle.ord.im.OrdAudio media)
```

説明

OrdAudio オブジェクトからオーディオ・クリップを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。このメソッドは、オーディオ・クリップを配信する前に、JspWriter.clear メソッドをコールしてページの出力バッファを消去します。そのため、ページは、デフォルトのバッファ出力モデルを使用する必要があります。

パラメータ

media
oracle.ord.im.OrdAudio 型のオブジェクト

戻り値

このメソッドは、OrdHttpResponseHandler クラスの sendAudio( ) メソッドを上書きします。

例外

java.lang.IllegalStateException

PageContext が指定されていない場合、この例外がスローされます。

OrdHttpResponseException

ソース・タイプが認識されない場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

OrdHttpJspResponseHandler.sendAudio( ) メソッドは、OrdHttpResponseHandler.sendAudio( ) メソッドを拡張したものです。ベース・クラスでのこのメソッドの例については、「OrdHttpJspResponseHandler リファレンス情報」の「sendAudio( )」を参照してください。

sendDoc( )

構文

```
public void sendDoc(oracle.ord.im.OrdDoc media)
```

説明

OrdDoc オブジェクトからメディア・データを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。このメソッドは、メディアを配信する前に、JspWriter.clear メソッドをコールしてページの出力バッファを消去します。そのため、ページは、デフォルトのバッファ出力モデルを使用する必要があります。

パラメータ

media
oracle.ord.im.OrdDoc 型のオブジェクト

戻り値

このメソッドは、OrdHttpResponseHandler クラスの ssendDoc( ) メソッドを上書きします。

例外

java.lang.IllegalStateException

PageContext が指定されていない場合、この例外がスローされます。

OrdHttpResponseException

ソース・タイプが認識されない場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

OrdHttpJspResponseHandler.sendDoc( ) メソッドは、OrdHttpResponseHandler.sendDoc( ) メソッドを拡張したものです。ベース・クラスでのこのメソッドの例については、「OrdHttpResponseHandler リファレンス情報」の「sendDoc( )」を参照してください。

sendImage( )

構文

```
public void sendImage(oracle.ord.im.OrdImage media)
```

説明

OrdImage オブジェクトからイメージを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。このメソッドは、イメージを配信する前に、JspWriter.clear メソッドをコールしてページの出力バッファを消去します。そのため、ページは、デフォルトのバッファ出力モデルを使用する必要があります。

パラメータ

media
oracle.ord.im.OrdAudio 型のオブジェクト

戻り値

このメソッドは、OrdHttpResponseHandler クラスの sendImage( ) メソッドを上書きします。

例外

java.lang.IllegalStateException

PageContext が指定されていない場合、この例外がスローされます。

OrdHttpResponseException

ソース・タイプが認識されない場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

OrdHttpJspResponseHandler.sendImage( ) メソッドは、OrdHttpResponseHandler.sendImage( ) メソッドを拡張したものです。ベース・クラスでのこのメソッドの例については、「OrdHttpResponseHandler リファレンス情報」の「sendImage( )」を参照してください。

sendResponse(String,int,BFILE,Timestamp)

構文

```
public void sendResponse(String contentType, int length,
                         oracle.sql.BFILE bfile, java.sql.Timestamp
                         lastModified)
```

説明

HTTP Response ヘッダーを作成して、データベースから BFILE のコンテンツを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。このメソッドは、イメージを配信する前に、JspWriter.clear メソッドをコールしてページの出力バッファを消去します。そのため、ページは、デフォルトのバッファ出力モデルを使用する必要があります。

パラメータ

contentType
コンテンツの MIME タイプを指定する文字列

length
データ長を指定する整数

bfile
oracle.sql.BFILE 型のオブジェクト

lastModified
データが最後に更新された日付および時刻を指定する java.sql.Timestamp オブジェクト（更新日付および時刻が使用できない場合は null）

戻り値

このメソッドは、OrdHttpResponseHandler クラスの sendResponse( ) メソッドを上書きします。

例外

java.lang.IllegalStateException

PageContext が指定されていない場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

java.lang.IllegalArgumentException

長さが負の値の場合、この例外がスローされます。

例

OrdHttpJspResponseHandler.sendResponse(String, int, BFILE, Timestamp) メソッドは、OrdHttpResponseHandler.sendResponse(String, int, BFILE, Timestamp) メソッドを拡張したものです。ベース・クラスでのこのメソッドの例については、「OrdHttpResponseHandler リファレンス情報」の「sendResponse(String,int,BFILE,Timestamp)」を参照してください。

sendResponse(String,int,BLOB,Timestamp)

構文

```
public void sendResponse(String contentType, int length,
                         oracle.sql.BLOB blob, java.sql.Timestamp
                         lastModified)
```

説明

HTTP Response ヘッダーを作成して、データベースから BLOB のコンテンツを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。このメソッドは、イメージを配信する前に、JspWriter.clear メソッドをコールしてページの出力バッファを消去します。そのため、ページは、デフォルトのバッファ出力モデルを使用する必要があります。

パラメータ

contentType
コンテンツの MIME タイプを指定する文字列

length
データ長を指定する整数

blob
oracle.sql.BLOB 型のオブジェクト

lastModified
データが最後に更新された日付および時刻を指定する java.sql.Timestamp オブジェクト（更新日付および時刻が使用できない場合は null）

戻り値

このメソッドは、OrdHttpResponseHandler クラスの sendResponse( ) メソッドを上書きします。

例外

java.lang.IllegalStateException

PageContext が指定されていない場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

java.lang.IllegalArgumentException

長さが負の値の場合、この例外がスローされます。

例

OrdHttpJspResponseHandler.sendResponse(String, int, BLOB, Timestamp) メソッドは、OrdHttpResponseHandler.sendResponse(String, int, BLOB, Timestamp) メソッドを拡張したものです。ベース・クラスでのこのメソッドの例については、「OrdHttpResponseHandler リファレンス情報」の「sendResponse(String,int,BLOB,Timestamp)」を参照してください。

sendResponse(String,int,InputStream,Timestamp)

構文

```
public void sendResponse(String contentType, int length,
                         java.io.InputStream in, java.sql.Timestamp
                         lastModified)
```

説明

HTTP Response ヘッダーを作成して、InputStream オブジェクトのコンテンツを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。このメソッドは、イメージを配信する前に、JspWriter.clear メソッドをコールしてページの出力バッファを消去します。そのため、ページは、デフォルトのバッファ出力モデルを使用する必要があります。

パラメータ

contentType
コンテンツの MIME タイプを指定する文字列

length
データ長を指定する整数

in
メディア・データの検索元の InputStream オブジェクト

lastModified
データが最後に更新された日付および時刻を指定する java.sql.Timestamp オブジェクト（更新日付および時刻が使用できない場合は null）

戻り値

このメソッドは、OrdHttpResponseHandler クラスの sendResponse( ) メソッドを上書きします。

例外

java.lang.IllegalStateException

PageContext が指定されていない場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

java.lang.IllegalArgumentException

長さが負の値の場合、この例外がスローされます。

例

OrdHttpJspResponseHandler.sendResponse(String, int, InputStream, Timestamp) メソッドは、OrdHttpResponseHandler.sendResponse(String, int, InputStream, Timestamp) メソッドを拡張したものです。ベース・クラスでのこのメソッドの例については、「OrdHttpResponseHandler リファレンス情報」の「sendResponse(String,int,InputStream,Timestamp)」を参照してください。

sendVideo( )

構文

```
public void sendVideo(oracle.ord.im.OrdVideo media)
```

説明

OrdVideo オブジェクトからビデオ・クリップを検索し、ブラウザへ配信します。

このメソッドは、If-Modified-Since ヘッダーおよび Last-Modified ヘッダーのサポートによって、ブラウザのコンテンツのキャッシュをサポートします。このメソッドは、ビデオ・クリップを配信する前に、JspWriter.clear メソッドをコールしてページの出力バッファを消去します。そのため、ページは、デフォルトのバッファ出力モデルを使用する必要があります。

パラメータ

media
oracle.ord.im.OrdVideo 型のオブジェクト

戻り値

このメソッドは、OrdHttpResponseHandler クラスの sendVideo( ) メソッドを上書きします。

例外

java.lang.IllegalStateException

PageContext が指定されていない場合、この例外がスローされます。

OrdHttpResponseException

ソース・タイプが認識されない場合、この例外がスローされます。

javax.servlet.ServletException

バイナリ出力ストリームへのアクセス中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データ読込みのための InputStream オブジェクトの取得中にエラーが発生した場合、この例外がスローされます。

java.io.IOException

メディア・データの読込み中にエラーが発生した場合、この例外がスローされます。

例

OrdHttpJspResponseHandler.sendVideo( ) メソッドは、OrdHttpResponseHandler.sendVideo( ) メソッドを拡張したものです。ベース・クラスでのこのメソッドの例については、「OrdHttpResponseHandler リファレンス情報」の「sendVideo( )」を参照してください。

setPageContext( )

構文

```
public void setPageContext(javax.servlet.jsp.PageContext
                           pageContext)
```

説明

応答ハンドラに対して PageContext オブジェクトを指定します。コンストラクタで PageContext オブジェクトを指定しなかった場合、send メソッドのコール前にこのメソッドをコールする必要があります。

パラメータ

pageContext
PageContext 型のオブジェクト

戻り値

なし

例外

なし

例

```
<jsp:useBean id="handler" scope="page"
             class="oracle.ord.im.OrdHttpJspResponseHandler"/>
<%
     OraclePreparedStatement stmt = (OraclePreparedStatement)
          conn.prepareStatement("select image from photos where id = ?");
     stmt.setString(1, request.getParameter("id"));
     OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
     if (rset.next( )){
          OrdImage media = (OrdImage)rset.getCustomDatum(1,
               OrdImage.getFactory( ));
          handler.setPageContext(pageContext);
          handler.sendImage(image);
          return;
     }else{
          response.setStatus(response.SC_NOT_FOUND);
     }
%>
```

OrdHttpUploadFile リファレンス情報

ここでは、OrdHttpUploadFile クラスのメソッドに関するリファレンス情報を説明します。

HTML 形式を使用したフォーム・ベースのファイルのアップロードでは、multipart/form-data フォーマットを使用して、POST 要求のフォーム・データおよびアップロード・ファイルをエンコードします。OrdHttpUploadFile クラスは、OrdHttpUploadFormData クラスによって解析済のアップロード・ファイルを表すために使用されます（OrdHttpUploadFormData クラスの詳細は、「OrdHttpUploadFormData リファレンス情報」を参照してください）。OrdHttpUploadFile クラスは、アップロード・ファイルに関する情報を取得し、ファイルのコンテンツに直接アクセスして、データベースの *inter*Media オブジェクトにコンテンツを簡単にロードするためのメソッドを提供します。

フィールドに有効なファイル名前が入力されたかどうかにかかわらず、HTML フォームにある FILE タイプのすべての入力フィールドは、OrdHttpUploadFile のパラメータを生成します。要件によっては、アプリケーションは、ユーザーが有効なファイル名を入力したかどうか、およびファイルがブラウザに正常にアップロードされたかどうかを判断するために、ファイル名の長さまたはコンテンツの長さ（あるいはその両方）をテストすることができます。たとえば、ユーザーがファイル名を入力していない場合、getOriginalFileName( ) メソッドが戻す文字列の長さは、0（ゼロ）になります。ただし、ユーザーが無効なファイル名や長さが 0（ゼロ）の空のファイル名を入力した場合、ファイル名の長さは 0（ゼロ）にならず、getContentLength( ) メソッドが戻すコンテンツ長が 0（ゼロ）になります。

getContentLength( )

構文

```
public int getContentLength( )
```

説明

アップロードされたメディア・ファイルの長さを戻します。存在しないファイル名、空のファイル名などの無効なファイル名を入力した場合は、0（ゼロ）を戻します。

パラメータ

なし

戻り値

アップロード・ファイルの長さを戻します。

例外

なし

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
...
OrdHttpUploadFile photo = formData.getFileParameter("photo");
String mimeType = photo.getMimeType( );
int i = photo.getContentLength( );
if (i == 0){
     displayUserError("The file is empty, invalid, or nonexistent");
}
...
photo.release( );
```

getInputStream( )

構文

```
public java.io.InputStream getInputStream( )
```

説明

アップロードされたデータの直接の読込みに使用可能な InputStream オブジェクトを戻します。アプリケーションは、終了時に、close( ) メソッドを使用してストリームをクローズする必要があります。

パラメータ

なし

戻り値

コンテンツの読込み元の InputStream オブジェクトを戻します。

例外

java.lang.IllegalStateException

アップロード・ファイルが、解放されたため使用できない場合、この例外がスローされます。

java.io.IOException

一時ファイルのオープン中にエラーが発生した場合、この例外がスローされます。

例

```
OrdImage dbImage = (OrdImage)rset.getCustomDatum(1, OrdImage.getFactory( ));
OrdHttpUploadFile uploadImage = formData.getFileParameter("photo");
InputStream photoInputStream = uploadImage.getInputStream( );
try{
     dbImage.loadDataFromInputStream(photoInputStream);
}
finally{
     photoInputStream.close( );
}
```

getMimeType( )

構文

```
public String getMimeType( )
```

説明

ファイルのアップロード時にブラウザが判断する、メディア・ファイルの MIME タイプを戻します。

一部のブラウザは、ファイル名を指定していない場合も、デフォルトの MIME タイプを戻します。そのためアプリケーションは、ファイルが正常にアップロードされたことを確認するために、ファイル名またはコンテンツ長を確認する必要があります。

パラメータ

なし

戻り値

ファイルの MIME タイプを、文字列で戻します。

例外

なし

例

このメソッドの例は、「getContentLength( )」を参照してください。

getOriginalFileName( )

構文

```
public String getOriginalFileName( )
```

説明

ブラウザが指定した元のファイル名を戻します。ファイル名が指定されていない場合、空の文字列を戻します。

パラメータ

なし

戻り値

ファイル名を、文字列で戻します。

例外

なし

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
...
OrdHttpUploadFile photo = formData.getFileParameter("photo");
String originalName = photo.getOriginalFileName( );
...
formData.release( );
```

getSimpleFileName( )

構文

```
public String getSimpleFileName( )
```

説明

単純なファイル名（ファイル名および拡張子）を戻します。ファイル名が指定されていない場合、空の文字列を戻します。

パラメータ

なし

戻り値

単純なファイル名を、文字列で戻します。

例外

なし

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
...
OrdHttpUploadFile photo = formData.getFileParameter("photo");
String name = photo.getSimpleFileName( );
...
formData.release( );
```

loadAudio(OrdAudio)

構文

```
public void loadAudio(oracle.ord.im.OrdAudio media)
```

説明

アップロード・ファイルを OrdAudio Java オブジェクトにロードし、オーディオ・データに基づいてプロパティを設定します。また、データベースにオーディオ・データをロードし、OrdAudio.setProperties( ) メソッドをコールして MIME タイプなどのプロパティを設定します。このメソッドは、既存のフォーマット・プラグインのコンテキスト情報を使用せず、プロパティの設定時にコメントを設定しません。このメソッドを使用するには、アプリケーションで、初期化された OrdAudio オブジェクトをデータベースからフェッチし、メソッドをコールしてオーディオ・データをデータベースにロードした後で、データベースの OrdAudio オブジェクトを更新します。

オーディオ・フォーマットが認識されないため setProperties( ) メソッドのコールに失敗した場合、このメソッドは、次のプロパティを設定します。

- MIME タイプ（ブラウザが指定した値に設定）
- 更新時刻（現在の日付および時刻に設定）

パラメータ

media
オーディオ・データのロード先の oracle.ord.im.OrdAudio オブジェクト

戻り値

なし

例外

java.io.IOException

メディア・データの読込みまたは書込み中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データの格納中に認識できないエラーが発生した場合、この例外がスローされます。

java.lang.IllegalStateException

アップロード・ファイルが、解放されたため使用できない場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
String id = formData.getParameter("id");
OrdHttpUploadFile uploadFile = formData.getFileParameter("soundfile");

OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("insert into songs (id,sound) values(?,
     ORDSYS.ORDAUDIO.INIT( ))");
stmt.setString(1, id);
stmt.executeUpdate( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("select sound from
     songs where id = ? for update");
stmt.setString(1, id);
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (!rset.next( )){
     throw new ServletException("new row not found in table");
}

OrdAudio sound = (OrdAudio)rset.getCustomDatum(1, OrdAudio.getFactory( ));
uploadFile.loadAudio(sound);
formData.release( );

rset.close( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("update songs set
     sound = ? where id = ?");
stmt.setCustomDatum(1, sound);
stmt.setString(2, id);
stmt.execute( );
stmt.close( );
conn.commit( );
```

loadAudio(OrdAudio,byte[ ][ ], boolean)

構文

```
public void loadAudio(oracle.ord.im.OrdAudio media,
                      byte[ ][ ] ctx, boolean setComments)
```

説明

アップロード・ファイルを OrdAudio Java オブジェクトにロードし、アプリケーションで指定されたフォーマット・プラグインのコンテキストを使用してプロパティを設定します。また、データベースにオーディオ・データをロードし、OrdAudio.setProperties( ) メソッドをコールして MIME タイプなどのプロパティを設定します。アプリケーションでは、フォーマット・プラグインのコンテキスト情報が提供され、OrdAudio オブジェクトのコメントを設定するかどうかが選択されます。このメソッドを使用するには、アプリケーションで、初期化された OrdAudio オブジェクトをデータベースからフェッチし、メソッドをコールしてオーディオ・データをデータベースにロードした後で、データベースの OrdAudio オブジェクトを更新します。

オーディオ・フォーマットが認識されないため setProperties( ) メソッドのコールに失敗した場合、このメソッドは、次のプロパティを設定します。

- MIME タイプ（ブラウザが指定した値に設定）
- 更新時刻（現在の日付および時刻に設定）

パラメータ

media
オーディオ・データのロード先の oracle.ord.im.OrdAudio オブジェクト

ctx
フォーマット・プラグインのコンテキスト情報

setComments
OrdAudio オブジェクトにコメントを設定するかどうかを示すブール値

戻り値

なし

例外

java.io.IOException

メディア・データの読込みまたは書込み中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データの格納中に認識できないエラーが発生した場合、この例外がスローされます。

java.lang.IllegalStateException

アップロード・ファイルが、解放されたため使用できない場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
String id = formData.getParameter("id");
OrdHttpUploadFile uploadFile = formData.getFileParameter("soundfile");

OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("insert into songs (id,sound) values(?,
     ORDSYS.ORDAUDIO.INIT( ))");
stmt.setString(1, id);
stmt.executeUpdate( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("select sound from
     songs where id = ? for update");
stmt.setString(1, id);
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (!rset.next( )){
     throw new ServletException("new row not found in table");
}

byte[ ] [ ] ctx = new byte[1][64];
OrdAudio sound = (OrdAudio)rset.getCustomDatum(1, OrdAudio.getFactory( ));
uploadFile.loadAudio(sound, ctx, true);
formData.release( );

rset.close( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("update songs set
     sound = ? where id = ?");
```

```
stmt.setCustomDatum(1, sound);
stmt.setString(2, id);
stmt.execute( );
stmt.close( );
conn.commit( );
```

loadBlob( )

構文

```
public void loadBlob(oracle.sql.BLOB blob)
```

説明

アップロードされたメディア・ファイルを BLOB にロードします。

パラメータ

blob
データのロード先の oracle.sql.BLOB オブジェクト

戻り値

なし

例外

java.io.IOException

メディア・データの読込みまたは書込み中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データの格納中に認識できないエラーが発生した場合、この例外がスローされます。

java.lang.IllegalStateException

アップロード・ファイルが、解放されたため使用できない場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
String id = formData.getParameter("id");
OrdHttpUploadFile uploadFile = formData.getFileParameter("docfile");

OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("insert into docs (id,doc_blob) values
     (?,EMPTY_BLOB( ))");
stmt.setString(1, id);
```

```
stmt.executeUpdate( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("select doc_blob
     from docs where id = ? for update" );
stmt.setString(1, id);
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (!rset.next( )){
     throw new ServletException("new row not found in table");
}

BLOB docBlob = rset.getBLOB(1);
uploadFile.loadBlob(docBlob);
formData.release( );

rset.close( );
stmt.close( );
conn.commit( );
```

loadDoc(OrdDoc)

構文

```
public void loadDoc(oracle.ord.im.OrdDoc media)
```

説明

アップロード・ファイルを OrdDoc Java オブジェクトにロードし、ドキュメントのコンテンツに基づいてプロパティを設定します。また、データベースにドキュメントをロードし、OrdDoc.setProperties( ) メソッドをコールして MIME タイプなどのプロパティを設定します。このメソッドは、既存のフォーマット・プラグインのコンテキスト情報を使用せず、プロパティの設定時にコメントを設定しません。このメソッドを使用するには、アプリケーションで、初期化された OrdDoc オブジェクトをデータベースからフェッチし、メソッドをコールしてドキュメントをデータベースにロードした後で、データベースの OrdDoc オブジェクトを更新します。

ドキュメント・フォーマットが認識されないため setProperties( ) メソッドのコールに失敗した場合、このメソッドは、次のプロパティを設定します。

- MIME タイプ（ブラウザが指定した値に設定）
- コンテンツ長（アップロード・ファイルの長さに設定）
- 更新時刻（現在の日付および時刻に設定）

パラメータ

media
ドキュメントのロード先の oracle.ord.im.OrdDoc オブジェクト

戻り値

なし

例外

java.io.IOException

メディア・データの読込みまたは書込み中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データの格納中に認識できないエラーが発生した場合、この例外がスローされます。

java.lang.IllegalStateException

アップロード・ファイルが、解放されたため使用できない場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
String id = formData.getParameter("id");
OrdHttpUploadFile uploadFile = formData.getFileParameter("docfile");

OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("insert into documents (id,doc) values(?,
     ORDSYS.ORDDOC.INIT( ))");
stmt.setString(1, id);
stmt.executeUpdate( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("select doc from
     documents where id = ? for update");
stmt.setString(1, id);
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (!rset.next( )){
     throw new ServletException("new row not found in table");
}

OrdDoc media = (OrdDoc)rset.getCustomDatum(1, OrdDoc.getFactory( ));
uploadFile.loadDoc(media);
formData.release( );

rset.close( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("update documents set
     doc = ? where id = ?");
stmt.setCustomDatum(1, doc);
stmt.setString(2, id);
stmt.execute( );
stmt.close( );
conn.commit( );
```

loadDoc(OrdDoc,byte[ ][ ],boolean)

構文

```
public void loadDoc(oracle.ord.im.OrdDoc media,
                    byte[ ][ ] ctx, boolean setComments)
```

説明

アップロード・ファイルを OrdDoc Java オブジェクトにロードし、アプリケーションで指定されたフォーマット・プラグインのコンテキストを使用してプロパティを設定します。また、データベースにドキュメントをロードし、OrdDoc.setProperties( ) メソッドをコールして MIME タイプなどのプロパティを設定します。アプリケーションでは、フォーマット・プラグインのコンテキスト情報が提供され、OrdDoc オブジェクトのコメントを設定するかどうかが選択されます。このメソッドを使用するには、アプリケーションで、初期化された OrdDoc オブジェクトをデータベースからフェッチし、メソッドをコールしてドキュメントをデータベースにロードした後で、データベースの OrdDoc オブジェクトを更新します。

ドキュメント・フォーマットが認識されないため setProperties( ) メソッドのコールに失敗した場合、このメソッドは、次のプロパティを設定します。

- MIME タイプ（ブラウザが指定した値に設定）
- コンテンツ長（アップロード・ファイルの長さに設定）
- 更新時刻（現在の日付および時刻に設定）

パラメータ

media
ドキュメントのロード先の oracle.ord.im.OrdDoc オブジェクト

ctx
フォーマット・プラグインのコンテキスト情報

setComments
オブジェクトにコメントを設定するかどうかを示すブール値

戻り値

なし

例外

java.io.IOException

メディア・データの読込みまたは書込み中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データの格納中に認識できないエラーが発生した場合、この例外がスローされます。

java.lang.IllegalStateException

アップロード・ファイルが、解放されたため使用できない場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
String id = formData.getParameter("id");
OrdHttpUploadFile uploadFile = formData.getFileParameter("docfile");

OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("insert into documents (id,doc) values(?,
     ORDSYS.ORDDOC.INIT( ))");
stmt.setString(1, id);
stmt.executeUpdate( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("select doc from
     documents where id = ? for update");
stmt.setString(1, id);
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (!rset.next( )){
     throw new ServletException("new row not found in table");
}

byte[ ] [ ] ctx = new byte[1][64];
OrdDoc media = (OrdDoc)rset.getCustomDatum(1, OrdDoc.getFactory( ));
uploadFile.loadDoc(media, ctx, true);
formData.release( );

rset.close( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("update documents set
     doc = ? where id = ?");
```

```
stmt.setCustomDatum(1, doc);
stmt.setString(2, id);
stmt.execute( );
stmt.close( );
conn.commit( );
```

loadImage(OrdImage)

構文

```
public void loadImage(oracle.ord.im.OrdImage media)
```

説明

アップロード・ファイルを OrdImage Java オブジェクトにロードし、イメージ・コンテンツに基づいてプロパティを設定します。また、データベースにイメージ・コンテンツをロードし、OrdImage.setProperties( ) メソッドをコールして MIME タイプ、長さ、高さ、幅などのプロパティを設定します。このメソッドを使用するには、アプリケーションで、初期化された OrdImage オブジェクトをデータベースからフェッチし、メソッドをコールしてイメージ・コンテンツをデータベースにロードした後で、データベースの OrdImage オブジェクトを更新します。

イメージ・フォーマットが認識されないため setProperties( ) メソッドのコールに失敗した場合、このメソッドは、次のプロパティを設定します。

- MIME タイプ（ブラウザが指定した値に設定）
- コンテンツ長（アップロード・ファイルの長さに設定）
- 更新時刻（現在の日付および時刻に設定）

パラメータ

media
イメージ・データのロード先の oracle.ord.im.OrdImage オブジェクト

戻り値

なし

例外

java.io.IOException

メディア・データの読込みまたは書込み中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データの格納中に認識できないエラーが発生した場合、この例外がスローされます。

java.lang.IllegalStateException

アップロード・ファイルが、解放されたため使用できない場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
String id = formData.getParameter("id");
OrdHttpUploadFile uploadFile = formData.getFileParameter("photofile");

OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("insert into photos (id,photo) values(?,
     ORDSYS.ORDIMAGE.INIT( ))");
stmt.setString(1, id);
stmt.executeUpdate( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("select image from
     photos where id = ? for update");
stmt.setString(1, id);
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (!rset.next( )){
     throw new ServletException("new row not found in table");
}

OrdImage photo = (OrdImage)rset.getCustomDatum(1, OrdImage.getFactory( ));
uploadFile.loadImage(photo);
formData.release( );

rset.close( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("update photos set
     photo = ? where id = ?");
stmt.setCustomDatum(1, photo);
stmt.setString(2, id);
stmt.execute( );
stmt.close( );
conn.commit( );
```

loadImage(OrdImage,String)

構文

```
public void loadImage(oracle.ord.im.OrdImage media, String cmd)
```

説明

アップロード・ファイルを OrdImage Java オブジェクトにロードし、アプリケーションが指定したコマンド文字列に従ってプロパティを設定します。このメソッドを使用するには、アプリケーションで、初期化された OrdImage オブジェクトをデータベースからフェッチし、メソッドをコールしてイメージ・コンテンツをデータベースにロードした後で、データベースの OrdImage オブジェクトを更新します。

パラメータ

media
アップロードされたデータのロード先の oracle.ord.im.OrdImage オブジェクト

cmd
設定するプロパティを指定する文字列

戻り値

なし

例外

java.io.IOException

メディア・データの読込みまたは書込み中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データの格納中に認識できないエラーが発生した場合、この例外がスローされます。

java.lang.IllegalStateException

アップロード・ファイルが、解放されたため使用できない場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
String id = formData.getParameter("id");
OrdHttpUploadFile uploadFile = formData.getFileParameter("photofile");

OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("insert into photos (id,photo) values(?,
     ORDSYS.ORDIMAGE.INIT( ))");
stmt.setString(1, id);
stmt.executeUpdate( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("select photo from
     photos where id = ? for update");
stmt.setString(1, id);
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (!rset.next( )){
     throw new ServletException("new row not found in table");
}

OrdImage photo = (OrdImage)rset.getCustomDatum(1, OrdImage.getFactory( ));
String cmd = getImagePropertiesCommand(photo);
uploadFile.loadImage(photo, cmd);
formData.release( );

rset.close( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("update photos set
     photo = ? where id = ?");
stmt.setCustomDatum(1, photo);
stmt.setString(2, id);
stmt.execute( );
stmt.close( );
conn.commit( );
```

loadVideo(OrdVideo)

構文

```
public void loadVideo(oracle.ord.im.OrdVideo media)
```

説明

アップロード・ファイルを OrdVideo Java オブジェクトにロードし、ビデオ・データに基づいてプロパティを設定します。また、データベースにビデオ・データをロードし、OrdVideo.setProperties( ) メソッドをコールして MIME タイプなどのプロパティを設定します。このメソッドは、既存のフォーマット・プラグインのコンテキスト情報を使用せず、プロパティの設定時にコメントを設定しません。このメソッドを使用するには、アプリケーションで、初期化された OrdVideo オブジェクトをデータベースからフェッチし、このメソッドをコールしてビデオ・データをデータベースにロードした後で、データベースの OrdVideo オブジェクトを更新します。

ビデオ・フォーマットが認識されないために setProperties( ) メソッドのコールが失敗した場合、このメソッドは次のプロパティを設定します。

- MIME タイプ（ブラウザが指定した値に設定）
- 更新時刻（現在の日付および時刻に設定）

パラメータ

media
アップロードされたビデオ・データのロード先の oracle.ord.im.OrdVideo オブジェクト

戻り値

なし

例外

java.io.IOException

メディア・データの読込みまたは書込み中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データの格納中に認識できないエラーが発生した場合、この例外がスローされます。

java.lang.IllegalStateException

アップロード・ファイルが、解放されたため使用できない場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
String id = formData.getParameter("id");
OrdHttpUploadFile uploadFile = formData.getFileParameter("videofile");

OraclePreparedStatement stmt = (OraclePreparedStatement)
    conn.prepareStatement("insert into movies (id,video) values(?,
    ORDSYS.ORDVIDEO.INIT( ))");
stmt.setString(1, id);
stmt.executeUpdate( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("select video from
    movies where id = ? for update");
stmt.setString(1, id);
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (!rset.next( )){
    throw new ServletException("new row not found in table");
}

OrdVideo media = (OrdVideo)rset.getCustomDatum(1, OrdVideo.getFactory( ));
uploadFile.loadVideo(media);
formData.release( );

rset.close( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("update movies set
    video = ? where id = ?");
stmt.setCustomDatum(1, video);
stmt.setString(2, id);
stmt.execute( );
stmt.close( );
conn.commit( );
```

loadVideo(OrdVideo,byte[ ][ ],boolean)

構文

```
public void loadVideo(oracle.ord.im.OrdVideo media,
                      byte[ ][ ] ctx, boolean setComments)
```

説明

アップロード・ファイルを OrdVideo Java オブジェクトにロードし、アプリケーションで指定されたフォーマット・プラグインのコンテキストを使用してプロパティを設定します。また、データベースにビデオ・データをロードし、OrdVideo.setProperties( ) メソッドをコールして MIME タイプなどのプロパティを設定します。アプリケーションでは、フォーマット・プラグインのコンテキスト情報が提供され、OrdVideo オブジェクトのコメントを設定するかどうかが選択されます。このメソッドを使用するには、アプリケーションで、初期化された OrdVideo オブジェクトをデータベースからフェッチし、このメソッドをコールしてビデオ・データをデータベースにロードした後で、データベースの OrdVideo オブジェクトを更新します。

ビデオ・フォーマットが認識されないために setProperties( ) メソッドのコールが失敗した場合、このメソッドは次のプロパティを設定します。

- MIME タイプ（ブラウザが指定した値に設定）
- 更新時刻（現在の日付および時刻に設定）

パラメータ

media
アップロードされたビデオ・データのロード先の oracle.ord.im.OrdVideo オブジェクト

ctx
フォーマット・プラグインのコンテキスト情報

setComments
オブジェクトにコメントを設定するかどうかを示すブール値

戻り値

なし

例外

java.io.IOException

メディア・データの読込みまたは書込み中にエラーが発生した場合、この例外がスローされます。

java.sql.SQLException

メディア・データの格納中に認識できないエラーが発生した場合、この例外がスローされます。

java.lang.IllegalStateException

アップロード・ファイルが、解放されたため使用できない場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
String id = formData.getParameter("id");
OrdHttpUploadFile uploadFile = formData.getFileParameter("videofile");

OraclePreparedStatement stmt = (OraclePreparedStatement)
     conn.prepareStatement("insert into movies (id,video) values(?,
     ORDSYS.ORDVIDEO.INIT( ))");
stmt.setString(1, id);
stmt.executeUpdate( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("select video from
     movies where id = ? for update");
stmt.setString(1, id);
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (!rset.next( )){
     throw new ServletException("new row not found in table");
}

byte[ ] [ ] ctx = new byte[1][64];
OrdVideo media = (OrdVideo)rset.getCustomDatum(1, OrdVideo.getFactory( ));
uploadFile.loadVideo(media, ctx, true);
formData.release( );

rset.close( );
stmt.close( );

stmt = (OraclePreparedStatement)conn.prepareStatement("update movies set
     video = ? where id = ?");
```

```
stmt.setCustomDatum(1, video);
stmt.setString(2, id);
stmt.execute( );
stmt.close( );
conn.commit( );
```

release( )

構文

```
public void release( )
```

説明

OrdHttpUploadFile オブジェクトが保持するすべてのリソースを解放します。具体的には、アップロード・ファイルのコンテンツの保持に使用したメモリーを解放するか、またはアップロード・ファイルのコンテンツの保持に使用した一時ファイルを削除します。アップロード・ファイルの処理を完了した後、アプリケーションは、このメソッドをコールして割り当てられたメモリーを解放し、それをガベージ・コレクションの候補にすることによって、メモリー使用量を最適化できます。

パラメータ

なし

戻り値

なし

例外

なし

例

このメソッドの例は、「getContentLength( )」を参照してください。

OrdHttpUploadFormData リファレンス情報

ここでは、OrdHttpUploadFormData クラスのメソッドに関するリファレンス情報を説明します。

HTML 形式を使用したファイルのアップロードでは、multipart/form-data フォーマットを使用して、POST 要求のフォーム・データおよびアップロード・ファイルをエンコードします。OrdHttpUploadFormData クラスを使用すると、POST データを解析し、通常の形式のフィールドまたはアップロード・ファイルのコンテンツを Java サーブレットまたは JavaServer Pages（JSP）へアクセス可能にすることによって、このような要求の処理が簡単になります。OrdHttpUploadFormData クラスは、テキストベースのフォーム・フィールド・パラメータにアクセスする、ServletRequest クラスが提供する getParameter( )、getParameterValues( ) および getParameterNames( ) メソッドと同じメソッドを提供します。アップロード・ファイルへのアクセスは、getFileParameter( )、getFileParameterValues( ) および getFileParameterNames( ) という、同じ機能を持つ一連のメソッドによって提供されます。getFileParameter( ) メソッドおよび getFileParameterValues( ) メソッドが戻す OrdHttpUploadFile オブジェクトは、各アップロード・ファイルの MIME タイプ、長さおよびコンテンツへのアクセスを簡略化します。

OrdHttpUploadFile クラスの詳細は、「OrdHttpUploadFile リファレンス情報」を参照してください。

次に、OrdHttpUploadFormData クラスの使用例を示します。

```
// Create an OrdHttpUploadFormData object and use it to parse the
// multipart/form-data message.
//
OrdHttpUploadFormData formData = new OrdHttpUploadFormData( request );
formData.parseFormData( );

// Get the description, location, and photograph.
//
String id = formData.getParameter("id");
String description = formData.getParameter("description");
String location = formData.getParameter("location");
OrdHttpUploadFile photo = formData.getFileParameter("photo");

// Prepare and execute a SQL statement to insert a new row into
// the table and return the sequence number for the new row.
// Disable auto-commit to allow the LOB to be written correctly.
//
conn.setAutoCommit(false);
PreparedStatement stmt = conn.prepareStatement("insert into photo_album (id,
     description, location, photo) " + " values (?, ?, ?, ORDSYS.ORDIMAGE.INIT(
))");
stmt.setString(1, id);
```

```
stmt.setString(2, description);
stmt.setString(3, location);
stmt.executeUpdate( );

// Prepare and execute a SQL statement to fetch the new OrdImage
// object from the database.
//
stmt = conn.prepareStatement("select photo from photo_album where id = ? for
     update");
stmt.setString(1, id);
OracleResultSet rset = (OracleResultSet)stmt.executeQuery( );
if (!rset.next( )){
     throw new ServletException("new row not found in table");
}
OrdImage media = (OrdImage)rset.getCustomDatum(1, OrdImage.getFactory( ));

// Load the photograph into the database and set the properties.
//
photo.loadImage(media);

// Prepare and execute a SQL statement to update the image object.
//
stmt = (OraclePreparedStatement)conn.prepareStatement("update photo_album set
     photo = ? where id = ?");
stmt.setCustomDatum(1, media);
stmt.setString(2 id);
stmt.execute( );
stmt.close( );

// Commit the changes.
//
conn.commit( );
```

問合せ文字列のパラメータおよびテキストベースの HTML 形式のフィールド・パラメータの処理に関する注意事項

データがパラメータ名に関連付けられているかどうかにかかわらず、要求の任意の問合せ文字列にあるすべてのパラメータは、対応する文字列のパラメータを生成します。同様に、HTML 形式のテキストベースのすべての入力フィールドも、フィールドにデータが入力されているかどうかにかかわらず、対応する文字列のパラメータを生成します。問合せ文字列のパラメータおよびテキストベースの入力フィールドを処理する場合、アプリケーションは、対応する String オブジェクトの長さを確認し、データが存在するかどうかを判断します。

parseFormData( ) メソッドは、すべての問合せ文字列のパラメータおよびフォーム・フィールド・パラメータを、1 つの順序付けされたパラメータ・セットにマージします。問合せ文字列のパラメータは、最初に処理され、次にフォーム・フィールド・パラメータが処理されます。このため、問合せ文字列のパラメータは、フォーム・フィールド・パラメータよりも優先されます。たとえば、問合せ文字列が arg=hello&arg=world の要求の場合、HTML 形式の 2 つの 'arg' フィールドに、'greetings' および 'everyone' の値が入力されると、結果のパラメータ・セットには、arg=(hello, world, greetings, everyone) というエントリが含まれます。

FILE 型の HTML 形式のフィールド・パラメータ処理に関する注意事項

入力フィールドに有効なファイル名が入力されたかどうかにかかわらず、HTML 形式の FILE 型のすべての入力フィールドは、対応する OrdUploadFile 型のパラメータを生成します。FILE 型のフィールドの処理時に、アプリケーションで、ファイル名の長さ、コンテンツの長さまたはこの 2 つの組合せのいずれかをテストして、ユーザーが有効なファイル名を入力したかどうか、およびファイルがブラウザによって正常にアップロードされたかどうかを判断できます。詳細は、「OrdHttpUploadFile リファレンス情報」の OrdHttpUploadFile クラスを参照してください。

西ヨーロッパ言語以外の使用に関する注意事項

Microsoft 社の Internet Explorer ブラウザでは、フォームの表示に使用されているものとは異なるキャラクタ・セット・エンコーディングを使用して、HTML 形式にデータを入力できます。たとえば、あるブラウザ・ウィンドウから日本語 Shift_JIS キャラクタ・セットのデータをコピーし、西ヨーロッパ（ISO）・キャラクタ・セットを使用して表示している、別のブラウザ・ウィンドウのフォームに貼り付けることができます。この場合、Internet Explorer は、multipart/form-data パーサーが重複データを検出できない方法で、そのフォーム・データを 2 回転送する可能性があります。さらに、西ヨーロッパ言語以外のフォーム・データが、Unicode エスケープ・シーケンスとして送信されたり、RAW バイナリ・フォームに存在したり、または POST データの別の部分で両方のフォーマットを使用して重複する場合があります。

Netscape ブラウザの場合、これらの問題は発生しませんが、正しいキャラクタ・セットが使用されるように注意する必要があります。たとえば、あるブラウザ・ウィンドウから日本語 Shift_JIS キャラクタ・セットのデータをコピーし、西ヨーロッパ（ISO）・キャラクタ・セットを使用して表示している、別のブラウザ・ウィンドウのフォームに貼り付けることはできますが、データをフォーム・フィールドに貼り付けると、それぞれの日本語 Shift_JIS 文字を構成する 2 バイトは、2 つの個別の西ヨーロッパ言語（ISO）文字として格納されます。

そのため、使用する Web ブラウザにかかわらず、正しいキャラクタ・セットを使用して HTML 形式を表示する必要があります。たとえば、HTML META タグを使用して、次のようにキャラクタ・セットを指定できます。

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=Shift_JIS">
```

enableParameterTranslation( )

構文

```
public void enableParameterTranslation(java.lang.Sting encoding)
```

説明

multipart/form-data の POST 要求の body の解析時に、指定されたキャラクタ・エンコーディングを使用して、すべての HTML 形式のパラメータ名およびすべてのテキストベースのパラメータ値を変換できるようにします。

要求パラメータのキャラクタ・エンコーディングは、HTTP 仕様では完全には定義されていません。ほとんどの Servlet コンテナは、サーブレットのデフォルト・エンコーディングである ISO 8859-1 を使用して解析します。そのため、デフォルトでこれらの Servlet コンテナと互換性のある API を指定するには、OrdHttpUploadFormData クラスでデフォルト・エンコーディング ISO 8859-1 を使用して multipart/form-data 要求パラメータも解析します。

parseFormData( ) メソッドをコールする前に、他のキャラクタ・エンコーディングを使用した要求および応答を処理するアプリケーションは、multipart/form-data の POST 要求の body の解析時に、enableParameterTranslation( ) メソッドをコールして、すべての HTML 形式のパラメータ名およびすべてのテキストベースの HTML 形式のパラメータ値を変換する場合に使用するキャラクタ・エンコーディングを指定できます。

注意：

- multipart/form-data の POST 要求に付属する問合せ文字列のパラメータは、multipart/form-data パラメータのリストにマージされる前には変換されません。これは、基盤となる Servlet コンテナまたは JSP エンジンが、問合せ文字列をデコードしたかどうか、あるいはパラメータ名または値を変換したかどうかを判断する方法がないためです。そのため、基盤となる Servlet コンテナや JSP エンジンが変換を実行しなかった場合、アプリケーションで、すべてのマルチバイトの問合せ文字列のパラメータ名または値を変換する必要があります。
- アップロード・ファイルのコンテンツおよび関連付けされたコンテンツ・タイプ属性は変換されません。これらは、常に ISO 8859-1 キャラクタ・エンコーディングを使用して表現されます。ただし、アップロード・ファイルのファイル名属性は変換されます。
- GET 要求の問合せ文字列のパラメータ、問合せ文字列、および application/x-www-form-urlencoded の POST 要求の POST データのパラメータは、変換されません。
- HTML 形式のパラメータ名および値の変換を正しく処理するには、Servlet コンテナまたは JSP エンジンが、GET 要求および application/x-www-form-urlencoded の POST 要求に対するパラメータ名および値を変換しても、アプリケーションで、multipart/form-data の POST 要求に対して、enableParameterTranslation( ) メソッドをコールする必要があります。
- アプリケーションがパラメータ名および値を変換するコードを含む場合、enableParameterTranslation( ) メソッドはコールしないでください。
- enableParameterTranslation( ) メソッドは、parseFormData( ) メソッドのコールを行う前にコールする必要があります。

ISO 8859-1 以外のキャラクタ・エンコーディングを使用して、multipart/form-data の POST 要求に対して enableParameterTranslation( ) メソッドをコールすると、次のメソッドに影響があります。

- getParameter( ): パラメータ名の引数および戻されるパラメータ値
- getParameterValues( ): パラメータ名の引数および戻されるパラメータ値
- getParameterNames( ): 戻されるパラメータ名
- getFileParameter( ): パラメータ名の引数のみ
- getFileParameterValues( ): パラメータ名の引数のみ
- getFileParameterNames( ): 戻されるパラメータ名

GET 要求および application/x-www-form-urlencoded の POST 要求の場合、getParameter( )、getParameterValues( ) および getParameterNames( ) メソッドをコールすると、基盤となる Servlet コンテナまたは JSP エンジンに直接渡されます。Servlet コンテナまたは JSP エンジンがサポートするパラメータ変換機能については、Servlet コンテナまたは JSP エンジンのドキュメントを参照してください。

パラメータ

encoding
キャラクタ・エンコーディングを指定する文字列

戻り値

なし

例外

なし

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.enableParameterTranslation("GB2312")
formData.parseFormData( );
```

getFileParameter( )

構文

```
public OrdHttpUploadFile getFileParameter(String parameterName)
```

説明

パラメータ名で識別されるアップロード・ファイルに関する情報を、OrdHttpUploadFile オブジェクトで戻します。

フィールドに有効なファイル名が入力されたかどうかにかかわらず、HTML 形式の FILE 型のすべての入力フィールドは、OrdHttpUploadFile のパラメータを生成します。

パラメータ

parameterName
アップロード・ファイルのパラメータ名を指定する文字列

戻り値

アップロード・ファイルのパラメータを、OrdHttpUploadFile オブジェクトで戻します。パラメータが存在しない場合は、null を戻します。

例外

java.lang.IllegalStateException

ServletRequest オブジェクトが指定されていない場合、multipart フォーム・データが解析されていない場合またはアップロード要求が解放されている場合は、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
...
OrdHttpUploadFile photo = formData.getFileParameter("photo");
photo.loadImage(media);
...
formData.release( );
```

getFileParameterNames( )

構文

```
public java.util.Enumeration getFileParameterNames( )
```

説明

HTML 形式の FILE 型のすべての入力フィールド名の列挙、または FILE 型の入力フィールドがない場合は、空の列挙を戻します。

フィールドに有効なファイル名が入力されたかどうかにかかわらず、HTML 形式の FILE 型のすべての入力フィールドは、OrdHttpUploadFile のパラメータを生成します。

パラメータ

なし

戻り値

アップロード・ファイルのパラメータ名を、文字列の列挙で戻します。

例外

java.lang.IllegalStateException

ServletRequest オブジェクトが指定されていない場合、multipart フォーム・データが解析されていない場合またはアップロード要求が解放されている場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData( request );
formData.parseFormData( );
...
Enumeration names = formData.getFileParameterNames( );
```

getFileParameterValues( )

構文

```
public OrdHttpUploadFile[ ] getFileParameterValues
  (String parameterName)
```

説明

指定されたパラメータ名を使用してアップロードされたすべてのファイルを表す、OrdHttpUploadFile オブジェクトの配列を戻します。フィールドに有効なファイル名が入力されたかどうかにかかわらず、HTML 形式の FILE 型のすべての入力フィールドは、OrdHttpUploadFile のパラメータを生成します。

パラメータ

parameterName
アップロード・ファイルのパラメータ名を指定する文字列

戻り値

アップロード・ファイルのパラメータを、OrdHttpUploadFile オブジェクトの配列で戻します。パラメータが存在しない場合は、null を戻します。

例外

java.lang.IllegalStateException

ServletRequest オブジェクトが指定されていない場合、multipart フォーム・データが解析されていない場合またはアップロード要求が解放されている場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData( request );
formData.parseFormData( );
...
OrdHttpUploadFile[ ] photo = formData.getFileParameterValues("photo")
```

getParameter( )

構文

```
public String getParameter(String parameterName)
```

説明

1 番目の問合せ文字列のパラメータ、または指定された名前を使用したテキストベースのフォームのフィールド・パラメータの値を戻します。パラメータが存在しない場合は null を戻します。要求の問合せ文字列のパラメータは、parseFormData( ) メソッドによって、テキストベースのフォームのフィールド・パラメータにマージされます。

このメソッドは、要求が multipart/form-data のアップロード要求ではない場合に、ServletRequest クラスの getParameterName( ) メソッドをコールします。

パラメータ

parameterName
パラメータ名を指定する文字列

戻り値

指定されたパラメータの値を、文字列で戻します。パラメータが存在しない場合は、null を戻します。

例外

java.lang.IllegalStateException

ServletRequest オブジェクトが指定されていない場合、multipart フォーム・データが解析されていない場合またはアップロード要求が解放されている場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData( request );
formData.parseFormData( );
...
String id = formData.getParameter("id");
```

getParameterNames( )

構文

```
public java.util.Enumeration getParameterNames( )
```

説明

すべての問合せ文字列のパラメータ名の列挙、および要求内のすべてのテキストベース・フォームのフィールド・データ・パラメータ名を戻します。問合せ文字列パラメータおよびテキストベース・フォームのフィールド・パラメータが存在しない場合は、空の列挙を戻します。要求の問合せ文字列のパラメータは、parseFormData( ) メソッドによって、テキストベースのフォームのフィールド・パラメータにマージされます。

このメソッドは、要求が multipart/form-data のアップロード要求ではない場合、ServletRequest クラスの getParameterNames( ) メソッドをコールします。

パラメータ

なし

戻り値

パラメータ名のリストを、文字列の列挙で戻します。

例外

java.lang.IllegalStateException

ServletRequest オブジェクトが指定されていない場合、multipart フォーム・データが解析されていない場合またはアップロード要求が解放されている場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData( request );
formData.parseFormData( );
...
Enumeration names = formData.getParameterNames( );
```

getParameterValues( )

構文

```
public String[ ] getParameterValues(String parameterName)
```

説明

すべての問合せ文字列のパラメータおよび指定されたパラメータ名を使用したテキストベース・フォームのフィールド・データ・パラメータ値を含む String オブジェクトの配列を戻します。パラメータが存在しない場合は null を戻します。要求の問合せ文字列のパラメータは、parseFormData( ) メソッドによって、テキストベース・フォームのフィールド・パラメータにマージされます。

このメソッドは、要求が multipart/form-data のアップロード要求ではない場合に、ServletRequest クラスの getParameterNames( ) メソッドをコールします。

パラメータ

parameterName
パラメータ名を指定する文字列

戻り値

パラメータ値を、文字列で戻します。パラメータが存在しない場合は、null を戻します。

例外

java.lang.IllegalStateException

ServletRequest オブジェクトが指定されていない場合、multipart フォーム・データが解析されていない場合またはアップロード要求が解放されている場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
formData.parseFormData( );
...
String[ ] ids = formData.getParameterValues("id");
```

isUploadRequest( )

構文

```
public boolean isUploadRequest( )
```

説明

要求が multipart/form-data エンコーディング・フォーマットを使用してエンコードされているかどうかをテストします。

パラメータ

なし

戻り値

Request Body が multipart/form-data エンコーディング・フォーマットを使用してエンコードされている場合は true を戻し、そうでない場合は false を戻します。

例外

java.lang.IllegalStateException

HttpServletRequest が指定されていない場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
if(formData.isUploadRequest( )){
     formData.parseFormData( );
     OrdHttpUploadFile uploadFile = formData.getFileParameter(...);
     ...
}
else{
     String param = request.getParameter(...);
     ...
}
```

OrdHttpUploadFormData( )

構文

```
public OrdHttpUploadFormData( )
```

説明

multipart/form-data 要求を解析する OrdHttpFormData オブジェクトを作成します。その後、アプリケーションで、ServletRequest オブジェクトを指定する必要があります。

パラメータ

なし

戻り値

なし

例外

なし

例

このメソッドの例は、「setServletRequest( )」を参照してください。

OrdHttpUploadFormData(ServletRequest)

構文

```
public OrdHttpUploadFormData(javax.servlet.ServletRequest request)
```

説明

multipart/form-data 要求を解析する OrdHttpUploadFormData オブジェクトを作成します。

パラメータ

request
ServletRequest 型のオブジェクト

戻り値

なし

例外

なし

例

このメソッドの例は、「getFileParameter( )」を参照してください。

parseFormData( )

構文

```
public void parseFormData( )
```

説明

multipart/form-data エンコーディングを使用してエンコードされた POST 要求の body を解析します。要求がアップロード要求ではない場合、このメソッドは機能しません。

パラメータ

なし

戻り値

なし

例外

java.lang.IllegalStateException

HttpServletRequest が指定されていない場合、この例外がスローされます。

java.io.IOException

Request Body の読込みまたは一時ファイルの書込み中にエラーが発生した場合、この例外がスローされます。

OrdHttpUploadException

multipart/form-data メッセージの解析中にエラーが発生した場合、この例外がスローされます。

例

このメソッドの例は、「getFileParameter( )」を参照してください。

release( )

構文

```
public void release( )
```

説明

アップロード・ファイルのコンテンツを保持するために使用した一時ファイルなど、OrdHttpUploadFormData オブジェクトが保持するすべてのリソースを解放します。一時ファイルの使用を可能にするアプリケーションでは、このメソッドをコールする必要があります。

パラメータ

なし

戻り値

なし

例外

なし

例

このメソッドの例は、「getFileParameter( )」を参照してください。

setMaxMemory( )

構文

```
public void setMaxMemory(int maxMemory, String tempFileDir)
```

説明

コンテンツが一時ディレクトリに格納される前に、アップロード・ファイルのコンテンツが消費可能な最大メモリー量を指定します。

デフォルトでは、アップロード・ファイルのコンテンツは、アプリケーションによってデータベースに格納されるまでメモリー内に保持されます。ビデオ・クリップなどのサイズの大きいファイルをアップロードするときは、コンテンツをデータベースに書き込む前に、メモリー消費量を制限し、ファイルのコンテンツを一時的にディスクに格納する方がよい場合があります。

注意： このメカニズムを使用するアプリケーションは、release( ) メソッドを使用して、不要になったすべての一時ファイルが削除されるようにする必要があります。詳細は、「release( )」メソッドを参照してください。

パラメータ

maxMemory
アップロード・ファイルのコンテンツが一時ファイルに格納される前に、要求内のすべてのアップロード・ファイルによって消費される最大メモリー量を指定する整数

tempFileDir
一時ファイルを格納する一時ファイル・ディレクトリを指定する文字列。java.io.tmpdir システム・プロパティが存在している場合、このパラメータの設定はオプションとなります。

戻り値

なし

例外

java.lang.IllegalArgumentException

maxMemory パラメータが負であるか、または tempFileDir パラメータに null が指定され、java.io.tmpdir システム・プロパティが存在していない場合、この例外がスローされます。

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData(request);
try{
     formData.setMaxMemory(65536,null);
     formData.parseFormData( );
     ...
     OrdHttpUploadFile photo = formData.getFileParameter("photo");
     photo.loadImage(media);
     ...
}
finally{
     formData.release( );
}
```

setServletRequest( )

構文

```
public void setServletRequest(javax.servlet.ServletRequest request)
```

説明

要求に対する ServletRequest オブジェクトを指定します。ServletRequest オブジェクトは、要求の解析前に、コンストラクタ内で指定するか、またはメソッドをコールして指定する必要があります。

パラメータ

request
ServletRequest 型のオブジェクト

戻り値

なし

例外

なし

例

```
OrdHttpUploadFormData formData = new OrdHttpUploadFormData( );
...
formData.setServletRequest(request);
```

A

Java Classes サンプル・プログラム

*inter*Media Java Classes のインストールによって、4 つのサンプル・ファイル（Java で記述されたプログラム）が提供されます。これらのファイルには、*inter*Media を使用して Java アプリケーションを構築する方法の例が記載されています。それぞれのファイルでは、様々なソースからデータベース・オブジェクトにデータをロードする方法、データベース・オブジェクトからファイル・システムにデータをダウンロードする方法、およびメディア・コンテンツからメタデータを抽出し表示する方法を示します。

Java サンプル・ファイルの名前は、次のとおりです。

- オーディオ用 : AudioExample.java
- ドキュメント用 : DocumentExample.java
- イメージ用 : ImageExample.java
- ビデオ用 : VideoExample.java

*inter*Media Java Classes のインストールによって、2 つの追加 Java サンプル・ファイルが提供されます。これらのサンプル・ファイルでは、*inter*Media Java Classes for servlets and JSP を使用して、メディア・データをアップロードし、検索する方法を示します。

追加 Java サンプル・ファイルの名前は、次のとおりです。

- Oracle *inter*Media JavaServer Pages Photo Album Demo
- Oracle *inter*Media Java Servlet Photo Album Demo

これらのサンプル・ファイルが含まれている README ファイルの検索方法については、A.1 項「サンプル・ファイルの検索」を参照してください。

A.1 サンプル・ファイルの検索

*inter*Media Java Classes 付属の Java サンプル・ファイルおよびその実行手順については、Oracle *inter*Media のインストール後、Oracle ホーム・ディレクトリ内の該当する README ファイルで参照できます。

- オーディオ用 :

```
<ORACLE_HOME>/ord/aud/demo/java/README.txt (on UNIX)
<ORACLE_HOME>¥ord¥aud¥demo¥java¥README.txt (on Windows NT)
```

- ドキュメント用 :

```
<ORACLE_HOME>/ord/doc/demo/java/README.txt (on UNIX)
<ORACLE_HOME>¥ord¥doc¥demo¥java¥README.txt (on Windows NT)
```

- イメージ用 :

```
<ORACLE_HOME>/ord/img/demo/java/README.txt (on UNIX)
<ORACLE_HOME>¥ord¥img¥demo¥java¥README.txt (on Windows NT)
```

- ビデオ用 :

```
<ORACLE_HOME>/ord/vid/demo/java/README.txt (on UNIX)
<ORACLE_HOME>¥ord¥vid\demo¥java¥README.txt (on Windows NT)
```

- *inter*Media JavaServer Pages Photo Album Demo 用 :

```
<ORACLE_HOME>/ord/http/demo/jsp/README.txt (on UNIX)
<ORACLE_HOME>¥ord¥http¥demo¥jsp¥README.txt (on Windows NT)
```

- *inter*Media Java Servlet Photo Album Demo 用 :

```
<ORACLE_HOME>/ord/http/demo/servlet/README.txt (on UNIX)
<ORACLE_HOME>¥ord¥http¥demo¥servlet¥README.txt (on Windows NT)
```

A.2 サンプル・ファイルを実行する前の準備

*inter*Media Java Classes に付属の Java サンプル・ファイルを実行するには、次の操作を行う必要があります。

1. Oracle9*i* を、Oracle *inter*Media とともにインストールします。

   Oracle9*i* データベース・サーバーのサーバー・マシンに Oracle *inter*Media がインストールされている必要があります。

2. Java 環境が適切であること、および Java プログラムをコンパイルおよび実行できるように設定されていることを確認してください。

3. サンプル・ファイルの実行の詳細は、A.1 項「サンプル・ファイルの検索」に示されている該当 README ファイルを参照してください。

B

例外およびエラー

この付録では、*inter*Media Java Classes によって発生する可能性のある例外およびエラーについて説明します。

B.1 Exception クラス

Exception クラス（および SQLException を含むサブクラス）は、ユーザーに関連する状態を示します。

```
public class java.lang.Exception extends java.lang.Throwable {
     //Constructs an Exception with no detailed message
     public Exception();

     //Constructs an Exception with a detailed message
     public Exception(String s);
}
```

B.2 IllegalArgumentException クラス

IllegalArgumentException クラスは、メソッドが無効または不適切な引数を渡したことを示します。

```
public class IllegalArgumentException extends RuntimeException{
     //Constructs an IllegalArgumentException with no detailed message
     public IllegalAgumentException( )
     //Constructs an IllegalArgumentExceptuon with the specified detailed
     //message
     public IllegalArgumentException(String s)
}
```

B.3 IllegalStateException クラス

IllegalStateException クラスは、メソッドが無効または不適切なタイミングでコールされたことを示します。Java 環境またはアプリケーションが要求された操作に不適切な状態にあります。

```
public class IllegalStateException extends java.lang.RuntimeException {
     //Constructs an IllegalStateException with no detailed message
     public IllegalStateException();

     //Constructs an IllegalStateException with the specified detailed message
     public IllegalStateException(String s);
}
```

B.4 IOException クラス

IOException クラスは、なんらかの I/O 例外が発生したことを示します。

```
public class java.io.IOException extends java.lang.Exception {
     //Constructs an IOException with no detailed message
     public IOException();

     //Constructs an IOException with the specified detailed message
     public IOException(String s);
}
```

B.5 OutOfMemoryError クラス

OutOfMemoryError クラスは、メモリーが不足し、ガベージ・コレクション（使用されなくなったオブジェクトの削除）を行っても使用可能なメモリーを増やすことができないため、JVM でオブジェクトを割り当てることができないことを示します。

```
public class java.lang.OutOfMemoryError extends java.lang.VirtualMachineError {
     //Constructs an OutOfMemoryError with no detailed message
     public OutOfMemoryError();

     //Constructs an OutOfMemoryError with a detailed message
     public OutOfMemoryError(string s);
}
```

B.6 OrdHttpResponseException クラス

oracle.ord.im.OrdHttpResponse クラスは、ServletException を継承したもので、メディア・データの検索中およびデータベースから HTTP クライアントへの配信中にエラーが発生したことを示します。

B.7 OrdHttpUploadException クラス

oracle.ord.im.OrdHttpUploadException クラスは、IOException を継承したもので、HTTP クライアントからデータベースへのメディア・データのアップロード中にエラーが発生したことを示します。このクラスの主な目的は、エラー・メッセージ・テキストのローカリゼーションを可能にすることです。

B.8 ServletException クラス

ServletException クラスは、障害が発生したときにサーブレットが生成する一般的な例外を定義します。

```
public class ServletException extends java.lang.Exception{
     //Constructs a new ServletException
     public ServletException( )

     //Constructs a new ServletException with the specified message
     public ServletException(String message)

     //Constructs a new ServletException with the specified message and
     //the "root cause" exception that interfered with the normal operation
     //of the servlet
     public ServletException(String message, java.lang.Throwable rootCause)

     //Constructs a new ServletException with the "root cause" exception that
     //interfered with the normal operation of the servlet
     public ServletException(java.lang.Throwable rootCause)

     //Gets the exception that caused the ServletException
     public java.lang.Throwable getRootCause( )
}
```

B.9 SQLException クラス

SQLException クラスは、データベース・アクセス・エラーについての情報を提供します。

```
public class java.sql.SQLException extends java.lang.Exception {
     //The following four methods are public constructors:

     //Constructs a fully specified SQLException
     public SQLException(String reason, String SQLState, int vendorCode);

     //Constructs a SQLException with vendorCode value of 0
     public SQLException(String reason, String SQLState);

     //Constructs a SQLException with vendorCode value of 0 and a null SQLState
     public SQLException(String reason);

     //Constructs a SQLException with vendorCode of 0, a null SQLState. and a
     //null reason
     public SQLException();

     //The following four methods are public instance methods:
```

```
    //Gets the vendor-specific exception code
    public int getErrorCode();

    //Gets the exception connected to this one
    public SQLException getNextException();

    //Gets the SQL state
    public String getSQLState();

    //Adds a SQLException to the end
    public synchronized void setNextException(SQLException ex);
}
```

C

使用されなくなったメソッド

次に、Oracle *inter*Media Audio、Image および Video Java Client リリース 8.1.5 から使用されなくなったメソッドを示します。

- public void appendToComments(int amount, String buffer)
- public int compareComments(CLOB dest, int amount, int start_in_comment, int start_in_compare_comment)
- public CLOB copyCommentsOut(CLOB dest, int amount, int from_loc, int to_loc)
- public void deleteComments( )
- public int eraseFromComments(int amount, int offset)
- public void flush( ) throws SQLException
- public String getAllAttributesAsString(byte[ ] ctx)
- public int getAudioDuration(byte[ ] ctx)
- public int getBitRate(byte[ ] ctx)
- public int getCommentLength( )
- public String getCommentsAsString( )
- public String getCompressionType(byte[ ] ctx)
- public byte[ ] getData(String tableName, String columnName, String condition)
- public String getEncoding(byte[ ] ctx)
- public String getFormat(byte[ ] ctx)
- public int getFrameRate(byte[ ] ctx)
- public int getFrameResolution(byte[ ] ctx)
- public FrameDimension getFrameSize(byte[ ] ctx)

- public int getNumberOfChannels(byte[ ] ctx)
- public int getNumberOfColors(byte[ ] ctx)
- public int getNumberOfFrames(byte[ ] ctx)
- public int getSampleSize(byte[ ] ctx)
- public int getSamplingRate(byte[ ] ctx)
- public int getVideoDuration(byte[ ] ctx)
- public boolean loadComments(String filename)
- public void loadCommentsFromFile(String loc, String fileName, int amount, int from_loc, int to_loc)
- public boolean loadData(String fileName, String tableName, String columnName, String condition)
- public int locateInComment(String pattern, int offset, int occurrence)
- public OrdAudio(Connection the_connection)
- public OrdVideo(Connection the_connection)
- public String readFromComments(int offset, int amount)
- public void refresh(boolean forUpdate)
- public void setBindParams(String tableName, String columnName, String condition)
- public void trimComments(int newlen)
- public void writeToComments(int offset, int amount, String buffer)

索引

I

L

M

O

P

R

S

T

W

あ

お

か

こ

て

ひ

ふ

め

れ