Oracle® *inter*Media

Java Classes Reference

10*g* Release 1 (10.1)

Part No. B10830-01

December 2003

Oracle *inter*Media Java Classes is a component of Oracle *inter*Media, a product designed to manage multimedia Web content within Oracle Database.

ORACLE®

Oracle *inter*Media Java Classes Reference, 10*g* Release 1 (10.1)

Part No. B10830-01

Primary Author: Sue Pelski

Contributors: Fengting Chen, Susan Mavris, Susan Shepard, Manjari Yalavarthy

Contents

Send Us Your Comments

Oracle *inter*Media Java Classes Reference, 10*g* Release 1 (10.1)

Part No. B10830-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: nedc-doc_us@oracle.com
- FAX: 603.897.3825 Attn: Oracle *inter*Media Documentation
- Postal service:
  Oracle Corporation
  Oracle *inter*Media Documentation
  One Oracle Drive
  Nashua, NH 03062-2804
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This guide describes how to use Oracle *inter*Media Java Classes ("*inter*Media Java Classes").

Audience

This guide is for developers or database administrators who are interested in storing, retrieving, and manipulating media data in Oracle Database, including developers of multimedia specialization applications. Users of this guide should have experience with SQL, PL/SQL, Java, Java Database Connectivity (JDBC), and *inter*Media.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community.

To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an

otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Organization

This guide contains the following chapters and an appendix:

Chapter 1, "Introduction to Oracle interMedia Java Classes"
Contains a general introduction to Oracle *inter*Media Java Classes.

Chapter 2, "OrdAudio"
Contains reference information on the OrdAudio class.

Chapter 3, "OrdDoc"
Contains reference information on the OrdDoc class.

Chapter 4, "OrdImage"
Contains reference information on the OrdImage class.

Chapter 5, "OrdImageSignature"
Contains reference information on the OrdImageSignature class.

Chapter 6, "OrdMediaUtil"
Contains reference information on the OrdMediaUtil class.

Chapter 7, "OrdVideo"
Contains reference information on the OrdVideo class.

Chapter 8, "JAI Input and Output Stream"
Contains reference information on input and output stream objects designed to work with Java Advanced Imaging.

Chapter 9, "Java Classes for Servlets and JSP"

Contains reference information on Java classes for servlets and JavaServer Pages.

Appendix A, "Deprecated Methods"

Contains information on methods that have been deprecated.

Related Documentation

This guide is not intended as a standalone document. It is a supplement to *Oracle interMedia User's Guide* and *Oracle interMedia Reference*. You need all three guides to successfully perform operations on *inter*Media objects using the Java interface.

For more information on using *inter*Media in a development environment, see the following documents in the Oracle documentation set:

- *Oracle Call Interface Programmer's Guide*
- *Oracle Database Application Developer's Guide - Fundamentals*
- *Oracle Database Concepts*
- *PL/SQL User's Guide and Reference*

For more information on using JDBC, see *Oracle Database JDBC Developer's Guide and Reference*.

For reference information on both Oracle *inter*Media Java Classes and Oracle *inter*Media Java Classes for Servlets and JSP in Javadoc format, see the Oracle API documentation (also known as Javadoc). The API documentation is available on the Oracle Database 10*g* Documentation Library CD-ROM and also from the documentation section of the Oracle Technology Network (OTN) Web site at

`http://otn.oracle.com/documentation`

For more information on Java, see the API documentation provided by Sun Microsystems at

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

For more information on the Java Advanced Imaging (JAI) API, see the following Web site (which is maintained by Sun Microsystems)

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

For information added after the release of this guide, refer to the online `README.txt` file in your Oracle home directory. Depending on your operating system, this file may be in the following location:

<ORACLE_HOME>/ord/im/admin/README.txt

See your operating system-specific installation guide for more information.

Printed documentation is available for sale in the Oracle Store at

http://oraclestore.oracle.com/

To download free release notes, installation documentation, white papers, or other collateral, go to the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

http://otn.oracle.com/admin/account/membership.html

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

http://otn.oracle.com/documentation

Conventions

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The java.lang.String object is sometimes abbreviated as String.

Although Boolean is a proper noun, it is presented as boolean in this guide when its use in Java code requires case-sensitivity.

The following conventions are also used in this guide:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
boldface text	Boldface text indicates a term defined in the text and glossary.
italic text	Italic text is used for emphasis, book titles, and user-supplied information.
`monospace text`	Monospace text is used in examples. It is also used in text for file and directory names, absolute path names, URL names, and for excerpts of examples.

Convention	Meaning
< >	Angle brackets enclose user-supplied names.
{ }	Braces in syntax or examples enclose two or more items, one of which is required.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1

Introduction to Oracle *inter*Media Java Classes

Oracle *inter*Media ("*inter*Media") provides Java Classes to enable users to write Java applications using *inter*Media objects. The OrdAudio, OrdDoc, OrdImage, and OrdVideo classes are used to represent instances of their corresponding database object types in a Java application. For more information about database object types, see *Oracle interMedia Reference* and *Oracle interMedia User's Guide*.

The material in this guide assumes prior knowledge of SQL, PL/SQL, Java, Java Database Connectivity (JDBC), and *inter*Media. For a list of related documentation, see the Preface in this guide.

1.1 Java Application Support

Oracle *inter*Media lets you store your multimedia information in a database table, retrieve it from the table, and manipulate it. Oracle *inter*Media Java Classes ("*inter*Media Java Classes") lets you write your own Java applications to use, manipulate, and modify media data stored in Oracle Database.

*inter*Media Java Classes lets an application retrieve an object from a result set and manipulate the contents of the object.

The client library, `ordim.jar`, is located in *`$ORACLE_HOME`*`/ord/im/jlib` on UNIX and *`<ORACLE_HOME>`*`\ord\im\jlib` on Windows.

1.2 Oracle *inter*Media Java Classes Sample Programs

Sample programs written in Java are provided in the installation of *inter*Media Java Classes. These sample programs provide examples of how to build Java

applications with *inter*Media. See *Oracle interMedia User's Guide* for information about finding and running these sample programs.

2

OrdAudio

This chapter contains reference information for the oracle.ord.im.OrdAudio class.

The OrdAudio class is used to represent an instance of the ORDSYS.ORDAudio database type in a Java application. The OrdAudio class includes a set of methods to get and set various object attributes, as well as a set of methods to perform various operations on an OrdAudio Java object.

Almost all methods operate on the attributes of the OrdAudio Java object in the application. The exceptions are those methods that access the audio data for read or write purposes, which are described in the following list:

- Methods that operate on the database BLOB specified by the localData attribute, read and write data stored in the database BLOB.
- Methods that operate on the database BFILE specified by the srcLocation and srcName attributes when the srcType attribute is "file," read data from the specified file.
- Methods that operate on the URL specified by the srcType, srcLocation, and srcName attributes when the srcType attribute is "http," read data from the resource at the specified URL.

If your application modifies the OrdAudio Java object or the audio data in the database, you must update the ORDAudio SQL object in the database to make those changes permanent.

Some methods in the OrdAudio Java class are handed off to a database source plug-in or database format plug-in for processing; these methods have `byte [ ] [ ] ctx` as a context parameter. Applications should allocate a 64-byte array to hold any context information that may be required by a source plug-in or a format plug-in. For example, a plug-in may initialize the context information in one call and use that information in a subsequent call. The source plug-in context requires one array; the format plug-in context requires another array. For most plug-ins, 64

bytes should be sufficient. Some user-defined plug-ins may need additional space. The following example shows how to allocate a plug-in context information array:

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

> **Note:** In the current release, no Oracle-supplied source plug-ins or format plug-ins maintain context. Also, not all user-written source plug-ins or format plug-ins maintain context. However, if you include the context parameter as described, your application should work with any current or future source plug-ins or format plug-ins.

See *Oracle interMedia Reference* for more information about plug-ins.

2.1 Prerequisites

In order to run OrdAudio methods, you will need to include the following import statements in your Java file:

```
import oracle.ord.im.OrdMediaUtil;
import oracle.ord.im.OrdAudio;
```

You may also need to import classes from the following Java packages:

```
java.io.
java.sql.
oracle.jdbc.
```

Before running OrdAudio methods, the following operations must have already been performed:

- A connection has been made to a table that contains a column of type OrdAudio.
- A local OrdAudio object has been created and populated with data.

For examples of making a connection and populating a local object, see *Oracle interMedia User's Guide*.

2.2 Reference Information

This section presents reference information on the methods that operate on OrdAudio objects.

checkProperties(byte[ ] [ ])

Format

public boolean checkProperties(byte[ ] [ ] ctx)

Description

Checks if the properties of the audio data are consistent with the attributes of the OrdAudio Java object.

Parameters

ctx
The format plug-in context information.

Return Value

This method returns true if the properties of the audio data are consistent with the attributes of the OrdAudio Java object; false otherwise.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding checkProperties( ) method in the database.

Examples

None.

clearLocal( )

Format

```
public void clearLocal( )
```

Description

Clears the local attribute to indicate that the audio data is stored externally.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the local attribute.

Examples

None.

closeSource(byte[ ] [ ])

Format

public int closeSource(byte[ ] [ ] ctx)

Description

Closes a data source.

Parameters

ctx
The source plug-in context information.

Return Value

This method returns the status as an integer, where zero indicates success and a non-zero value indicates a failure code specific to the source plug-in.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding closeSource( ) method in the database.

Examples

None.

deleteContent( )

Format

public void deleteContent( )

Description

Deletes any data stored in the database BLOB specified by the localData attribute.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding deleteContent( ) method in the database.

Examples

None.

export(byte[ ] [ ],String, String, String)

Format

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

Description

Exports the data from the BLOB specified by the localData attribute. This method calls the corresponding export( ) method in the database to export the audio data to a location specified by the srcType, srcLocation, and srcName parameters.

Not all source plug-ins support this method. Only the "file" source type is natively supported.

This method will work only if you are running Oracle release 8.1.7 or later.

The remainder of this description describes the use of the export( ) method and the Oracle-supplied "file" source plug-in. User-written plug-ins will behave differently.

The export( ) method implemented by the "file" source plug-in copies, but does not modify, the audio data stored in the database BLOB specified by the localData attribute.

After exporting the audio data, all the audio property attributes remain unchanged. However, the srcType, srcLocation, and srcName attributes are updated with values of the srcType, srcLocation, and srcName parameters passed to the export( ) method. After calling the export( ) method, if you no longer intend to manage the audio data within the database, call the clearLocal( ) method to indicate that the audio data is stored outside the database, and call the deleteContent( ) method to delete the audio data stored in the database BLOB.

See *Oracle interMedia Reference* for information about the privileges required to write to a database directory object. See *Oracle Database Java Developer's Guide* and the java.io.FilePermission class in the Java API for information about security and performance.

Parameters

ctx
The source plug-in context information.

srcType
The source type to which the content will be exported.

srcLocation
The source location to which the content will be exported.

srcName
The source name to which the content will be exported.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding export( ) method in the database.

Examples

None.

getAllAttributes(byte[ ] [ ])

Format

public CLOB getAllAttributes(byte[ ] [ ] ctx)

Description

Returns the values of the audio properties in a temporary CLOB in a form defined by the format plug-in. For natively supported formats, the information is presented as a comma-separated list of attributes. The list of attributes is of the form attributeName=attributeValue, where the list contains the following attributes: format, mimeType, encoding, numberOfChannels, samplingRate, sampleSize, compressionType, and audioDuration. For user-defined formats, the information is presented in a form defined by the format plug-in.

> **Note:** The application must free the temporary CLOB after reading the information it contains.

Parameters

ctx
The format plug-in context information.

Return Value

This method returns the values of the attributes as a temporary CLOB, oracle.sql.CLOB.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getAllAttributes( ) method in the database.

Examples

None.

getAttribute(byte[ ] [ ], String)

Format

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

Description

Returns the value of the requested audio property. This method is used by user-defined format plug-ins to return the value of an audio property that is not available as an attribute of the OrdAudio Java object. This method is not implemented by any Oracle-supplied format plug-ins.

Parameters

ctx
The format plug-in context information.

name
The property or attribute name.

Return Value

This method returns the value of the attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getAttribute( ) method in the database.

Examples

None.

getAudioDuration( )

Format

public int getAudioDuration( )

Description

Returns the value of the audioDuration attribute.

Parameters

None.

Return Value

This method returns the value of the audioDuration attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the audioDuration attribute.

Examples

None.

getBFILE( )

Format

public oracle.sql.BFILE getBFILE( )

Description

Returns a BFILE locator from the database when the srcType attribute is "file." This method calls the corresponding getBFILE( ) method in the database, which creates the BFILE using the srcLocation and srcName attributes.

Parameters

None.

Return Value

This method returns an oracle.sql.BFILE locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getBFILE( ) method in the database.

Examples

None.

getComments( )

Format

public oracle.sql.CLOB getComments( )

Description

Returns the CLOB locator from the comments attribute.

Parameters

None.

Return Value

This method returns the value of the comments attribute as an oracle.sql.CLOB locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the comments attribute.

Examples

None.

getCompressionType( )

Format

public String getCompressionType( )

Description

Returns the value of the compressionType attribute.

Parameters

None.

Return Value

This method returns the value of the compressionType attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the compressionType attribute.

Examples

None.

getContent( )

Format

public oracle.sql.BLOB getContent( )

Description

Returns the BLOB locator from the localData attribute.

Parameters

None.

Return Value

This method returns an oracle.sql.BLOB locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

getContentInLob(byte[ ] [ ], String, String)

Format

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

Description

Returns the data from the BLOB specified by the localData attribute in a temporary BLOB in the database. This method creates a temporary BLOB in the database, reads the data from the BLOB specified by the localData attribute, writes the data to the temporary BLOB, then returns the temporary BLOB locator to the caller.

> **Note:** The application must free the temporary BLOB after accessing the data it contains.

Parameters

ctx
The format plug-in context information.

mimetype
A String array, 1 element in length, into which the mimeType attribute is written as element 0.

format
A String array, 1 element in length, into which the format attribute is written as element 0.

Return Value

This method returns the audio data in a temporary oracle.sql.BLOB locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs creating the temporary BLOB or executing the corresponding getContentInLob( ) method in the database.

Examples

None.

getContentLength( )

Format

public int getContentLength( )

Description

Returns the length of the audio data. This method calls the corresponding getContentLength( ) method in the database.

This method is not supported for all source types. For example, the "http" source type does not support this method.

Parameters

None.

Return Value

This method returns the value of the contentLength attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getContentLength( ) method in the database.

Examples

None.

getContentLength(byte[ ][ ])

Format

public int getContentLength(byte[ ] [ ] ctx)

Description

Returns the length of the audio data using source plug-in context information. This method calls the corresponding getContentLength( ) method in the database.

This method is not supported for all source types. For example, the "http" source type does not support this method.

Parameters

ctx
The source plug-in context information.

Return Value

This method returns the value of the contentLength attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getContentLength( ) method in the database.

Examples

None.

getDataInByteArray( )

Format

public byte[ ] getDataInByteArray( )

Description

Returns a byte array containing the data from the database BLOB specified by the localData attribute.

Parameters

None.

Return Value

This method returns the byte array containing the data.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

java.io.IOException

This exception is thrown if an error occurs reading the data from the BLOB.

java.lang.OutOfMemoryError

This exception is thrown if sufficient memory cannot be allocated to hold the data.

Examples

None.

getDataInFile(String)

Format

public boolean getDataInFile(String filename)

Description

Writes the data from the database BLOB specified by the localData attribute to a local file.

Parameters

filename
The name of the file to which the data will be written.

Return Value

This method returns true if the data is written to the file successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

java.io.IOException

This exception is thrown if an error occurs reading the data from the BLOB or writing the data to the output file.

Examples

None.

getDataInStream( )

Format

public InputStream getDataInStream( )

Description

Returns an InputStream object from which the data in the database BLOB specified by the localData attribute can be read.

Parameters

None.

Return Value

This method returns an InputStream object from which the data will be read.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

Examples

None.

getDescription( )

Format

public String getDescription( )

Description

Returns the value of the description attribute.

Parameters

None.

Return Value

This method returns the value of the description attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the description attribute.

Examples

None.

getEncoding( )

Format

public String getEncoding( )

Description

Returns the value of the encoding attribute.

Parameters

None.

Return Value

This method returns the value of the encoding attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the encoding attribute.

Examples

None.

getORADataFactory( )

Format

public static oracle.sql.ORADataFactory getORADataFactory( )

Description

Returns the OrdAudio ORADataFactory interface for use by the getORAData( ) method. Specify the getORADataFactory( ) method as the factory parameter of the getORAData( ) method when retrieving an OrdAudio object from an OracleResultSet or OracleCallableStatement object.

Parameters

None.

Return Value

This method returns the OrdAudio implementation of the ORADataFactory interface.

Exceptions

None.

Examples

None.

getFormat( )

Format

public String getFormat( )

Description

Returns the value of the format attribute.

Parameters

None.

Return Value

This method returns the value of the format attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the format attribute.

Examples

None.

getMimeType( )

Format

public String getMimeType( )

Description

Returns the value of the mimeType attribute.

Parameters

None.

Return Value

This method returns the value of the mimeType attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the mimeType attribute.

Examples

None.

getNumberOfChannels( )

Format

public int getNumberOfChannels( )

Description

Returns the value of the numberOfChannels attribute.

Parameters

None.

Return Value

This method returns the value of the numberOfChannels attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the numberOfChannels attribute.

Examples

None.

getSampleSize( )

Format

public int getSampleSize( )

Description

Returns the value of the sampleSize attribute.

Parameters

None.

Return Value

This method returns the value of the sampleSize attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the sampleSize attribute.

Examples

None.

getSamplingRate( )

Format

public int getSamplingRate( )

Description

Returns the value of the samplingRate attribute.

Parameters

None.

Return Value

This method returns the value of the samplingRate attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the samplingRate attribute.

Examples

None.

getSource( )

Format

public String getSource( )

Description

Returns the source information in the form: srcType://srcLocation/srcName.

Parameters

None.

Return Value

This method returns the source information, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getSource( ) method in the database.

Examples

None.

getSourceLocation( )

Format

public String getSourceLocation( )

Description

Returns the value of the srcLocation attribute.

Parameters

None.

Return Value

This method returns the value of the srcLocation attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcLocation attribute.

Examples

None.

getSourceName( )

Format

public String getSourceName( )

Description

Returns the value of the srcName attribute.

Parameters

None.

Return Value

This method returns the value of the srcName attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcName attribute.

Examples

None.

getSourceType( )

Format

public String getSourceType( )

Description

Returns the value of the srcType attribute.

Parameters

None.

Return Value

This method returns the value of the srcType attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcType attribute.

Examples

None.

getUpdateTime( )

Format

public java.sql.Timestamp getUpdateTime( )

Description

Returns the value of the updateTime attribute.

Parameters

None.

Return Value

This method returns the value of the updateTime attribute as a java.sql.Timestamp object.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the updateTime attribute.

Examples

None.

importData(byte[ ] [ ])

Format

public void importData(byte[ ] [ ] ctx)

Description

Imports data from an external source into the database BLOB specified by the localData attribute. The external data source is specified by the srcType, srcLocation, and srcName attributes.

Parameters

ctx
The source plug-in context information.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding import( ) method in the database.

Examples

None.

importFrom(byte[ ] [ ], String, String, String)

Format

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

Description

Imports data from an external source into the database BLOB specified by the localData attribute. The external data source is specified by the srcType, srcLocation, and srcName parameters. The srcType, srcLocation, and srcName attributes are updated with values of the srcType, srcLocation, and srcName parameters passed to the importFrom( ) method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

srcType
The source type from which the data will be imported.

srcLocation
The source location from which the data will be imported.

srcName
The source name from which the data will be imported.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding importFrom( ) method in the database.

Examples

None.

isLocal( )

Format

public boolean isLocal( )

Description

Indicates if the audio data is stored locally in the database in a BLOB specified by the localData attribute.

Parameters

None.

Return Value

This method returns true if the data is stored locally in the database in a BLOB; false otherwise.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

loadDataFromByteArray(byte[ ])

Format

public boolean loadDataFromByteArray(byte[ ] byteArr)

Description

Loads data from a byte array into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

byteArr
A byte array from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the byte array.

Examples

None.

loadDataFromFile(String)

Format

public boolean loadDataFromFile(String filename)

Description

Loads data from a file into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

filename
The name of the file from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the data file.

Examples

None.

loadDataFromInputStream(InputStream)

Format

public boolean loadDataFromInputStream(InputStream inpStream)

Description

Loads data from an InputStream object into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

inpStream
The InputStream object from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the InputStream object.

Examples

None.

openSource(byte[ ], byte[ ] [ ])

Format

public int openSource(byte[ ] userarg, byte[ ] [ ] ctx)

Description

Opens a data source.

Parameters

userarg
Additional source plug-in information that may be required by user-defined source plug-ins.

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

Return Value

This method returns the status as an integer, where zero indicates success and a non-zero value indicates a failure code specific to the source plug-in.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding openSource( ) method in the database.

Examples

None.

processAudioCommand(byte[ ] [ ], String, String, byte[ ] [ ])

Format

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

Description

Calls the format plug-in in the database to execute a command. This method is used with user-written format plug-ins only; it raises an exception if used with the format plug-ins supplied by Oracle.

Parameters

ctx
The format plug-in context information.

cmd
The command to be executed by the format plug-in.

args
The arguments of the command.

result
A byte array of the form [1][*n*] into which the result of the command execution is written.

Return Value

This method returns the results of executing the command.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding processAudioCommand( ) method in the database.

Examples

None.

processSourceCommand(byte[ ] [ ], String, String, byte[ ] [ ])

Format

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

Description

Calls the source plug-in in the database to execute a command. This method is used with user-written plug-ins only; it raises an exception if used with the source plug-ins supplied by Oracle.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

cmd
The command to be executed by the source plug-in.

args
The arguments of the command.

result
A byte array of the form [1][*n*] into which the result of the command execution is written.

Return Value

This method returns the results of executing the command.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding processSourceCommand( ) method in the database.

Examples

None.

readFromSource(byte[ ] [ ], int, int, byte[ ] [ ])

Format

public int readFromSource(byte[ ] [ ] ctx, int startpos, int numbytes, byte[ ] [ ] buffer)

Description

Reads data from the data source. This method reads the specified number of bytes into the application buffer from the data source, starting at the specified position in the data source.

Not all source plug-ins require that the data source be opened before it can be read. However, to ensure that an application will work with any current or future source plug-ins, call the openSource(byte[ ], byte[ ] [ ]) method before calling this method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

startpos
The start position in the data source.

numbytes
The number of bytes to be read from the data source.

buffer
A byte array of the form [1][*n*], where *n* is greater than or equal to numbytes.

Return Value

This method returns the number of bytes read, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding readFromSource( ) method in the database.

Examples

None.

setAudioDuration(int)

Format

public void setAudioDuration(int audioDuration)

Description

Sets the value of the audioDuration attribute.

The setProperties( ) method sets this attribute automatically for certain audio formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the audio data itself.

Parameters

audioDuration
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the audioDuration attribute.

Examples

None.

setComments(CLOB)

Format

public void setComments(oracle.sql.CLOB comments)

Description

Sets the value of the comments attribute.

The comments attribute is reserved for use by *inter*Media. You can set your own value, but it could be overwritten by Oracle *inter*Media Annotator or by the setProperties( ) method.

Parameters

comments
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the comments attribute.

Examples

None.

setCompressionType(String)

Format

public void setCompressionType(String compressionType)

Description

Sets the value of the compressionType attribute.

The setProperties( ) method sets this attribute automatically for certain audio formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the audio data itself.

Parameters

compressionType
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the compressionType attribute.

Examples

None.

setDescription(String)

Format

public void setDescription(String description)

Description

Sets the value of the description attribute.

Parameters

description
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the description attribute.

Examples

None.

setEncoding(String)

Format

public void setEncoding(String encoding)

Description

Sets the value of the encoding attribute.

The setProperties( ) method sets this attribute automatically for certain audio formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the audio data itself.

Parameters

encoding
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the encoding attribute.

Examples

None.

setFormat(String)

Format

public void setFormat(String format)

Description

Sets the value of the format attribute.

The format attribute determines which format plug-in is used to handle calls to methods that operate on the audio data. In particular, the setProperties( ) method uses the format attribute to determine which format plug-in to call to parse the audio data properties. See the setProperties( ) method for more information on how to initialize the format attribute before calling the setProperties( ) method, and for information on how the setProperties( ) method in the default, Oracle-supplied plug-in, sets the value of the format attribute. Calling the setFormat( ) method sets only the attribute value; it does not modify the audio data itself.

Parameters

format
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the format attribute.

Examples

None.

setKnownAttributes(String, String, int, int, int, String, int)

Format

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

Description

Sets the values of the known attributes of the OrdAudio Java object.

The setProperties( ) method sets the following attributes automatically for certain audio formats: format, encoding, numberOfChannels, samplingRate, sampleSize, compressionType, and audioDuration. Use this method only if you are not using the setProperties( ) method. This method sets only the specified attribute values; it does not modify the audio data itself.

Parameters

format
The new attribute value, as a String.

encoding
The new attribute value, as a String.

numberOfChannels
The new attribute value, as an integer.

samplingRate
The new attribute value, as an integer.

sampleSize
The new attribute value, as an integer.

compressionType
The new attribute value, as a String.

audioDuration
The new attribute value, as an integer.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setKnownAttributes( ) method in the database.

Examples

None.

setLocal( )

Format

public void setLocal( )

Description

Sets the value of the local attribute to indicate that the audio data is stored locally in the database in a BLOB specified by the localData attribute.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

setMimeType(String)

Format

public void setMimeType(String mimeType)

Description

Sets the value of the mimeType attribute.

The setProperties( ) method sets this attribute automatically for certain audio formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the audio data itself.

Parameters

mimeType
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the mimeType attribute.

Examples

None.

setNumberOfChannels(int)

Format

public void setNumberOfChannels(int numberOfChannels)

Description

Sets the value of the numberOfChannels attribute.

The setProperties( ) method sets this attribute automatically for certain audio formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the audio data itself.

Parameters

numberOfChannels
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the numberOfChannels attribute.

Examples

None.

setProperties(byte[ ][ ])

Format

public void setProperties(byte[ ] [ ] ctx)

Description

Parses the audio data properties and sets the values of the attributes in the OrdAudio Java object. This method sets the values of the format, mimeType, encoding, numberOfChannels, samplingRate, sampleSize, compressionType, and audioDuration attributes. An attribute is set to null if the corresponding property cannot be extracted for a specific audio format. This method throws a SQLException error if the audio format is not recognized.

The format attribute determines which format plug-in is used to parse the audio data properties. If the format attribute is null when the setProperties( ) method is called, then the default, Oracle-supplied, format plug-in is used to parse the audio data properties and fill in various attributes, including the actual audio data format, for supported audio formats. See *Oracle interMedia Reference* for information on the audio formats supported by the Oracle-supplied format plug-ins. Note that the ORDAudio.init methods in the database always set the value of the format attribute to null. If the format attribute is not null, then the format plug-in specified by the format attribute will be called when the setProperties( ) method is called.

Parameters

ctx
The format plug-in context information.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setProperties( ) method in the database.

Examples

None.

setProperties(byte[ ][ ], boolean)

Format

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

Description

Parses the audio data properties, sets the values of the attributes in the OrdAudio Java object, and optionally populates the CLOB specified by the comments attribute. This method sets the values of the format, mimeType, encoding, numberOfChannels, samplingRate, sampleSize, compressionType, and audioDuration attributes. An attribute is set to null if the corresponding property cannot be extracted for a specific audio format. If the setComments parameter is true, this method also populates the CLOB specified by the comments attribute with all extracted properties in XML form. If the setComments parameter is false, the comments attribute is not modified. This method throws a SQLException error if the audio format is not recognized.

The format attribute determines which format plug-in is used to parse the audio data properties. If the format attribute is null when the setProperties( ) method is called, then the default, Oracle-supplied, format plug-in is used to parse the audio data properties and fill in various attributes, including the actual audio data format, for supported audio formats. See *Oracle interMedia Reference* for information on the audio formats supported by the Oracle-supplied format plug-ins. Note that the ORDAudio.init methods in the database always set the value of the format attribute to null. If the format attribute is not null, then the format plug-in specified by the format attribute will be called when the setProperties( ) method is called.

Parameters

ctx
The format plug-in context information.

setComments
A Boolean value that specifies whether or not to populate the CLOB specified by the comments attribute.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setProperties( ) method in the database.

Examples

None.

setSampleSize(int)

Format

public void setSampleSize(int sampleSize)

Description

Sets the value of the sampleSize attribute.

The setProperties( ) method sets this attribute automatically for certain audio formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the audio data itself.

Parameters

sampleSize
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the sampleSize attribute.

Examples

None.

setSamplingRate(int)

Format

public void setSamplingRate(int samplingRate)

Description

Sets the value of the samplingRate attribute.

The setProperties( ) method sets this attribute automatically for certain audio formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the audio data itself.

Parameters

samplingRate
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the samplingRate attribute.

Examples

None.

setSource(String, String, String)

Format

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

Description

Sets the values of the srcType, srcLocation, and srcName attributes.

Parameters

srcType
The type of the source.

srcLocation
The location of the source.

srcName
The name of the source.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcType, srcLocation, or srcName attributes.

Examples

None.

setUpdateTime(Timestamp)

Format

public void setUpdateTime(java.sql.Timestamp currentTime)

Description

Sets the value of the updateTime attribute. This method sets the value of the updateTime attribute to the specified time, or to the current SYSDATE time if the currentTime attribute is specified as null.

Parameters

currentTime
The update time, or the null value, used to set the value of the updateTime attribute to the current SYSDATE time.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setUpdateTime( ) method in the database.

Examples

None.

trimSource(byte[ ] [ ], int)

Format

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

Description

Trims the data to the specified length.

Not all source plug-ins support trim operations. For example, applications can trim the data stored in a BLOB specified by the localData attribute; however, the "file" and "http" data source types do not support write access, and so do not support this method. Furthermore, those source plug-ins that do support write access may not support the trim operation.

Not all source plug-ins require that the data source be opened before it can be modified. However, to ensure that an application will work with any current or future source plug-ins, call the openSource(byte[ ], byte[ ] [ ]) method before calling this method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

newLen
The length to which the data is to be trimmed.

Return Value

This method returns the status as an integer, where zero indicates success and a non-zero value indicates a failure code specific to the source plug-in.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding trimSource( ) method in the database.

Examples

None.

writeToSource(byte[ ] [ ], int, int, byte[ ])

Format

public int writeToSource(byte[ ] [ ] ctx, int startpos, int numbytes, byte[ ] buffer)

Description

Writes data to the data source. This method writes the specified number of bytes from the application buffer to the data source, starting at the specified position in the data source.

Not all source plug-ins support write operations. For example, applications can write to a BLOB specified by the localData attribute; however, the "file" and "http" data source types do not support write access, and so do not support this method. Furthermore, those source plug-ins that do support write access may support only sequential write access, and may not support write access to arbitrary starting positions within the data source.

Not all source plug-ins require that the data source be opened before it can be written. However, to ensure that an application will work with any current or future source plug-ins, call the openSource(byte[ ], byte[ ] [ ]) method before calling this method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

startpos
The start position in the data source.

numbytes
The number of bytes to be written to the data source.

buffer
A byte array containing the data to be written.

Return Value

This method returns the number of bytes written, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding writeToSource( ) method in the database.

Examples

None.

3

OrdDoc

This chapter contains reference information for the oracle.ord.im.OrdDoc class.

The OrdDoc class is used to represent an instance of the ORDSYS.ORDDoc database type in a Java application. The OrdDoc class includes a set of methods to get and set various object attributes, as well as a set of methods to perform various operations on an OrdDoc Java object.

Almost all methods operate on the attributes of the OrdDoc Java object in the application. The exceptions are those methods that access the media data for read or write purposes, which are described in the following list:

- Methods that operate on the database BLOB specified by the localData attribute, read and write data stored in the database BLOB.
- Methods that operate on the database BFILE specified by the srcLocation and srcName attributes when the srcType attribute is "file," read data from the specified file.
- Methods that operate on the URL specified by the srcType, srcLocation, and srcName attributes when the srcType attribute is "http," read data from the resource at the specified URL.

If your application modifies the OrdDoc Java object, or the media data in the database, you must update the ORDDoc SQL object in the database to make those changes permanent.

Some methods in the OrdDoc Java class are handed off to a database source plug-in or database format plug-in for processing; these methods have `byte [ ] [ ] ctx` as a context parameter. Applications should allocate a 64-byte array to hold any context information that may be required by a source plug-in or a format plug-in. For example, a plug-in may initialize the context information in one call and use that information in a subsequent call. The source plug-in context requires one array; the format plug-in context requires another array. For most plug-ins, 64

bytes should be sufficient. Some user-defined plug-ins may need additional space. The following example illustrates how to allocate a plug-in context information array:

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

> **Note:** In the current release, no Oracle-supplied source plug-ins or format plug-ins maintain context. Also, not all user-written source plug-ins or format plug-ins maintain context. However, if you include the context parameter as described, your application should work with any current or future source plug-ins or format plug-ins.

See *Oracle interMedia Reference* for more information about plug-ins.

3.1 Prerequisites

In order to run OrdDoc methods, you will need to include the following import statements in your Java file:

```
import oracle.ord.im.OrdMediaUtil;
import oracle.ord.im.OrdDoc;
```

You may also need to import classes from the following Java packages:

```
java.io.
java.sql.
oracle.jdbc.
```

Before running OrdDoc methods, the following operations must have already been performed:

- A connection has been made to a table that contains a column of type OrdDoc.
- A local OrdDoc object has been created and populated with data.

For examples of making a connection and populating a local object, see *Oracle interMedia User's Guide*.

3.2 Reference Information

This section presents reference information on the methods that operate on OrdDoc objects.

clearLocal( )

Format

public void clearLocal( )

Description

Clears the local attribute to indicate that the media data is stored externally.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the local attribute.

Examples

None.

closeSource(byte[ ] [ ])

Format

public int closeSource(byte[ ] [ ] ctx)

Description

Closes a data source.

Parameters

ctx
The source plug-in context information.

Return Value

This method returns the status as an integer, where zero indicates success and a non-zero value indicates a failure code specific to the source plug-in.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding closeSource( ) method in the database.

Examples

None.

deleteContent( )

Format

public void deleteContent( )

Description

Deletes any data stored in the database BLOB specified by the localData attribute.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding deleteContent( ) method in the database.

Examples

None.

export(byte[ ] [ ], String, String, String)

Format

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

Description

Exports the data from the BLOB specified by the localData attribute. This method calls the corresponding export( ) method in the database to export the media data to a location specified by the srcType, srcLocation, and srcName parameters.

Not all source plug-ins support the export( ) method. Only the "file" source type is natively supported.

This method will work only if you are running Oracle release 8.1.7 or later.

The remainder of this description describes the use of the export( ) method and the Oracle-supplied "file" source plug-in. User-written plug-ins will behave differently.

The export( ) method implemented by the Oracle-supplied "file" source plug-in copies, but does not modify, the media data stored in the database BLOB specified by the localData attribute.

After exporting the media data, all the media property attributes remain unchanged. However, the srcType, srcLocation, and srcName attributes are updated with values of the srcType, srcLocation, and srcName parameters passed to the export( ) method. After calling the export( ) method, if you no longer intend to manage the media data within the database, call the clearLocal( ) method to indicate that the media data is stored outside the database, and call the deleteContent( ) method to delete the media data stored in the database BLOB.

See *Oracle interMedia Reference* for information about the privileges required to write to a database directory object. See *Oracle Database Java Developer's Guide* and the java.io.FilePermission class in the Java API for information about security and performance.

Parameters

ctx
The source plug-in context information.

srcType
The source type to which the content will be exported.

srcLocation
The source location to which the content will be exported.

srcName
The source name to which the content will be exported.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding export( ) method in the database.

Examples

None.

getBFILE( )

Format

public oracle.sql.BFILE getBFILE( )

Description

Returns a BFILE locator from the database when the value of the srcType attribute is file. This method calls the corresponding getBFILE( ) method in the database, which creates the BFILE using the srcLocation and srcName attributes.

Parameters

None.

Return Value

This method returns an oracle.sql.BFILE locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getBFILE( ) method in the database.

Examples

None.

getComments( )

Format

public oracle.sql.CLOB getComments( )

Description

Returns the CLOB locator from the comments attribute.

Parameters

None.

Return Value

This method returns the value of the comments attribute as an oracle.sql.CLOB locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the comments attribute.

Examples

None.

getContent( )

Format

public oracle.sql.BLOB getContent( )

Description

Returns the BLOB locator from the localData attribute.

Parameters

None.

Return Value

This method returns an oracle.sql.BLOB locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

getContentInLob(byte[ ] [ ], String, String)

Format

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

Description

Returns the data from the BLOB specified by the localData attribute in a temporary BLOB in the database. This method creates a temporary BLOB in the database, reads the data from the BLOB specified by the localData attribute, writes the data to the temporary BLOB, then returns the temporary BLOB locator to the caller.

Note: The application must free the temporary BLOB after accessing the data it contains.

Parameters

ctx
The source plug-in context information.

mimetype
A String array, 1 element in length, into which the mimeType attribute is written as element 0.

format
A String array, 1 element in length, into which the format attribute is written as element 0.

Return Value

This method returns the media data in a temporary oracle.sql.BLOB locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs creating the temporary BLOB or executing the corresponding getContentInLob( ) method in the database.

Examples

None.

getContentLength( )

Format

public int getContentLength( )

Description

Returns the value of the contentLength attribute.

Parameters

None.

Return Value

This method returns the value of the contentLength attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the contentLength attribute.

Examples

None.

getDataInByteArray( )

Format

public byte[ ] getDataInByteArray( )

Description

Returns a byte array containing the data from the database BLOB specified by the localData attribute.

Parameters

None.

Return Value

This method returns a byte array containing the data.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

java.io.IOException

This exception is thrown if an error occurs reading the data from the BLOB.

java.lang.OutOfMemoryError

This exception is thrown if sufficient memory cannot be allocated to hold the data.

Examples

None.

getDataInFile(String)

Format

public boolean getDataInFile(String filename)

Description

Writes the data from the database BLOB specified by the localData attribute to a local file.

Parameters

filename
The name of the file to which the data will be written.

Return Value

This method returns true if the data is written to the file successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

java.io.IOException

This exception is thrown if an error occurs reading the data from the BLOB or writing the data to the output file.

Examples

None.

getDataInStream( )

Format

public InputStream getDataInStream( )

Description

Returns an InputStream object from which the data in the database BLOB specified by the localData attribute can be read.

Parameters

None.

Return Value

This method returns an InputStream object from which the data will be read.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

Examples

None.

getORADataFactory( )

Format

public static oracle.sql.ORADataFactory getORADataFactory( )

Description

Returns the OrdDoc ORADataFactory interface for use by the getORAData( ) method. Specify the getORADataFactory( ) method as the factory parameter of the getORAData( ) method when retrieving an OrdDoc object from an OracleResultSet or OracleCallableStatement object.

Parameters

None.

Return Value

This method returns the OrdDoc implementation of the ORADataFactory interface.

Exceptions

None.

Examples

None.

getFormat( )

Format

public String getFormat( )

Description

Returns the value of the format attribute.

Parameters

None.

Return Value

This method returns the value of the format attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the format attribute.

Examples

None.

getMimeType( )

Format

public String getMimeType( )

Description

Returns the value of the mimeType attribute.

Parameters

None.

Return Value

This method returns the value of the mimeType attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the mimeType attribute.

Examples

None.

getSource( )

Format

public String getSource( )

Description

Returns the source information in the form: srcType://srcLocation/srcName.

Parameters

None.

Return Value

This method returns the source information, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getSource( ) method in the database.

Examples

None.

getSourceLocation( )

Format

public String getSourceLocation( )

Description

Returns the value of the srcLocation attribute.

Parameters

None.

Return Value

This method returns the value of the srcLocation attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcLocation attribute.

Examples

None.

getSourceName( )

Format

public String getSourceName( )

Description

Returns the value of the srcName attribute.

Parameters

None.

Return Value

This method returns the value of the srcName attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcName attribute.

Examples

None.

getSourceType( )

Format

public String getSourceType( )

Description

Returns the value of the srcType attribute.

Parameters

None.

Return Value

This method returns the value of the srcType attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcType attribute.

Examples

None.

getUpdateTime( )

Format

public java.sql.Timestamp getUpdateTime( )

Description

Returns the value of the updateTime attribute.

Parameters

None.

Return Value

This method returns the value of the updateTime attribute as a java.sql.Timestamp object.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the updateTime attribute.

Examples

None.

importData(byte[ ] [ ], boolean)

Format

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

Description

Imports data from an external source into the database BLOB specified by the localData attribute, and optionally calls the setProperties( ) method. If the setProp parameter is true, this method calls the setProperties( ) method in the database to set the property attributes and populate the CLOB specified by the comments attribute. The external data source is specified by the srcType, srcLocation, and srcName attributes.

Parameters

ctx
The source plug-in context information.

setProp
A Boolean value that specifies whether or not to call the setProperties( ) method.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding import( ) method in the database.

Examples

None.

importFrom(byte[ ] [ ], String, String, String, boolean)

Format

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

Description

Imports data from an external source into the database BLOB specified by the localData attribute, and optionally calls the setProperties( ) method. If the setProp parameter is true, this method calls the setProperties( ) method in the database to set the property attributes and populate the CLOB specified by the comments attribute. The external data source is specified by the srcType, srcLocation, and srcName parameters. The srcType, srcLocation, and srcName attributes are updated with values of the srcType, srcLocation, and srcName parameters passed to the importFrom( ) method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

srcType
The source type from which the data will be imported.

srcLocation
The source location from which the data will be imported.

srcName
The source name from which the data will be imported.

setProp
A Boolean value that specifies whether or not to call the setProperties( ) method.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding importFrom( ) method in the database.

Examples

None.

isLocal( )

Format

public boolean isLocal( )

Description

Indicates if the media data is stored locally in the database in a BLOB specified by the localData attribute.

Parameters

None.

Return Value

This method returns true if the data is stored locally in the database in a BLOB; false otherwise.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

loadDataFromByteArray(byte[ ])

Format

public boolean loadDataFromByteArray(byte[ ] byteArr)

Description

Loads data from a byte array into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

byteArr
The name of the local byte array from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the byte array.

Examples

None.

loadDataFromFile(String)

Format

public boolean loadDataFromFile(String filename)

Description

Loads data from a file into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

filename
The name of the local file from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the data file.

Examples

None.

loadDataFromInputStream(InputStream)

Format

public boolean loadDataFromInputStream(InputStream inpStream)

Description

Loads data from an InputStream object into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

inpStream
The name of the InputStream object from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the InputStream object.

Examples

None.

openSource(byte[ ], byte[ ] [ ])

Format

public int openSource(byte[ ] userarg, byte[ ] [ ] ctx)

Description

Opens a data source.

Parameters

userarg
Additional source plug-in information that may be required by user-defined source plug-ins.

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

Return Value

This method returns the status as an integer, where zero indicates success and a non-zero value indicates a failure code specific to the source plug-in.

Exceptions

java.lang.Exception

This exception is thrown if an error occurs executing the corresponding openSource( ) method in the database.

Examples

None.

processSourceCommand(byte[ ] [ ], String, String, byte[ ] [ ])

Format

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

Description

Calls the source plug-in in the database to execute a command. This method is used with user-written plug-ins only; it raises an exception if used with the source plug-ins supplied by Oracle.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

cmd
The command to be executed by the source plug-in.

args
The arguments of the command.

result
A byte array of the form [1][*n*] into which the result of the command execution is written.

Return Value

This method returns the results of executing the command.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding processSourceCommand( ) method in the database.

Examples

None.

readFromSource(byte[ ] [ ], int, int, byte[ ] [ ])

Format

public int readFromSource(byte[ ] [ ] ctx, int startpos, int numbytes, byte[ ] [ ] buffer)

Description

Reads data from the data source. This method reads the specified number of bytes into the application buffer from the data source starting at the specified position in the data source.

Not all source plug-ins require that the data source be opened before it can be read. However, to ensure that an application will work with any current or future source plug-ins, call the openSource(byte[ ], byte[ ] [ ]) method before calling this method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

startpos
The start position in the data source.

numbytes
The number of bytes to be read from the data source.

buffer
A byte array of the form [1][*n*], where *n* is greater than or equal to numbytes.

Return Value

This method returns the number of bytes read, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding readFromSource( ) method in the database.

Examples

None.

setComments(CLOB)

Format

public void setComments(oracle.sql.CLOB comments)

Description

Sets the value of the comments attribute.

The comments attribute is reserved for use by *inter*Media. You can set your own value, but it could be overwritten by Oracle *inter*Media Annotator or by the setProperties(byte[ ] [ ], boolean) method.

Parameters

comments
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the comments attribute.

Examples

None.

setContentLength(int)

Format

public void setContentLength(int contentLength)

Description

Sets the value of the contentLength attribute.

The setProperties( ) method sets this attribute automatically for certain media formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the media data itself.

Parameters

contentLength
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the contentLength attribute.

Examples

None.

setFormat(String)

Format

public void setFormat(String format)

Description

Sets the value of the format attribute.

The format attribute determines which format plug-in is used to handle calls to methods that operate on the media data. In particular, the setProperties( ) method uses the format attribute to determine which format plug-in to call to parse the media data properties. See the setProperties( ) method for more information on how to initialize the format attribute before calling the setProperties( ) method, and for information on how the setProperties( ) method in the default, Oracle-supplied plug-in, sets the value of the format attribute. Calling this method sets only the attribute value; it does not modify the media data itself.

Parameters

format
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the format attribute.

Examples

None.

setLocal( )

Format

public void setLocal( )

Description

Sets the value of the local attribute to indicate that the media data is stored locally in the database in a BLOB specified by the localData attribute.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

setMimeType(String)

Format

public void setMimeType(String mimeType)

Description

Sets the value of the mimeType attribute.

The setProperties( ) method sets this attribute automatically for certain media formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the media data itself.

Parameters

mimeType
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the mimeType attribute.

Examples

None.

setProperties(byte[ ] [ ], boolean)

Format

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

Description

Parses the media data properties, sets the values of the attributes in the OrdDoc Java object, and optionally populates the CLOB specified by the comments attribute. This method sets the values of the format, mimeType, and contentLength attributes. An attribute is set to null if the corresponding property cannot be extracted for a specific media format. If the setComments parameter is true, this method also populates the CLOB specified by the comments attribute with all extracted properties in XML form. If the setComments parameter is false, the comments attribute is not modified. This method throws a SQLException error if the media format is not recognized.

The format attribute determines which format plug-in is used to parse the media data properties. If the format attribute is null when the setProperties( ) method is called, then the default, Oracle-supplied, format plug-in is used to parse the media data properties and fill in various attributes, including the actual media data format, for supported media formats. See *Oracle interMedia Reference* for information on the media formats supported by the Oracle-supplied format plug-ins. Note that the ORDDoc.init methods in the database always set the value of the format attribute to null. If the format attribute is not null, then the format plug-in specified by the format attribute will be called when the setProperties( ) method is called.

Parameters

ctx
The format plug-in context information.

setComments
A Boolean value that specifies whether or not to populate the CLOB specified by the comments attribute.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setProperties( ) method in the database.

Examples

None.

setSource(String, String, String)

Format

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

Description

Sets the values of the srcType, srcLocation, and srcName attributes.

Parameters

srcType
The type of the source.

srcLocation
The location of the source.

srcName
The name of the source.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcType, srcLocation, or srcName attributes.

Examples

None.

setUpdateTime(Timestamp)

Format

public void setUpdateTime(java.sql.Timestamp currentTime)

Description

Sets the value of the updateTime attribute. This method sets the value of the updateTime attribute to the specified time, or to the current SYSDATE time if the currentTime attribute is specified as null.

Parameters

currentTime
The update time, or the null value, used to set the value of the updateTime attribute to the current SYSDATE time.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setUpdateTime( ) method in the database.

Examples

None.

trimSource(byte[ ] [ ], int)

Format

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

Description

Trims the data to the specified length.

Not all source plug-ins support trim operations. For example, applications can trim the data stored in a BLOB specified by the localData attribute; however, the "file" and "http" data source types do not support write access, and so do not support this method. Furthermore, those source plug-ins that do support write access may not support the trim operation.

Not all source plug-ins require that the data source be opened before it can be modified. However, to ensure that an application will work with any current or future source plug-ins, call the openSource(byte[ ], byte[ ] [ ]) method before calling this method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

newLen
The length to which the data is to be trimmed.

Return Value

This method returns the status as an integer, where zero indicates success and a non-zero value indicates a failure code specific to the source plug-in.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding trimSource( ) method in the database.

Examples

None.

writeToSource(byte[ ] [ ], int, int, byte[ ])

Format

public int writeToSource(byte[ ] [ ] ctx, int startpos, int numbytes, byte[ ] buffer)

Description

Writes data to the data source. This method writes the specified number of bytes from the application buffer to the data source, starting at the specified position in the data source.

Not all source plug-ins support write operations. For example, applications can write to a BLOB specified by the localData attribute; however, the "file" and "http" data source types do not support write access, and so do not support this method. Furthermore, those source plug-ins that do support write access may support only sequential write access, and may not support write access to arbitrary starting positions within the data source.

Not all source plug-ins require that the data source be opened before it can be written. However, to ensure that an application will work with any current or future source plug-ins, call the openSource(byte[ ], byte[ ] [ ]) method before calling this method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

startpos
The start position in the data source.

numbytes
The number of bytes to be written to the data source.

buffer
A byte array containing the data to be written.

Return Value

This method returns the number of bytes written, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding writeToSource( ) method in the database.

Examples

None.

4

OrdImage

This chapter contains reference information for the oracle.ord.im.OrdImage class.

The OrdImage class is used to represent an instance of the ORDSYS.ORDImage database type in a Java application. The OrdImage class includes a set of methods to get and set various object attributes, as well as a set of methods to perform various operations on an OrdImage Java object.

Almost all methods operate on the attributes of the OrdImage Java object in the application. The exceptions are those methods that access the image data for read or write purposes, which are described in the following list:

- Methods that operate on the database BLOB specified by the localData attribute, read and write data stored in the database BLOB.
- Methods that operate on the database BFILE specified by the srcLocation and srcName attributes when the srcType attribute is "file," read data from the specified file.
- Methods that operate on the URL specified by the srcType, srcLocation, and srcName attributes when the srcType attribute is "http," read data from the resource at the specified URL.

If your application modifies the OrdImage Java object, or the image data in the database, you must update the ORDImage SQL object in the database to make those changes permanent.

Some methods in the OrdImage Java class are handed off to a database source plug-in for processing; these methods have `byte [ ] [ ] ctx` as a context parameter. Applications should allocate a 64-byte array to hold any context information that may be required by a source plug-in. For example, a plug-in may initialize the context information in one call and use that information in a subsequent call. For most plug-ins, 64 bytes should be sufficient. Some user-defined

plug-ins may need additional space. The following example shows how to allocate a plug-in context information array:

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

> **Note:** In the current release, no Oracle-supplied source plug-ins maintain context. Also, not all user-written source plug-ins maintain context. However, if you include the context parameter as described, your application should work with any current or future source plug-ins.

See *Oracle interMedia Reference* for more information about plug-ins.

4.1 Prerequisites

In order to run OrdImage methods, you will need to include the following import statements in your Java file:

```
import oracle.ord.im.OrdMediaUtil;
import oracle.ord.im.OrdImage;
```

You may also need to import classes from the following Java packages:

```
java.io.
java.sql.
oracle.jdbc.
```

Before running OrdImage methods, the following operations must have already been performed:

- A connection has been made to a table that contains a column of type OrdImage.
- A local OrdImage object has been created and populated with data.

For examples of making a connection and populating a local object, see *Oracle interMedia User's Guide*.

4.2 Reference Information

This section presents reference information on the methods that operate on OrdImage objects.

checkProperties( )

Format

public boolean checkProperties( )

Description

Checks if the properties of the image data are consistent with the attributes of the OrdImage Java object.

Parameters

None.

Return Value

This method returns true if the properties of the image data are consistent with the attributes of the OrdImage Java object; false otherwise.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding checkProperties( ) method in the database.

Examples

None.

clearLocal( )

Format

public void clearLocal( )

Description

Clears the local attribute to indicate that the image data is stored externally.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the local attribute.

Examples

None.

copy(OrdImage)

Format

public void copy(OrdImage dest)

Description

Copies an OrdImage Java object. This method calls the corresponding copy( ) method in the database. The copy( ) method copies all the attributes of the current OrdImage object to the destination OrdImage object with the exception of the BLOB specified by the localData attribute. If the image data is stored locally in the database, then the data is copied *from* the BLOB specified by the localData attribute in the current OrdImage object *to* the BLOB specified by the localData attribute in the destination object.

Parameters

dest
The destination OrdImage object to which the data will be copied.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs calling the corresponding copy( ) method in the database.

Examples

None.

deleteContent( )

Format

public void deleteContent( )

Description

Deletes any data stored in the database BLOB specified by the localData attribute.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding deleteContent( ) method in the database.

Examples

None.

export(byte[ ] [ ], String, String, String)

Format

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

Description

Exports the data from the BLOB specified by the localData attribute. This method calls the corresponding export( ) method in the database to export the image data to a location specified by the srcType, srcLocation, and srcName parameters.

Not all source plug-ins support the export( ) method. For example, the "file" source type is the only Oracle-supplied source type that supports the export( ) method.

This method will work only if you are running Oracle release 8.1.7 or later.

The remainder of this description describes the use of the export( ) method and the Oracle-supplied "file" source plug-in. User-written plug-ins will behave differently.

The export( ) method implemented by the Oracle-supplied "file" source plug-in copies, but does not modify, the media data stored in the database BLOB specified by the localData attribute.

After exporting the image data, all the image property attributes remain unchanged, however, the srcType, srcLocation, and srcName attributes are updated with values of the srcType, srcLocation, and srcName parameters passed to the export( ) method. After calling the export( ) method, if you no longer intend to manage the image data within the database, call the clearLocal( ) method to indicate the image data is stored outside the database and call the deleteContent( ) method to delete the image data stored in the database BLOB.

See *Oracle interMedia Reference* for information about the privileges required to write to a database directory object. See *Oracle Database Java Developer's Guide* and the java.io.FilePermission class in the Java API for information about security and performance.

Parameters

ctx
The source plug-in context information.

srcType
The source type to which the content will be exported.

srcLocation
The source location to which the content will be exported.

srcName
The source name to which the content will be exported.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding export( ) method in the database.

Examples

None.

getBFILE( )

Format

public oracle.sql.BFILE getBFILE( )

Description

Returns a BFILE locator from the database when the value of the srcType attribute is file. This method calls the corresponding getBFILE( ) method in the database, which creates the BFILE using the srcLocation and srcName attributes.

Parameters

None.

Return Value

This method returns an oracle.sql.BFILE locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getBFILE( ) method in the database.

Examples

None.

getCompressionFormat( )

Format

public String getCompressionFormat( )

Description

Returns the value of the compressionFormat attribute.

Parameters

None.

Return Value

This method returns the value of the compressionFormat attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the compressionFormat attribute.

Examples

None.

getContent( )

Format

public oracle.sql.BLOB getContent( )

Description

Returns the BLOB locator from the localData attribute.

Parameters

None.

Return Value

This method returns an oracle.sql.BLOB locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

getContentFormat( )

Format

public String getContentFormat( )

Description

Returns the value of the contentFormat attribute.

Parameters

None.

Return Value

This method returns the value of the contentFormat attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the contentFormat attribute.

Examples

None.

getContentLength( )

Format

public int getContentLength( )

Description

Returns the value of the contentLength attribute.

Parameters

None.

Return Value

This method returns the value of the contentLength attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the contentLength attribute.

Examples

None.

getDataInByteArray( )

Format

public byte[ ] getDataInByteArray( )

Description

Returns a byte array containing the data from the database BLOB specified by the localData attribute.

Parameters

None.

Return Value

This method returns the byte array containing the data.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

java.io.IOException

This exception is thrown if an error occurs reading the data from the BLOB.

java.lang.OutOfMemoryError

This exception is thrown if sufficient memory cannot be allocated to hold the data.

Examples

None.

getDataInFile(String)

Format

public boolean getDataInFile(String filename)

Description

Writes the data from the database BLOB specified by the localData attribute to a local file.

Parameters

filename
The name of the file to which the data will be written.

Return Value

This method returns true if the data is written to the file successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

java.io.IOException

This exception is thrown if an error occurs reading the data from the BLOB or writing the data to the output file.

Examples

None.

getDataInStream( )

Format

public InputStream getDataInStream( )

Description

Returns an InputStream object from which the data in the database BLOB specified by the localData attribute can be read.

Parameters

None.

Return Value

This method returns an InputStream object from which the data will be read.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

Examples

None.

getORADataFactory( )

Format

public static oracle.sql.ORADataFactory getORADataFactory( )

Description

Returns the OrdImage ORADataFactory interface for use by the getORAData( ) method. Specify the getORADataFactory( ) method as the factory parameter of the getORAData( ) method when retrieving an OrdImage object from an OracleResultSet or OracleCallableStatement object.

Parameters

None.

Return Value

This method returns the OrdImage implementation of the ORADataFactory interface.

Exceptions

None.

Examples

None.

getFormat( )

Format

public String getFormat( )

Description

Returns the value of the fileFormat attribute.

Parameters

None.

Return Value

This method returns the value of the fileFormat attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the fileFormat attribute.

Examples

None.

getHeight( )

Format

public int getHeight( )

Description

Returns the value of the height attribute.

Parameters

None.

Return Value

This method returns the value of the height attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the height attribute.

Examples

None.

getMimeType( )

Format

public String getMimeType( )

Description

Returns the value of the mimeType attribute.

Parameters

None.

Return Value

This method returns the value of the mimeType attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the mimeType attribute.

Examples

None.

getSource( )

Format

public String getSource( )

Description

Returns the source information in the form: srcType://srcLocation/srcName.

Parameters

None.

Return Value

This method returns the source information, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getSource( ) method in the database.

Examples

None.

getSourceLocation( )

Format

public String getSourceLocation( )

Description

Returns the value of the srcLocation attribute.

Parameters

None.

Return Value

This method returns the value of the srcLocation attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcLocation attribute.

Examples

None.

getSourceName( )

Format

public String getSourceName( )

Description

Returns the value of the srcName attribute.

Parameters

None.

Return Value

This method returns the value of the srcName attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcName attribute.

Examples

None.

getSourceType( )

Format

public String getSourceType( )

Description

Returns the value of the srcType attribute.

Parameters

None.

Return Value

This method returns the value of the srcType attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcType attribute.

Examples

None.

getUpdateTime( )

Format

public java.sql.Timestamp getUpdateTime( )

Description

Returns the value of the updateTime attribute.

Parameters

None.

Return Value

This method returns the value of the updateTime attribute as a java.sql.Timestamp object.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the updateTime attribute.

Examples

None.

getWidth( )

Format

```
public int getWidth( )
```

Description

Returns the value of the width attribute.

Parameters

None.

Return Value

This method returns the value of the width attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the width attribute.

Examples

None.

importData(byte[ ] [ ])

Format

public void importData(byte[ ] [ ] ctx)

Description

Imports data from an external source into the database BLOB specified by the localData attribute. The external data source is specified by the srcType, srcLocation, and srcName attributes. After importing the image data, by default, this method automatically calls the setProperties( ) method in the database to set the property attributes. If you are importing a foreign image whose format is not understood by *inter*Media, call the setFormat(String) method to set the fileFormat to a String beginning with "other" to disable the automatic call to the setProperties( ) method.

Parameters

ctx
The source plug-in context information.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding import( ) method or the setProperties( ) method in the database.

Examples

None.

importFrom(byte[ ] [ ], String, String, String)

Format

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

Description

Imports data from an external source into the database BLOB specified by the localData attribute. The external data source is specified by the srcType, srcLocation, and srcName parameters. The srcType, srcLocation, and srcName attributes are updated with values of the srcType, srcLocation, and srcName parameters passed to the importFrom( ) method. After importing the image data, by default, this method automatically calls the setProperties( ) method in the database to set the property attributes. If you are importing a foreign image whose format is not understood by *inter*Media, call the setFormat(String) method to set the fileFormat to "other" to disable the automatic call to the setProperties( ) method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

srcType
The source type from which the data will be imported.

srcLocation
The source location from which the data will be imported.

srcName
The source name from which the data will be imported.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding importFrom( ) method or the setProperties( ) method in the database.

Examples

None.

isLocal( )

Format

```
public boolean isLocal( )
```

Description

Indicates if the image data is stored locally in the database in a BLOB specified by the localData attribute.

Parameters

None.

Return Value

This method returns true if the data is stored locally in the database in a BLOB; false otherwise.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

loadDataFromByteArray(byte[ ])

Format

public boolean loadDataFromByteArray(byte[ ] byteArr)

Description

Loads data from a byte array into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

byteArr
The name of the byte array from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the byte array.

Examples

None.

loadDataFromFile(String)

Format

public boolean loadDataFromFile(String filename)

Description

Loads data from a file into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

filename
The name of the file from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the data file.

Examples

None.

loadDataFromInputStream(InputStream)

Format

public boolean loadDataFromInputStream(InputStream inpStream)

Description

Loads data from an InputStream object into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

inpStream
The name of the InputStream object from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the InputStream object.

Examples

None.

process(String)

Format

public void process(String cmd)

Description

Performs one or more image processing operations on the image data in the database BLOB specified by the localData attribute. This method calls the corresponding process( ) method in the database to perform the image processing operations specified by the cmd parameter.

See *Oracle interMedia Reference* for more information on the various image processing operations that can be performed on an image.

Parameters

cmd
A String that specifies a list of image processing operations to perform on the image.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding process( ) method in the database.

Examples

None.

processCopy(String, OrdImage)

Format

public void processCopy(String cmd, OrdImage dest)

Description

Copies the image data to the destination object and performs one or more image processing operations on the image data. If the source image data is stored externally to the database, then it is imported into the database BLOB specified by the localData attribute in the destination OrdImage object. Otherwise, the image data is copied *from* the BLOB specified by localData attribute in the current OrdImage object *to* the BLOB specified by the localData attribute in the destination object.

Parameters

cmd
A String that specifies a list of image processing operations to perform on the image.

dest
The destination OrdImage object.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs calling the corresponding processCopy( ) method in the database.

Examples

None.

setCompressionFormat(String)

Format

public void setCompressionFormat(String compressionFormat)

Description

Sets the value of the compressionFormat attribute.

The setProperties( ) method sets this attribute automatically for certain media formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the media data itself.

Parameters

compressionFormat
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the compressionFormat attribute.

Examples

None.

setContentFormat(String)

Format

public void setContentFormat(String contentFormat)

Description

Sets the value of the contentFormat attribute.

The setProperties( ) method sets this attribute automatically for certain media formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the media data itself.

Parameters

contentFormat
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the contentFormat attribute.

Examples

None.

setContentLength(int)

Format

public void setContentLength(int contentLength)

Description

Sets the value of the contentLength attribute.

The setProperties( ) method sets this attribute automatically for certain media formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the media data itself.

Parameters

contentLength
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the contentLength attribute.

Examples

None.

setFormat(String)

Format

public void setFormat(String fileFormat)

Description

Sets the value of the fileFormat attribute.

The setProperties( ) method sets this attribute automatically for certain media formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the media data itself. Set the fileFormat attribute to a String beginning with "other" to disable the automatic call to the setProperties( ) method by the importData( ) and importFrom( ) methods.

Parameters

fileFormat
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the fileFormat attribute.

Examples

None.

setHeight(int)

Format

public void setHeight(int height)

Description

Sets the value of the height attribute.

The setProperties( ) method sets this attribute automatically for certain image formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the image data itself.

Parameters

height
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the height attribute.

Examples

None.

setLocal( )

Format

public void setLocal( )

Description

Sets the value of the local attribute to indicate that the image data is stored locally in the database in a BLOB specified by the localData attribute.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

setMimeType(String)

Format

public void setMimeType(String mimeType)

Description

Sets the value of the mimeType attribute.

The setProperties( ) method sets this attribute automatically for certain media formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the media data itself.

Parameters

mimeType
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the mimeType attribute.

Examples

None.

setProperties( )

Format

public void setProperties( )

Description

Parses the image data properties and sets the values of the attributes in the OrdImage Java object. This method sets the values of the height, width, contentLength, fileFormat, contentFormat, compressionFormat, and mimeType attributes. An attribute is set to null if the corresponding property cannot be extracted for a specific image format. This method throws a SQLException error if the image format is not recognized.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setProperties( ) method in the database.

Examples

None.

setProperties(String)

Format

public void setProperties(String description)

Description

Writes the characteristics of a foreign image into the appropriate attribute fields. This method sets the values of various attributes of the OrdImage object based on a set of characteristics that describes the image properties. With this information, *inter*Media is able to process certain foreign image formats. For more information on setting image characteristics for foreign images, see *Oracle interMedia Reference*.

Parameters

description
A String that specifies the image characteristics to set for the foreign image.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setProperties( ) method in the database.

Examples

None.

setSource(String, String, String)

Format

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

Description

Sets the values of the srcType, srcLocation, and srcName attributes.

Parameters

srcType
The type of the source.

srcLocation
The location of the source.

srcName
The name of the source.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcType, srcLocation, or srcName attributes.

Examples

None.

setUpdateTime(Timestamp)

Format

public void setUpdateTime(java.sql.Timestamp currentTime)

Description

Sets the value of the updateTime attribute. This method sets the value of the updateTime attribute to the specified time, or to the current SYSDATE time if the currentTime attribute is specified as null.

Parameters

currentTime
The update time, or the null value, used to set the value of the updateTime attribute to the current SYSDATE time.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setUpdateTime( ) method in the database.

Examples

None.

setWidth(int)

Format

public void setWidth(int width)

Description

Sets the value of the width attribute.

The setProperties( ) method sets this attribute automatically for certain image formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the image data itself.

Parameters

width
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the width attribute.

Examples

None.

5

OrdImageSignature

This chapter contains reference information for the oracle.ord.im.OrdImageSignature class.

The OrdImageSignature class is used to represent an instance of the ORDSYS.ORDImageSignature database type in a Java application. The OrdImageSignature class includes a method to generate an image signature, as well as static methods to compare two image signatures.

If your application modifies the OrdImageSignature object, you must perform a SQL update operation to make those changes permanent.

Image signature indexes are used to improve performance of database operations involving image matching. To take advantage of the increased performance that is possible using image matching with image signature indexes on the underlying tables, use the IMGSimilar and IMGScore SQL operators.

See *Oracle interMedia Reference* for more information about image signature indexes.

> **Note:** All *inter*Media features are available with the Standard Edition of Oracle Database, except image indexing, which uses the ORDImageSignature object. The image indexing feature requires bit-mapped indexing, which is available only when you install the Enterprise Edition of Oracle Database.

5.1 Prerequisites

In order to run OrdImageSignature methods, you will need to include the following import statements in your Java file:

```
import oracle.ord.im.OrdMediaUtil;
import oracle.ord.im.OrdImageSignature;
```

You may also need to import classes from the following Java packages:

```
java.io.
java.sql.
oracle.jdbc.
```

Before running OrdImageSignature methods, the following operations must have already been performed:

- A connection has been made to a table that contains a column of type OrdImageSignature.
- A local OrdImageSignature object has been created and populated with data.

5.2 Reference Information

This section presents reference information on the methods that operate on OrdImageSignature objects.

evaluateScore(OrdImageSignature, OrdImageSignature, String)

Format

public static float evaluateScore(OrdImageSignature signature1, OrdImageSignature signature2, String attrWeights)

Description

Compares two image signatures, returning a score that indicates the degree of difference between the image signatures. This method compares the image signatures in signature1 and signature2 using weights specified for one or more visual attributes. Returns a score between 0.0 and 100.0, where a lower value indicates a closer match.

Specify a weight in the range 0.0 to 1.0 for one or more of the following visual attributes: color, shape, texture, location.

You must specify a value greater than 0.0 for at least one of the following attributes: color, shape, or texture. The location attribute indicates the importance of the distribution of the color, shape, or texture features in the images. During processing, the values are normalized such that they total 1.0. For example:

```
color=0.7,shape=0.3
```

Note: The ORDImageSignature evaluateScore( ) method operates on two image signatures, not on indexes on database tables. Therefore, this method cannot take advantage of the increased performance that is possible using image matching with image signature indexes on the underlying tables. To use image signature indexes, use the IMGSimilar and IMGScore SQL operators.

See *Oracle interMedia Reference* for more information about image signature indexes.

Parameters

signature1
The first OrdImageSignature.

signature2
An OrdImageSignature to be compared to signature1.

attrWeights
A String that specifies a list of one or more visual attributes and the weight to be applied to each attribute.

Return Value

This method returns the score, as a floating-point value.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs calling the evaluateScore( ) method in the database.

Examples

None.

generateSignature(OrdImage)

Format

public void generateSignature(OrdImage img)

Description

Generates an image signature for the specified image.

Parameters

img
An OrdImage object from which to generate the signature.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs calling the generateSignature( ) method in the database.

Examples

None.

getORADataFactory( )

Format

public static oracle.sql.ORADataFactory getORADataFactory( )

Description

Returns the OrdImageSignature ORADataFactory interface for use by the getORAData( ) method. Specify the getORADataFactory( ) method as the factory parameter of the getORAData( ) method when retrieving an OrdImageSignature object from an OracleResultSet or OracleCallableStatement object.

Parameters

None.

Return Value

This method returns the OrdImageSignature implementation of the ORADataFactory interface.

Exceptions

None.

Examples

None.

isSimilar(OrdImageSignature, OrdImageSignature, String, float)

Format

public static int isSimilar(OrdImageSignature signature1, OrdImageSignature signature2,
String attrWeights, float threshold)

Description

Compares two image signatures, returning a status that indicates if the degree of difference between the image signatures is within a specified threshold. This method compares the image signatures in signature1 and signature2 using weights specified for one or more visual attributes. The result of the comparison is a score between 0.0 and 100.0, where a lower value indicates a closer match. If the score is less than or equal to the specified threshold, the images are considered a match and the method returns 1; otherwise, the method returns 0.

Specify a weight in the range 0.0 to 1.0 for one or more of the following visual attributes: color, shape, texture, location.

You must specify a value greater than 0.0 for at least one of the following attributes: color, shape, or texture. The location attribute indicates the importance of the distribution of the color, shape, or texture features in the images. During processing, the values are normalized such that they total 1.0. For example:

```
color=0.7,shape=0.3
```

Note: The ORDImageSignature isSimilar( ) method operates on two image signatures, not on indexes on database tables. Therefore, this method cannot take advantage of the increased performance that is possible using image matching with image signature indexes on the underlying tables. To use image signature indexes, use the IMGSimilar and IMGScore SQL operators.

See *Oracle interMedia Reference* for more information about image signature indexes.

Parameters

signature1
The first OrdImageSignature.

signature2
An OrdImageSignature to be compared to signature1.

attrWeights
A String that specifies a list of one or more visual attributes and the weight to be applied to each attribute.

threshold
A floating-point value that specifies the degree of similarity required for the two images to be considered a match.

Return Value

This method returns an integer value of 1 if the images match; otherwise, it returns 0.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs calling the isSimilar( ) method in the database.

Examples

None.

6

OrdMediaUtil

This chapter contains reference information for the oracle.ord.im.OrdMediaUtil class.

The OrdMediaUtil class allows other *inter*Media Java classes to connect to Oracle Database and access data in tables that contain columns of *inter*Media object types. This class includes methods for downloading and uploading data.

The OrdMediaUtil class also includes a Java compatibility initialization function to enable client-side applications to maintain compatibility with previous releases of the *inter*Media object types (OrdAudio, OrdDoc, OrdImage, and OrdVideo).

Oracle may improve the *inter*Media object types by adding new object attributes in a future release of *inter*Media. Client-side applications may be able to maintain compatibility with previous releases of the *inter*Media object types (OrdAudio, OrdDoc, OrdImage, and OrdVideo), even after a database upgrade that changes the object types, if they make a call to this compatibility initialization function at the beginning of the application.

Client-side applications written in Java using *inter*Media Java Classes should call the OrdMediaUtil imCompatibilityInit( ) method after connecting to Oracle Database. This Java function takes an OracleConnection object as an argument.

See the imCompatibilityInit(OracleConnection) method for more information about the Java compatibility initialization function.

See *Oracle interMedia User's Guide* for an example of the OrdMediaUtil.imCompatibilityInit( ) method.

6.1 Prerequisites

Before running OrdMediaUtil methods, the following operations must have already been performed:

- A connection has been made to a table that contains columns of the appropriate *inter*Media object types (OrdAudio, OrdDoc, and so on).
- Local *inter*Media objects have been created and populated with data.

For examples of making a connection and populating a local object, see *Oracle interMedia User's Guide*.

6.2 Reference Information

This section presents reference information on the methods that operate on OrdMediaUtil objects.

getDataInByteArray(BLOB)

Format

public static byte[ ] getDataInByteArray(oracle.sql.BLOB mediaLobLocator)

Description

Downloads data from a database to a byte array.

Parameters

mediaLobLocator
The BLOB from which data is downloaded, and which returns a byte array containing downloaded data.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

java.io.IOException

This exception is thrown if an error occurs reading the data from the BLOB.

java.lang.OutOfMemoryError

This exception is thrown if sufficient memory cannot be allocated to hold the data.

Examples

None.

getDataInFile(String, BLOB)

Format

public static boolean getDataInFile(java.lang.String filename, oracle.sql.BLOB mediaLobLocator)

Description

Downloads data from a database to a file.

Parameters

filename
The name of the file into which the data is downloaded.

mediaLobLocator
The BLOB from which the data is downloaded.

Return Value

This method returns true if the download is successful; false otherwise.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

java.io.IOException

This exception is thrown if an error occurs reading the data from the BLOB or writing the data to the output file.

Examples

None.

imCompatibilityInit(OracleConnection)

Format

public static void imCompatibilityInit(oracle.jdbc.OracleConnection con)

Description

Uses the class oracle.jdbc.OracleConnection. This placeholder method is a compatibility initialization function that may help applications maintain compatibility with future releases of *inter*Media.

Parameters

con
An instance of the OracleConnection object.

Return Value

None.

Exceptions

java.lang.Exception

This exception is thrown if an error occurs executing the SQL connection statements.

Examples

None.

loadData(String, BLOB)

Format

public static boolean loadData(java.lang.String filename, oracle.sql.BLOB mediaLobLocator)

Description

Loads data from a file.

Parameters

filename
The name of the file from which the data is loaded.

mediaLobLocator
The BLOB into which data is loaded.

Return Value

This method returns true if the load is successful; false otherwise.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the data file.

Examples

None.

loadDataFromByteArray(byte[ ], BLOB)

Format

public static boolean loadDataFromByteArray(byte[ ] byteArr, oracle.sql.BLOB mediaLobLocator)

Description

Loads data from a byte array.

Parameters

byteArr
The name of the byte array from which the data is loaded.

mediaLobLocator
The BLOB into which data is loaded.

Return Value

This method returns true if the load is successful; false otherwise.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the byte array.

Examples

None.

loadDataFromInputStream(InputStream, BLOB)

Format

public static boolean loadDataFromInputStream(java.io.InputStream inStream, oracle.sql.BLOB mediaLobLocator)

Description

Loads data from an input stream.

Parameters

inStream
The name of the input stream from which the data is loaded.

mediaLobLocator
The BLOB into which the data is loaded.

Return Value

This method returns true if the load is successful; false otherwise.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the InputStream object.

java.lang.OutOfMemoryError

This exception is thrown if sufficient memory cannot be allocated to hold the data.

Examples

None.

7

OrdVideo

This chapter contains reference information for the oracle.ord.im.OrdVideo class.

The OrdVideo class is used to represent an instance of the ORDSYS.ORDVideo database type in a Java application. The OrdVideo class includes a set of methods to get and set various object attributes, as well as a set of methods to perform various operations on an OrdVideo Java object.

Almost all methods operate on the attributes of the OrdVideo Java object in the application. The exceptions are those methods that access the video data for read or write purposes, which are described in the following list:

- Methods that operate on the database BLOB specified by the localData attribute, read and write data stored in the database BLOB.
- Methods that operate on the database BFILE specified by the srcLocation and srcName attributes when the srcType attribute is "file," read data from the specified file.
- Methods that operate on the URL specified by the srcType, srcLocation, and srcName attributes when the srcType attribute is "http," read data from the resource at the specified URL.

If your application modifies the OrdVideo Java object, or the video data in the database, you must update the ORDVideo SQL object in the database to make those changes permanent.

Some methods in the OrdVideo Java class are handed off to a database source plug-in or database format plug-in for processing; these methods have `byte [ ] [ ] ctx` as a context parameter. Applications should allocate a 64-byte array to hold any context information that may be required by a source plug-in or a format plug-in. For example, a plug-in may initialize the context information in one call and use that information in a subsequent call. The source plug-in context requires one array; the format plug-in context requires another array. For most plug-ins, 64

bytes should be sufficient. Some user-defined plug-ins may need additional space. The following example shows how to allocate a plug-in context information array:

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

> **Note:** In the current release, no Oracle-supplied source plug-ins or format plug-ins maintain context. Also, not all user-written source plug-ins or format plug-ins maintain context. However, if you include the context parameter as described, your application should work with any current or future source plug-ins or format plug-ins.

See *Oracle interMedia Reference* for more information about plug-ins.

7.1 Prerequisites

In order to run OrdVideo methods, you will need to include the following import statements in your Java file:

```
import oracle.ord.im.OrdMediaUtil;
import oracle.ord.im.OrdVideo;
```

You may also need to import classes from the following Java packages:

```
java.io.
java.sql.
oracle.jdbc.
```

Before running OrdVideo methods, the following operations must have already been performed:

- A connection has been made to a table that contains a column of type OrdVideo.
- A local OrdVideo object has been created and populated with data.

For examples of making a connection and populating a local object, see *Oracle interMedia User's Guide*.

7.2 Reference Information

This section presents reference information on the methods that operate on OrdVideo objects.

checkProperties(byte[ ] [ ])

Format

public boolean checkProperties(byte[ ] [ ] ctx)

Description

Checks if the properties of the video data are consistent with the attributes of the OrdVideo Java object.

Parameters

ctx
The format plug-in context information.

Return Value

This method returns true if the properties of the video data are consistent with the attributes of the OrdVideo Java object; false otherwise.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding checkProperties( ) method in the database.

Examples

None.

clearLocal( )

Format

public void clearLocal( )

Description

Clears the local attribute to indicate that the video data is stored externally.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the local attribute.

Examples

None.

closeSource(byte[ ] [ ])

Format

public int closeSource(byte[ ] [ ] ctx)

Description

Closes a data source.

Parameters

ctx
The source plug-in context information.

Return Value

This method returns the status as an integer, where zero indicates success and a non-zero value indicates a failure code specific to the source plug-in.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding closeSource( ) method in the database.

Examples

None.

deleteContent( )

Format

public void deleteContent( )

Description

Deletes any data stored in the database BLOB specified by the localData attribute.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding deleteContent( ) method in the database.

Examples

None.

export(byte[ ] [ ], String, String, String)

Format

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

Description

Exports the data from the BLOB specified by the localData attribute. This method calls the corresponding export( ) method in the database to export the video data to a location specified by the srcType, srcLocation, and srcName parameters.

Not all source plug-ins support this method. Only the "file" source type is natively supported.

This method will work only if you are running Oracle release 8.1.7 or later.

The remainder of this description describes the use of the export( ) method and the Oracle-supplied "file" source plug-in. User-written plug-ins will behave differently.

The export( ) method implemented by the "file" source plug-in copies, but does not modify, the video data stored in the database BLOB specified by the localData attribute.

After exporting the video data, all the video property attributes remain unchanged. However, the srcType, srcLocation, and srcName attributes are updated with values of the srcType, srcLocation, and srcName parameters passed to the export( ) method. After calling the export( ) method, if you no longer intend to manage the video data within the database, call the clearLocal( ) method to indicate the video data is stored outside the database, and call the deleteContent( ) method to delete the video data stored in the database BLOB.

See *Oracle interMedia Reference* for information about the privileges required to write to a database directory object. See *Oracle Database Java Developer's Guide* and the java.io.FilePermission class in the Java API for information about security and performance.

Parameters

ctx
The source plug-in context information.

srcType
The source type to which the content will be exported.

srcLocation
The source location to which the content will be exported.

srcName
The source name to which the content will be exported.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding export( ) method in the database.

Examples

None.

getAllAttributes(byte[ ] [ ])

Format

public oracle.sql.CLOB getAllAttributes(byte[ ] [ ] ctx)

Description

Returns the values of the video properties in a temporary CLOB in a form defined by the format plug-in. For natively supported formats, the information is presented as a comma-separated list of attributes in the form attributeName=attributeValue, where the list contains the following attributes: format, mimeType, width, height, frameResolution, frameRate, videoDuration, numberOfFrames, compressionType, numberOfColors, and bitRate. For user-defined formats, the information is presented in a form defined by the format plug-in.

> **Note:** The application must free the temporary CLOB after reading the information it contains.

Parameters

ctx[ ]
The format plug-in context information.

Return Value

This method returns the values of the attributes as a temporary CLOB, oracle.sql.CLOB.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getAllAttributes( ) method in the database.

Examples

None.

getAttribute(byte[ ] [ ], String)

Format

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

Description

Returns the value of the requested video property. This method is used by user-defined format plug-ins to return the value of a video property that is not available as an attribute of the OrdVideo Java object. This method is not implemented by any Oracle-supplied format plug-ins.

Parameters

ctx
The format plug-in context information.

name
The property or attribute name.

Return Value

This method returns the value of the attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getAttribute( ) method in the database.

Examples

None.

getBFILE( )

Format

public oracle.sql.BFILE getBFILE( )

Description

Returns a BFILE locator from the database when the value of the srcType attribute is file. This method calls the corresponding getBFILE( ) method in the database, which creates the BFILE using the srcLocation and srcName attributes.

Parameters

None.

Return Value

This method returns an oracle.sql.BFILE locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getBFILE( ) method in the database.

Examples

None.

getBitRate( )

Format

public int getBitRate( )

Description

Returns the value of the bitRate attribute.

Parameters

None.

Return Value

This method returns the value of the bitRate attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the bitRate attribute.

Examples

None.

getComments( )

Format

public oracle.sql.CLOB getComments( )

Description

Returns the CLOB locator from the comments attribute.

Parameters

None.

Return Value

This method returns the value of the comments attribute, as an oracle.sql.CLOB locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the comments attribute.

Examples

None.

getCompressionType( )

Format

public String getCompressionType( )

Description

Returns the value of the compressionType attribute.

Parameters

None.

Return Value

This method returns the value of the compressionType attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the compressionType attribute.

Examples

None.

getContent( )

Format

public oracle.sql.BLOB getContent( )

Description

Returns the BLOB locator from the localData attribute.

Parameters

None.

Return Value

This method returns an oracle.sql.BLOB locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

getContentInLob(byte[ ] [ ], String, String)

Format

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

Description

Returns the data from the BLOB specified by the localData attribute in a temporary BLOB in the database. This method creates a temporary BLOB in the database, reads the data from the BLOB specified by the localData attribute, writes the data to the temporary BLOB, then returns the temporary BLOB locator to the caller.

> **Note:** The application must free the temporary BLOB after accessing the data it contains.

Parameters

ctx
The source plug-in context information.

mimetype
A String array, 1 element in length, into which the mimeType attribute is written as element 0.

format
A String array, 1 element in length, into which the format attribute is written as element 0.

Return Value

This method returns the video data in a temporary oracle.sql.BLOB locator.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs creating the temporary BLOB or executing the corresponding getContentInLob( ) method in the database.

Examples

None.

getContentLength( )

Format

public int getContentLength( )

Description

Returns the length of the video data. This method calls the corresponding getContentLength( ) method in the database.

This method is not supported for all source types. For example, the "http" source type does not support this method.

Parameters

None.

Return Value

This method returns the value of the contentLength attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getContentLength( ) method in the database.

Examples

None.

getContentLength(byte[ ][ ])

Format

public int getContentLength(byte[ ] [ ] ctx)

Description

Returns the length of the video data using source plug-in context information. This method calls the corresponding getContentLength( ) method in the database.

Parameters

ctx
The source plug-in context information.

Return Value

This method returns the value of the contentLength attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getContentLength( ) method in the database.

Examples

None.

getDataInByteArray( )

Format

public byte[ ] getDataInByteArray( )

Description

Returns a byte array containing the data from the database BLOB specified by the localData attribute.

Parameters

None.

Return Value

This method returns a byte array containing the data.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

java.io.IOException

This exception is thrown if an error occurs reading the data from the BLOB.

java.lang.OutOfMemoryError

This exception is thrown if sufficient memory cannot be allocated to hold the data.

Examples

None.

getDataInFile(String)

Format

public boolean getDataInFile(String filename)

Description

Writes the data from the database BLOB specified by the localData attribute to a local file.

Parameters

filename
The name of the file to which the data will be written.

Return Value

This method returns true if the data is written to the file successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

java.io.IOException

This exception is thrown if an error occurs reading the data from the BLOB or writing the data to the output file.

Examples

None.

getDataInStream( )

Format

public InputStream getDataInStream( )

Description

Returns an InputStream object from which the data in the database BLOB specified by the localData attribute can be read.

Parameters

None.

Return Value

This method returns an InputStream object from which the data will be read.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute.

Examples

None.

getDescription( )

Format

public String getDescription( )

Description

Returns the value of the description attribute.

Parameters

None.

Return Value

This method returns the value of the description attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the description attribute.

Examples

None.

getORADataFactory( )

Format

public static oracle.sql.ORADataFactory getORADataFactory( )

Description

Returns the OrdVideo ORADataFactory interface for use by the getORAData( ) method. Specify the getORADataFactory( ) method as the factory parameter of the getORAData( ) method when retrieving an OrdVideo object from an OracleResultSet or OracleCallableStatement object.

Parameters

None.

Return Value

This method returns the OrdVideo implementation of the ORADataFactory interface.

Exceptions

None.

Examples

None.

getFormat( )

Format

public String getFormat( )

Description

Returns the value of the format attribute.

Parameters

None.

Return Value

This method returns the value of the format attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the format attribute.

Examples

None.

getFrameRate( )

Format

public int getFrameRate( )

Description

Returns the value of the frameRate attribute.

Parameters

None.

Return Value

This method returns the value of the frameRate attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the frameRate attribute.

Examples

None.

getFrameResolution( )

Format

public int getFrameResolution( )

Description

Returns the value of the frameResolution attribute.

Parameters

None.

Return Value

This method returns the value of the frameResolution attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the frameResolution attribute.

Examples

None.

getHeight( )

Format

public int getHeight( )

Description

Returns the value of the height attribute.

Parameters

None.

Return Value

This method returns the value of the height attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the height attribute.

Examples

None.

getMimeType( )

Format

public String getMimeType( )

Description

Returns the value of the mimeType attribute.

Parameters

None.

Return Value

This method returns the value of the mimeType attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the mimeType attribute.

Examples

None.

getNumberOfColors( )

Format

public int getNumberOfColors( )

Description

Returns the value of the numberOfColors attribute.

Parameters

None.

Return Value

This method returns the value of the numberOfColors attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the numberOfColors attribute.

Examples

None.

getNumberOfFrames( )

Format

public int getNumberOfFrames( )

Description

Returns the value of the numberOfFrames attribute.

Parameters

None.

Return Value

This method returns the value of the numberOfFrames attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the numberOfFrames attribute.

Examples

None.

getSource( )

Format

public String getSource( )

Description

Returns the source information in the form srcType://srcLocation/srcName.

Parameters

None.

Return Value

This method returns the source information, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding getSource( ) method in the database.

Examples

None.

getSourceLocation( )

Format

public String getSourceLocation( )

Description

Returns the value of the srcLocation attribute.

Parameters

None.

Return Value

This method returns the value of the srcLocation attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcLocation attribute.

Examples

None.

getSourceName( )

Format

public String getSourceName( )

Description

Returns the value of the srcName attribute.

Parameters

None.

Return Value

This method returns the value of the srcName attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcName attribute.

Examples

None.

getSourceType( )

Format

public String getSourceType( )

Description

Returns the value of the srcType attribute.

Parameters

None.

Return Value

This method returns the value of the srcType attribute, as a String.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcType attribute.

Examples

None.

getUpdateTime( )

Format

public java.sql.Timestamp getUpdateTime( )

Description

Returns the value of the updateTime attribute.

Parameters

None.

Return Value

This method returns the value of the updateTime attribute as a java.sql.Timestamp object.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the updateTime attribute.

Examples

None.

getVideoDuration( )

Format

public int getVideoDuration( )

Description

Returns the value of the videoDuration attribute.

Parameters

None.

Return Value

This method returns the value of the videoDuration attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the videoDuration attribute.

Examples

None.

getWidth( )

Format

```
public int getWidth( )
```

Description

Returns the value of the width attribute.

Parameters

None.

Return Value

This method returns the value of the width attribute, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the width attribute.

Examples

None.

importData(byte[ ] [ ])

Format

public void importData(byte[ ] [ ] ctx)

Description

Imports data from an external source into the database BLOB specified by the localData attribute. The external data source is specified by the srcType, srcLocation, and srcName attributes.

Parameters

ctx
The source plug-in context information.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding import( ) method in the database.

Examples

None.

importFrom(byte[ ] [ ], String, String, String)

Format

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

Description

Imports data from an external source into the database BLOB specified by the localData attribute. The external data source is specified by the srcType, srcLocation, and srcName parameters. The srcType, srcLocation, and srcName attributes are updated with values of the srcType, srcLocation, and srcName parameters passed to the importFrom( ) method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

srcType
The source type from which the data will be imported.

srcLocation
The source location from which the data will be imported.

srcName
The source name from which the data will be imported.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding importFrom( ) method in the database.

Examples

None.

isLocal( )

Format

```
public boolean isLocal( )
```

Description

Indicates if the video data is stored locally in the database in a BLOB specified by the localData attribute.

Parameters

None.

Return Value

This method returns true if the data is stored locally in the database in a BLOB; false otherwise.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

loadDataFromByteArray(byte[ ])

Format

public boolean loadDataFromByteArray(byte[ ] byteArr)

Description

Loads data from a byte array into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

byteArr
A byte array from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the byte array.

Examples

None.

loadDataFromFile(String)

Format

public boolean loadDataFromFile(String filename)

Description

Loads data from a file into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

filename
The name of the file from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the data file.

Examples

None.

loadDataFromInputStream(InputStream)

Format

public boolean loadDataFromInputStream(InputStream inpStream)

Description

Loads data from an InputStream object into the database BLOB specified by the localData attribute. Before loading the data, this method calls the deleteContent( ) method to delete any existing data in the BLOB. It also calls the setLocal( ) method to set the local flag. In addition, this method calls the setUpdateTime(Timestamp) method to set the updateTime attribute to the current SYSDATE time.

Parameters

inpStream
The InputStream object from which the data will be loaded.

Return Value

This method returns true if the data is loaded successfully; otherwise, an exception is raised if an error occurs. This method never returns false.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing an object attribute or executing a method in the database.

java.io.IOException

This exception is thrown if an error occurs reading the InputStream object.

Examples

None.

openSource(byte[ ], byte[ ] [ ])

Format

public int openSource(byte[ ] userarg, byte[ ] [ ] ctx)

Description

Opens a data source.

Parameters

userarg
Additional source plug-in information that may be required by user-defined source plug-ins.

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

Return Value

This method returns the status as an integer, where zero indicates success and a non-zero value indicates a failure code specific to the source plug-in.

Exceptions

java.lang.Exception

This exception is thrown if an error occurs executing the corresponding openSource( ) method in the database.

Examples

None.

processSourceCommand(byte[ ] [ ], String, String, byte[ ] [ ])

Format

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

Description

Calls the source plug-in in the database to execute a command. This method is used with user-written plug-ins only; this method raises an exception if used with the source plug-ins supplied by Oracle.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

cmd
The command to be executed by the source plug-in.

args
The command arguments.

result
A byte array of the form [1][*n*] into which the result of the command execution is written.

Return Value

This method returns the results of executing the command.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding processSourceCommand( ) method in the database.

Examples

None.

processVideoCommand(byte[ ] [ ], String, String, byte[ ] [ ])

Format

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

Description

Calls the format plug-in in the database to execute a command. This method is used with user-written format plug-ins only; this method raises an exception if used with the format plug-ins supplied by Oracle.

Parameters

ctx
The format plug-in context information.

cmd
The command to be executed by the format plug-in.

args
The command arguments.

result
A byte array of the form [1][*n*] into which the result of the command execution is written.

Return Value

This method returns the results of executing the command.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding processVideoCommand( ) method in the database.

Examples

None.

readFromSource(byte[ ] [ ], int, int, byte[ ] [ ])

Format

public int readFromSource(byte[ ] [ ] ctx, int startpos, int numbytes, byte[ ] [ ] buffer)

Description

Reads data from the data source. This method reads the specified number of bytes into the application buffer from the data source starting at the specified position in the data source.

Not all source plug-ins require that the data source be opened before it can be read. However, to ensure that an application will work with any current or future source plug-ins, call the openSource(byte[ ], byte[ ] [ ]) method before calling this method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

startpos
The start position in the data source.

numbytes
The number of bytes to be read from the data source.

buffer
A byte array of the form [1][*n*], where *n* is greater than or equal to numbytes.

Return Value

This method returns the number of bytes read, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding readFromSource( ) method in the database.

Examples

None.

setBitRate(int)

Format

public void setBitRate(int bitRate)

Description

Sets the value of the bitRate attribute.

The setProperties( ) method sets this attribute automatically for certain video formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the video data itself.

Parameters

bitRate
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the bitRate attribute.

Examples

None.

setComments(CLOB)

Format

public void setComments(oracle.sql.CLOB comments)

Description

Sets the value of the comments attribute.

The comments attribute is reserved for use by *inter*Media. You can set your own value, but it could be overwritten by Oracle *inter*Media Annotator or by the setProperties( ) method.

Parameters

comments
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the comments attribute.

Examples

None.

setCompressionType(String)

Format

public void setCompressionType(String compressionType)

Description

Sets the value of the compressionType attribute.

The setProperties( ) method sets this attribute automatically for certain video formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the video data itself.

Parameters

compressionType
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the compressionType attribute.

Examples

None.

setDescription(String)

Format

public void setDescription(String description)

Description

Sets the value of the description attribute.

Parameters

description
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the description attribute.

Examples

None.

setFormat(String)

Format

public void setFormat(String format)

Description

Sets the value of the format attribute.

The format attribute determines which format plug-in is used to handle calls to methods that operate on the video data. In particular, the setProperties( ) method uses the format attribute to determine which format plug-in to call to parse the video data properties. See the setProperties( ) method for more information on how to initialize the format attribute before calling the setProperties( ) method, and for information on how the setProperties( ) method in the default, Oracle-supplied plug-in, sets the value of the format attribute. Calling the setFormat( ) method sets only the attribute value; it does not modify the video data itself.

Parameters

format
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the format attribute.

Examples

None.

setFrameRate(int)

Format

public void setFrameRate(int frameRate)

Description

Sets the value of the frameRate attribute.

The setProperties( ) method sets this attribute automatically for certain video formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the video data itself.

Parameters

frameRate
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the frameRate attribute.

Examples

None.

setFrameResolution(int)

Format

public void setFrameResolution(int frameResolution)

Description

Sets the value of the frameResolution attribute.

The setProperties( ) method sets this attribute automatically for certain video formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the video data itself.

Parameters

frameResolution
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the frameResolution attribute.

Examples

None.

setHeight(int)

Format

public void setHeight(int height)

Description

Sets the value of the height attribute.

The setProperties( ) method sets this attribute automatically for certain video formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the video data itself.

Parameters

height
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the height attribute.

Examples

None.

setKnownAttributes(String, int, int, int, int, int, int, String, int, int)

Format

```
public void setKnownAttributes(String format, int width, int height, int frameResolution,
                               int frameRate, int videoDuration, int numberOfFrames,
                               String compressionType, int numberOfColors,
                                    int bitRate)
```

Description

Sets the values of the known attributes of the OrdVideo Java object.

The setProperties( ) method sets the values of the following attributes automatically for certain video formats: format, width, height, frameResolution, frameRate, videoDuration, numberOfFrames, compressionType, numberOfColors, and bitRate. Use this method only if you are not using the setProperties( ) method. This method sets only the specified attribute values; it does not modify the video data itself.

Parameters

format
The new attribute value, as a String.

width
The new attribute value, as an integer.

height
The new attribute value, as an integer.

frameResolution
The new attribute value, as an integer.

frameRate
The new attribute value, as an integer.

videoDuration
The new attribute value, as an integer.

numberOfFrames
The new attribute value, as an integer.

compressionType
The new attribute value, as a String.

numberOfColors
The new attribute value, as an integer.

bitRate
The new attribute value, as an integer.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setKnownAttributes( ) method in the database.

Examples

None.

setLocal( )

Format

public void setLocal( )

Description

Sets the value of the local attribute to indicate that the video data is stored locally in the database in a BLOB specified by the localData attribute.

Parameters

None.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the localData attribute.

Examples

None.

setMimeType(String)

Format

public void setMimeType(String mimeType)

Description

Sets the value of the mimeType attribute.

The setProperties( ) method sets this attribute automatically for certain video formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the video data itself.

Parameters

mimeType
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the mimeType attribute.

Examples

None.

setNumberOfColors(int)

Format

public void setNumberOfColors(int numberOfColors)

Description

Sets the value of the numberOfColors attribute.

The setProperties( ) method sets this attribute automatically for certain video formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the video data itself.

Parameters

numberOfColors
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the numberOfColors attribute.

Examples

None.

setNumberOfFrames(int)

Format

public void setNumberOfFrames(int numberOfFrames)

Description

Sets the value of the numberOfFrames attribute.

The setProperties( ) method sets this attribute automatically for certain video formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the video data itself.

Parameters

numberOfFrames
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the numberOfFrames attribute.

Examples

None.

setProperties(byte[ ][ ])

Format

public void setProperties(byte[ ] [ ] ctx)

Description

Parses the video data properties and sets the values of the attributes in the OrdVideo Java object. This method sets the values of the format, mimeType, width, height, frameResolution, frameRate, videoDuration, numberOfFrames, compressionType, numberOfColors, and bitRate attributes. An attribute is set to null if the corresponding property cannot be extracted for a specific video format. This method throws a SQLException error if the video format is not recognized.

The format attribute determines which format plug-in is used to parse the video data properties. If the format attribute is null when the setProperties( ) method is called, then the default, Oracle-supplied, format plug-in is used to parse the video data properties and fill in various attributes, including the actual video data format, for supported video formats. See *Oracle interMedia Reference* for information on the video formats supported by the Oracle-supplied format plug-ins. Note that the ORDVideo.init methods in the database always set the value of the format attribute to null. If the format attribute is not null, then the format plug-in specified by the format attribute will be called when the setProperties( ) method is called.

Parameters

ctx
The format plug-in context information.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setProperties( ) method in the database.

Examples

None.

setProperties(byte[ ][ ], boolean)

Format

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

Description

Parses the video data properties, sets the values of the attributes in the OrdVideo Java object, and optionally populates the CLOB specified by the comments attribute. This method sets the values of the format, mimeType, width, height, frameResolution, frameRate, videoDuration, numberOfFrames, compressionType, numberOfColors, and bitRate attributes. An attribute is set to null if the corresponding property cannot be extracted for a specific video format. If the setComments parameter is true, this method also populates the CLOB specified by the comments attribute with all extracted properties in XML form. If the setComments parameter is false, the comments attribute is not modified. This method throws a SQLException error if the video format is not recognized.

The format attribute determines which format plug-in is used to parse the video data properties. If the format attribute is null when the setProperties( ) method is called, then the default, Oracle-supplied, format plug-in is used to parse the video data properties and fill in various attributes, including the actual video data format, for supported video formats. See *Oracle interMedia Reference* for information on the video formats supported by the Oracle-supplied format plug-ins. Note that the ORDVideo.init methods in the database always set the value of the format attribute to null. If the format attribute is not null, then the format plug-in specified by the format attribute will be called when the setProperties( ) method is called.

Parameters

ctx
The format plug-in context information.

setComments
A Boolean value that specifies whether or not to populate the CLOB specified by the comments attribute.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setProperties( ) method in the database.

Examples

None.

setSource(String, String, String)

Format

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

Description

Sets the values of the srcType, srcLocation, and srcName attributes.

Parameters

srcType
The type of the source.

srcLocation
The location of the source.

srcName
The name of the source.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the srcType, srcLocation, or srcName attributes.

Examples

None.

setUpdateTime(Timestamp)

Format

public void setUpdateTime(java.sql.Timestamp currentTime)

Description

Sets the value of the updateTime attribute. This method sets the value of the updateTime attribute to the specified time, or to the current SYSDATE time if currentTime is specified as null.

Parameters

currentTime
The update time, or the null value, used to set the value of the updateTime attribute to the current SYSDATE time.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding setUpdateTime( ) method in the database.

Examples

None.

setVideoDuration(int)

Format

public void setVideoDuration(int videoDuration)

Description

Sets the value of the videoDuration attribute.

The setProperties( ) method sets this attribute automatically for certain video formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the video data itself.

Parameters

videoDuration
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the videoDuration attribute.

Examples

None.

setWidth(int)

Format

public void setWidth(int width)

Description

Sets the value of the width attribute.

The setProperties( ) method sets this attribute automatically for certain video formats; use this method only if you are not using the setProperties( ) method. This method sets only the attribute value; it does not modify the video data itself.

Parameters

width
The new attribute value.

Return Value

None.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs accessing the width attribute.

Examples

None.

trimSource(byte[ ] [ ], int)

Format

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

Description

Trims the data to the specified length.

Not all source plug-ins support trim operations. For example, applications can trim the data stored in a BLOB specified by the localData attribute; however, the "file" and "http" data source types do not support write access, and so do not support this method. Furthermore, those source plug-ins that do support write access may not support the trim operation.

Not all source plug-ins require that the data source be opened before it can be modified. However, to ensure that an application will work with any current or future source plug-ins, call the openSource( ) method before calling this method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

newLen
The length to which the data is to be trimmed.

Return Value

This method returns the status as an integer, where zero indicates success and a non-zero value indicates a failure code specific to the source plug-in.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding trimSource( ) method in the database.

Examples

None.

writeToSource(byte[ ] [ ], int, int, byte[ ])

Format

public int writeToSource(byte[ ] [ ] ctx, int startpos, int numbytes, byte[ ] buffer)

Description

Writes data to the data source. This method writes the specified number of bytes from the application buffer to the data source starting at the specified position in the data source.

Not all source plug-ins support write operations. For example, applications can write to a BLOB specified by the localData attribute; however, the "file" and "http" data source types do not support write access, and so do not support this method. Furthermore, those source plug-ins that do support write access may support only sequential write access, and may not support write access to arbitrary starting positions within the data source.

Not all source plug-ins require that the data source be opened before it can be written. However, to ensure that an application will work with any current or future source plug-ins, call the openSource( ) method before calling this method.

Parameters

ctx
The source plug-in context information. See *Oracle interMedia Reference* for more information.

startpos
The start position in the data source.

numbytes
The number of bytes to be written to the data source.

buffer
A byte array containing the data to be written.

Return Value

This method returns the number of bytes written, as an integer.

Exceptions

java.sql.SQLException

This exception is thrown if an error occurs executing the corresponding writeToSource( ) method in the database.

Examples

None.

8

JAI Input and Output Stream

As an extension to Java, Sun Microsystems has provided the Java Advanced Imaging (JAI) API. JAI lets you introduce advanced image processing operations in your Java applications. With Oracle *inter*Media ("*inter*Media"), you can read and write images stored in the database from your JAI applications.

Oracle *inter*Media Java Classes ("*inter*Media Java Classes") provides APIs for three types of stream objects, which let you read data from BLOBs and BFILEs and write to BLOBs in your JAI applications. These stream objects are not meant to replace the input and output stream objects provided by Sun Microsystems; these objects are included to provide an interface to image data stored in BLOBs and BFILEs in OrdImage objects that can be used by JAI without loss in performance.

The stream objects are as follows:

- BfileInputStream, which allows you to read data from an Oracle BFILE associated with the stream. (See BfileInputStream Object.)
- BlobInputStream, which allows you to read data from an Oracle BLOB associated with the stream. (See BlobInputStream Object.)
- BlobOutputStream, which allows you to write data from a buffer to an Oracle BLOB associated with the stream. (See BlobOutputStream Object.)

8.1 Prerequisites

In order to to use the JAI stream objects, you will need to include the following import statements in your Java file:

```
import oracle.sql.BLOB;
import oracle.sql.BFILE;
```

In order to use JAI with *inter*Media methods, you will also need to import classes from the `oracle.ord.media.jai.io` package into your Java file.

For more information on the Java Advanced Imaging API, see the following Web site (which is maintained by Sun Microsystems)

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

BfileInputStream Object

This section presents reference information on the methods associated with the BfileInputStream object, which provides an interface for JAI to read data from BFILEs. It is a subclass of com.sun.media.jai.codec.SeekableStream and java.io.InputStream; it implements java.io.DataInput.

Before running the methods associated with the BfileInputStream object, the following operations must have already been performed:

- The following import statements have been included:

```
import javax.media.jai.JAI;
import java.awt.image.RenderedImage;
```

- A local BfileInputStream object named inStream has been created.

BfileInputStream(BFILE)

Format

public BfileInputStream(oracle.sql.BFILE bfile)

Description

Creates a BfileInputStream object that reads from the specified BFILE. The constructor uses the maximum chunk size defined for a BFILE. The BFILE will be opened after this constructor executes.

Parameters

bfile
The BFILE from which data will be read.

Return Value

None.

Exceptions

java.io.IOException

java.sql.SQLException

Examples

None.

BfileInputStream(BFILE, int)

Format

public BfileInputStream(oracle.sql.BFILE bfile, int chunkSize)

Description

Creates a BfileInputStream object that reads from the specified BFILE. The constructor uses the specified chunk size. The BFILE will be opened after this constructor executes.

Parameters

bfile
The BFILE from which data will be read.

chunkSize
The maximum amount of data to read from the BFILE at one time.

Return Value

None.

Exceptions

java.io.IOException

java.sql.SQLException

Examples

None.

canSeekBackwards( )

Format

public boolean canSeekBackwards( )

Description

Checks whether or not the stream can read backwards. Because the BfileInputStream object can read backwards, this method will always return true.

Parameters

None.

Return Value

This method returns true.

Exceptions

None.

Examples

None.

close( )

Format

public void close( )

Description

Closes the BfileInputStream object, releasing any resources being used. The BFILE automatically closes after the stream closes.

Parameters

None.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

getBFILE( )

Format

public oracle.sql.BFILE getBFILE( )

Description

Returns the BFILE associated with the BfileInputStream object.

Parameters

None.

Return Value

This method returns the BFILE associated with the BfileInputStream object.

Exceptions

None.

Examples

None.

getFilePointer( )

Format

public long getFilePointer( )

Description

Returns the offset from the beginning of the BFILE at which the next read operation will occur.

Parameters

None.

Return Value

This method returns the offset from the beginning of the BFILE at which the next read operation will occur, in bytes.

Exceptions

java.io.IOException

Examples

None.

mark(int)

Format

public void mark(int readLimit)

Description

Marks the current position in the BfileInputStream object. A call to the reset( ) method will return you to the last marked position in the BfileInputStream object.

Parameters

readLimit
This argument is ignored by the class.

Return Value

None.

Exceptions

None.

Examples

None.

markSupported( )

Format

public boolean markSupported( )

Description

Checks whether or not the BfileInputStream object supports marking. Because the BfileInputStream object supports marking, this method will always return true.

Parameters

None.

Return Value

This method returns true.

Exceptions

None.

Examples

None.

read( )

Format

public int read( )

Description

Reads a single byte from the BFILE associated with the BfileInputStream object.

Parameters

None.

Return Value

This method returns the byte of data that is read, or -1 if the end of the BFILE has been reached.

Exceptions

java.io.IOException

Examples

None.

read(byte[ ])

Format

public int read(byte[ ] buffer)

Description

Reads data from the BFILE into the specified buffer.

Parameters

buffer
The buffer into which the data is read.

Return Value

This method returns the number of bytes read into the buffer, or -1 if the end of the BFILE was reached before any data was read. The value cannot exceed the length of the buffer.

Exceptions

java.io.IOException

Examples

None.

read(byte[ ], int, int)

Format

public int read(byte[ ]buffer, int off, int len)

Description

Reads up to the specified length of data from the BFILE into the specified buffer, starting from the specified offset.

Parameters

buffer
The buffer into which the data is read.

off
The offset from the beginning of the buffer at which data will be written, in bytes.

len
The maximum number of bytes to be read into the buffer.

Return Value

This method returns the number of bytes read, or -1 if the end of the BFILE was reached before any data was read. The value cannot exceed the length of the buffer.

Exceptions

java.io.IOException

Examples

None.

remaining( )

Format

public long remaining( )

Description

Returns the number of unread bytes remaining in the BFILE.

Parameters

None.

Return Value

This method returns the number of unread bytes in the BFILE.

Exceptions

None.

Examples

None.

reset( )

Format

public void reset( )

Description

Repositions the stream to the position of the last valid mark.

Parameters

None.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

seek(long)

Format

public void seek(long pos)

Description

Sets the offset from the beginning of the BFILE at which the next read operation should occur.

Parameters

pos
The offset from the beginning of the BFILE at which the next read operation should occur.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

skip(long)

Format

public long skip(long n)

Description

Attempts to skip over the specified number of bytes in the BFILE.

The number of bytes skipped may be smaller than the specified number; for example, the number would be smaller if the end of the file is reached.

Parameters

n
The number of bytes to be skipped.

Return Value

This method returns the number of bytes that are actually skipped.

Exceptions

java.io.IOException

Examples

None.

BlobInputStream Object

This section presents reference information on the methods associated with the BlobInputStream object, which provides an interface for JAI to read data from BLOBs. It is a subclass of com.sun.media.jai.codec.SeekableStream and java.io.InputStream; it implements java.io.DataInput.

Before running the methods associated with the BlobInputStream object, the following operations must have already been performed:

- The following import statements have been included:

```
import javax.media.jai.JAI;
import java.awt.image.RenderedImage;
```

- A local BlobInputStream object named inStream has been created.

BlobInputStream(BLOB)

Format

public BlobInputStream(oracle.sql.BLOB blob)

Description

Creates a BlobInputStream object that reads from the specified BLOB. The constructor uses an optimal chunk size that is determined by the database.

Parameters

blob
The BLOB from which data will be read.

Return Value

None.

Exceptions

java.io.IOException

java.sql.SQLException

Examples

None.

BlobInputStream(BLOB, int)

Format

public BlobInputStream(oracle.sql.BLOB blob, int chunkSize)

Description

Creates a BlobInputStream object that reads from the specified BLOB. The constructor uses the specified chunk size.

Parameters

blob
The BLOB from which data will be read.

chunkSize
The maximum amount of data to read from the BLOB at one time.

Return Value

None.

Exceptions

java.io.IOException

java.sql.SQLException

Examples

None.

canSeekBackwards( )

Format

public boolean canSeekBackwards( )

Description

Checks whether or not the stream can read backwards. Because the BlobInputStream object can read backwards, this method will always return true.

Parameters

None.

Return Value

This method returns true.

Exceptions

None.

Examples

None.

close( )

Format

public void close( )

Description

Closes the BlobInputStream object, releasing any resources being used.

Parameters

None.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

getBLOB( )

Format

public oracle.sql.BLOB getBLOB( )

Description

Returns the BLOB associated with the BlobInputStream object.

Parameters

None.

Return Value

This method returns the BLOB associated with the BlobInputStream object.

Exceptions

None.

Examples

None.

getFilePointer( )

Format

public long getFilePointer( )

Description

Returns the offset from the beginning of the BLOB at which the next read operation will occur.

Parameters

None.

Return Value

This method returns the offset from the beginning of the BLOB at which the next read operation will occur, in bytes.

Exceptions

java.io.IOException

Examples

None.

mark(int)

Format

public void mark(int readLimit)

Description

Marks the current position in the BlobInputStream object. A call to the reset( ) method will return you to the last marked position in the BlobInputStream object.

Parameters

readLimit
This argument is ignored by the class.

Return Value

None.

Exceptions

None.

Examples

None.

markSupported( )

Format

public boolean markSupported( )

Description

Checks whether or not the BlobInputStream object supports marking. Because the BlobInputStream object supports marking, this method will always return true.

Parameters

None.

Return Value

This method returns true.

Exceptions

None.

Examples

None.

read( )

Format

public int read( )

Description

Reads a single byte from the BLOB associated with the BlobInputStream object.

Parameters

None.

Return Value

This method returns the byte of data that is read, or -1 if the end of the BLOB has been reached.

Exceptions

java.io.IOException

Examples

None.

read(byte[ ])

Format

public int read(byte[ ] buffer)

Description

Reads data from the BLOB into the specified buffer.

Parameters

buffer
The buffer into which the data is read.

Return Value

This method returns the number of bytes read into the buffer, or -1 if the end of the BLOB was reached before any data was read. The value cannot exceed the length of the buffer.

Exceptions

java.io.IOException

Examples

None.

read(byte[ ], int, int)

Format

public int read(byte[ ]buffer, int off, int len)

Description

Reads up to the specified length of data from the BLOB into the specified buffer, starting from the specified offset.

Parameters

buffer
The buffer into which the data is read.

off
The offset from the beginning of the buffer at which data will be written, in bytes.

len
The maximum number of bytes to be written to the buffer.

Return Value

This method returns the number of bytes read into the buffer, or -1 if the end of the BLOB has been reached. The value cannot exceed the length of the buffer.

Exceptions

java.io.IOException

Examples

None.

remaining( )

Format

public long remaining( )

Description

Returns the number of unread bytes remaining in the BLOB.

Parameters

None.

Return Value

This method returns the number of unread bytes in the BLOB.

Exceptions

None.

Examples

None.

reset( )

Format

public void reset( )

Description

Repositions the stream to the position of the last valid mark.

Parameters

None.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

seek(long)

Format

public void seek(long pos)

Description

Sets the offset from the beginning of the BLOB at which the next read operation should occur.

Parameters

pos
The offset from the beginning of the BLOB at which the next read operation should occur.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

skip(long)

Format

public long skip(long n)

Description

Attempts to skip over the specified number of bytes in the BLOB.

The number of bytes skipped may be smaller than the specified number; for example, the number would be smaller if the end of the file is reached.

Parameters

n
The number of bytes to be skipped.

Return Value

This method returns the number of bytes that are actually skipped.

Exceptions

java.io.IOException

Examples

None.

BlobOutputStream Object

This section presents reference information on the methods associated with the BlobOutputStream object, which provides an interface for JAI to write data to BLOBs. It is a subclass of java.io.OutputStream.

Before running the methods associated with the BlobOutputStream object, the following operations must have already been performed:

- The following import statements have been included:

```
import javax.media.jai.JAI;
import java.awt.image.RenderedImage;
```

- A local BlobOutputStream object named outStream has been created.

BlobOutputStream(BLOB)

Format

public BlobOutputStream (oracle.sql.BLOB blob)

Description

Creates a BlobOutputStream object that writes to the specified BLOB, using an optimal chunk size that is determined by the database. Creating an object of this type implicitly trims the data in the BLOB to a length of zero.

Parameters

blob
The BLOB to which data will be written.

Return Value

None.

Exceptions

java.io.IOException

java.sql.SQLException

Examples

None.

BlobOutputStream(BLOB, int)

Format

public BlobOutputStream(oracle.sql.BLOB blob, int chunkSize)

Description

Creates a BlobOutputStream object that writes to the specified BLOB, using the given integer as the maximum chunk size. Creating an object of this type implicitly trims the data in the BLOB to a length of zero.

Parameters

blob
The BLOB to which data will be written

chunkSize
The maximum amount of data to write to the BLOB at one time.

Return Value

None.

Exceptions

java.io.IOException

java.sql.SQLException

Examples

None.

close( )

Format

public void close( )

Description

Closes the output stream and releases any system resources associated with this stream. Before closing the stream, this method automatically calls the flush( ) method to write any buffered bytes to the BLOB.

Parameters

None.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

flush( )

Format

public void flush( )

Description

Flushes the output stream and forces any buffered output bytes to be written to the BLOB.

Parameters

None.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

getFilePointer( )

Format

public long getFilePointer( )

Description

Returns the offset from the beginning of the BLOB at which the next write operation will occur.

Parameters

None.

Return Value

This method returns the offset from the beginning of the BLOB at which the next write operation will occur, in bytes.

Exceptions

java.io.IOException

Examples

None.

length( )

Format

public long length( )

Description

Returns the current length of the output stream.

Parameters

None.

Return Value

This method returns the current length of the output stream.

Exceptions

java.io.IOException

Examples

None.

seek(long)

Format

public void seek(long pos)

Description

Sets the file-pointer offset, measured from the beginning of this stream, at which the next write operation occurs.

The offset may be set beyond the end of the stream. Setting the offset beyond the end of the stream does not change the stream length; the stream length will change only by writing after the offset has been set beyond the end of the stream.

Parameters

pos
The offset position, measured in bytes from the beginning of the stream, at which to set the file pointer.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

write(byte[ ])

Format

public void write(byte[ ] buffer)

Description

Writes all bytes in the specified byte array to the BLOB.

Parameters

buffer
An array of bytes to be written to the BLOB.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

write(byte[ ], int, int)

Format

public void write(byte[ ] buffer, int off, int len)

Description

Writes the specified number of bytes from the specified byte array to the BLOB.

Parameters

buffer
The buffer containing the data to be written to the BLOB.

off
The starting position for the offset in the buffer.

len
The number of bytes to write to the BLOB.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

write(int)

Format

public void write(int b)

Description

Writes the specified byte to the BLOB.

Parameters

b
The byte to be written to the BLOB. Only the low-order byte is written; the upper 24 bits are ignored.

Return Value

None.

Exceptions

java.io.IOException

Examples

None.

9

Java Classes for Servlets and JSP

Oracle *inter*Media Java Classes ("*inter*Media Java Classes") for servlets and JavaServer Pages (JSP) provides the following classes to facilitate retrieving and uploading media data from and to Oracle Database:

- oracle.ord.im.OrdHttpJspResponseHandler (See OrdHttpJspResponseHandler Class.)
- oracle.ord.im.OrdHttpResponseHandler (See OrdHttpResponseHandler Class.)
- oracle.ord.im.OrdHttpUploadFile (See OrdHttpUploadFile Class.)
- oracle.ord.im.OrdHttpUploadFormData (See OrdHttpUploadFormData Class.)
- oracle.ord.im.OrdMultipartFilter (See OrdMultipartFilter Class.)
- oracle.ord.im.OrdMultipartWrapper (See OrdMultipartWrapper Class.)

The OrdHttpJspResponseHandler class facilitates the retrieval of media data from the database, and its delivery to a browser or other HTTP client from a JSP page. The OrdHttpResponseHandler class provides the same features for a Java servlet.

Form-based file uploading using HTML forms encodes form data and uploaded files in POST requests using the multipart/form-data format. The OrdHttpUploadFormData class facilitates the processing of such requests by parsing the POST data and making the contents of regular form fields and the contents of uploaded files readily accessible to Java servlets or JSP pages. The OrdHttpUploadFile class facilitates the handling of uploaded files by providing an API that applications can call to load media data into Oracle Database.

The OrdMultipartFilter class filters form-based file upload servlet requests whose contents are encoded in multipart/form-data format. This class wraps a request object with an OrdMultipartWrapper object that parses the content. When used with the OrdMultipartWrapper object, the OrdMultipartFilter class provides

transparent access to the parameters and files in the servlet request with multipart/form-data encoding.

The OrdMultipartWrapper class wraps the HttpServletRequest object and provides access to the contents in the HTTP request that is encoded using multipart/form-data encoding. This class provides access to the text-based form field parameters and the uploaded files.

The client library, `ordhttp.jar`, is located in *`$ORACLE_HOME`*`/ord/im/jlib` on UNIX and *`<ORACLE_HOME>`*`\ord\im\jlib` on Windows.

9.1 Prerequisites

In order to run *inter*Media methods for servlets and JSP pages, you will need to import classes from the `oracle.ord.im` package into your Java file.

You may also need to import classes from the following Java packages:

```
java.sql.
java.io.
javax.servlet.
javax.servlet.http.
oracle.jdbc.
oracle.sql.
```

OrdHttpJspResponseHandler Class

This section presents reference information on the methods of the oracle.ord.im.OrdHttpJspResponseHandler class.

The OrdHttpJspResponseHandler class facilitates the retrieval of media data from Oracle Database, and its delivery to a browser or another HTTP client from a JSP page.

This class inherits the DEFAULT_BUFFER_SIZE field from the OrdHttpResponseHandler class.

An Important Note on JSP Engines

JSP engines are not required to support access to the servlet binary output stream. Therefore, not all JSP engines support the delivery of media data using the OrdHttpJspResponseHandler class.

All media data stored in the database using the *inter*Media types, including text documents stored using the OrdDoc type, is stored using a binary LOB data type. Media data stored internally in the database is stored using a BLOB. Media data stored in an operating system file outside the database is stored using a BFILE. Therefore, all media data is delivered to the browser through the servlet binary output stream, using the ServletOutputStream class.

All the send methods in the OrdHttpJspResponseHandler class mirror the initial processing of the jsp:forward tag. Specifically, these send methods call the JspWriter clear( ) method to clear the output buffer of the page prior to obtaining the binary output stream. However, JSP engines are not required to support a call to the ServletResponse.getOutputStream method from within a JSP page. A JSP engine that does not support this typically throws an IllegalStateException error from the getOutputStream method. However, the exact behavior is implementation-specific.

If your JSP engine does not support access to the binary output stream from within a JSP page, then you must use a servlet to deliver media data. For example, perform one of the following operations:

- Use the jsp:forward tag to forward a multimedia retrieval request to a servlet.
- Construct multimedia retrieval URLs to retrieve the data directly from a servlet.

An Important Note on Return Statements

When delivering media data from a JSP page, a return statement is always required following a call to any of the send methods of the OrdHttpJspResponseHandler

class. The return statement is necessary to ensure that no other data is written to the output stream of the JSP page following the media data.

An `if ( true ) { ... return; }` construct may be used to avoid the "statement not reachable" error that may result from the presence of additional code, generated by the JSP engine, at the end of a compiled page. This construct, which mirrors exactly the code produced by some JSP engines to handle the `<jsp:forward ... >` directive, is shown in the example provided later in this section.

Note: An *inter*Media Java object, such as an OrdImage object, is not dependent on the JDBC statement or result set from which it was obtained. However, an *inter*Media Java object is dependent on the JDBC connection it is using and on which the SQL statement was executed or from which the result set was obtained. Therefore, having obtained an *inter*Media Java object from the database, an application must not release the JDBC connection before delivering the media data to the browser.

All the send methods in this class call the JspWriter clear( ) method to clear the page's output buffer prior to delivering the media. Therefore, the page must use the buffered output model, which is the default.

The following example demonstrates how to use the OrdHttpJspResponseHandler class to retrieve an image from the database and deliver it to a browser. The return statement ensures that the trailing newline characters following the final end tag (represented by a percent mark and right-angle bracket, or `%>`) are not transmitted to the browser as part of the image.

The `if ( true ) { ... return; }` construct is used to avoid the "statement not reachable" error that would otherwise be produced by this example due to the additional statements, generated by the JSP engine, at the end of the compiled page.

```
<%@ 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( )

Format

public OrdHttpJspResponseHandler( )

Description

Creates an OrdHttpJspResponseHandler object to handle the response to a multimedia retrieval request. The application must subsequently specify the PageContext object by calling the setPageContext( ) method.

The default constructor is typically invoked implicitly when the OrdHttpJspResponseHandler class is used as a JavaBean.

Parameters

None.

Return Value

None.

Exceptions

None.

Examples

None.

OrdHttpJspResponseHandler(PageContext)

Format

public OrdHttpJspResponseHandler(javax.servlet.jsp.PageContext pageContext)

Description

Creates an OrdHttpJspResponseHandler object to handle the response to a multimedia retrieval request, and specifies the PageContext object for the response handler.

Parameters

pageContext
An object of type PageContext.

Return Value

None.

Exceptions

None.

Examples

None.

sendAudio(OrdAudio)

Format

public void sendAudio(oracle.ord.im.OrdAudio media)

Description

Retrieves an audio clip from an OrdAudio object and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers. This method calls the JspWriter clear( ) method to clear the output buffer of the page prior to delivering the audio clip. Therefore, the page must use the buffered output model, which is the default.

This method extends the OrdHttpResponseHandler sendAudio(OrdAudio) method. See sendAudio(OrdAudio) in OrdHttpResponseHandler Class for information about this method in the base class.

Parameters

media
An object of type oracle.ord.im.OrdAudio.

Return Value

This method overrides the sendAudio(OrdAudio) method in class OrdHttpResponseHandler.

Exceptions

java.lang.IllegalStateException

This exception is thrown if PageContext has not been specified.

OrdHttpResponseException

This exception is thrown if the source type is not recognized.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendDoc(OrdDoc)

Format

public void sendDoc(oracle.ord.im.OrdDoc media)

Description

Retrieves media data from an OrdDoc object and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers. This method calls the JspWriter clear( ) method to clear the output buffer of the page prior to delivering the media. Therefore, the page must use the buffered output model, which is the default.

This method extends the OrdHttpResponseHandler sendDoc(OrdDoc) method. See sendDoc(OrdDoc) in OrdHttpResponseHandler Class for information about this method in the base class.

Parameters

media
An object of type oracle.ord.im.OrdDoc.

Return Value

This method overrides the sendDoc(OrdDoc) method in class OrdHttpResponseHandler.

Exceptions

java.lang.IllegalStateException

This exception is thrown if PageContext has not been specified.

OrdHttpResponseException

This exception is thrown if the source type is not recognized.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendImage(OrdImage)

Format

public void sendImage(oracle.ord.im.OrdImage media)

Description

Retrieves an image from an OrdImage object and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers. This method calls the JspWriter clear( ) method to clear the output buffer of the page prior to delivering the media. Therefore, the page must use the buffered output model, which is the default.

This method extends the OrdHttpResponseHandler sendImage(OrdImage) method. See sendImage(OrdImage) in OrdHttpResponseHandler Class for information about this method in the base class.

Parameters

media
An object of type oracle.ord.im.OrdImage.

Return Value

This method overrides the sendImage(OrdImage) method in class OrdHttpResponseHandler.

Exceptions

java.lang.IllegalStateException

This exception is thrown if PageContext has not been specified.

OrdHttpResponseException

This exception is thrown if the source type is not recognized.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendResponse(String, int, BFILE, Timestamp)

Format

```
public void sendResponse(String contentType, int length, oracle.sql.BFILE bfile,
                         java.sql.Timestamp lastModified)
```

Description

Builds the HTTP response header, then retrieves the contents of the BFILE from the database and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers. This method calls the JspWriter clear( ) method to clear the output buffer of the page prior to delivering the media. Therefore, the page must use the buffered output model, which is the default.

This method extends the OrdHttpResponseHandler sendResponse(String, int, BFILE, Timestamp) method. See sendResponse(String, int, BFILE, Timestamp) in OrdHttpResponseHandler Class for information about this method in the base class.

Parameters

contentType
A String that specifies the MIME type of the content.

length
An integer that specifies the length of the data.

bfile
An object of type oracle.sql.BFILE from which the media data is retrieved.

lastModified
A java.sql.Timestamp object that specifies the date and time when the data was last modified, or null if no last modified date and time are available.

Return Value

This method overrides the sendResponse(String, int, BFILE, Timestamp) method in class OrdHttpResponseHandler.

Exceptions

java.lang.IllegalStateException

This exception is thrown if PageContext has not been specified.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

java.lang.IllegalArgumentException

This exception is thrown if the length is negative.

Examples

None.

sendResponse(String, int, BLOB, Timestamp)

Format

```
public void sendResponse(String contentType, int length, oracle.sql.BLOB blob,
                         java.sql.Timestamp lastModified)
```

Description

Builds the HTTP response header, then retrieves the contents of the BLOB from the database and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers. This method calls the JspWriter clear( ) method to clear the output buffer of the page prior to delivering the media. Therefore, the page must use the buffered output model, which is the default.

This method extends the OrdHttpResponseHandler.sendResponse(String, int, BLOB, Timestamp) method. See sendResponse(String, int, BLOB, Timestamp) in OrdHttpResponseHandler Class for information about this method in the base class.

Parameters

contentType
A String that specifies the MIME type of the content.

length
An integer that specifies the length of the data.

blob
An object of type oracle.sql.BLOB from which the media data is retrieved.

lastModified
A java.sql.Timestamp object that specifies the date and time when the data was last modified, or null if no last modified date and time are available.

Return Value

This method overrides the sendResponse(String, int, BLOB, Timestamp) method in class OrdHttpResponseHandler.

Exceptions

java.lang.IllegalStateException

This exception is thrown if PageContext has not been specified.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

java.lang.IllegalArgumentException

This exception is thrown if the length is negative.

Examples

None.

sendResponse(String, int, InputStream, Timestamp)

Format

```
public void sendResponse(String contentType, int length, java.io.InputStream in,
                         java.sql.Timestamp lastModified)
```

Description

Builds the HTTP response header, then retrieves the contents of the InputStream object and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers. This method calls the JspWriter clear( ) method to clear the output buffer of the page prior to delivering the media. Therefore, the page must use the buffered output model, which is the default.

This method extends the OrdHttpResponseHandler sendResponse(String, int, InputStream, Timestamp) method. See sendResponse(String, int, InputStream, Timestamp) in OrdHttpResponseHandler Class for information about this method in the base class.

Parameters

contentType
A String that specifies the MIME type of the content.

length
An integer that specifies the length of the data.

in
An InputStream object from which the media data is retrieved.

lastModified
A java.sql.Timestamp object that specifies the date and time when the data was last modified, or null if no last modified date and time are available.

Return Value

This method overrides the sendResponse(String, int, InputStream, Timestamp) method in class OrdHttpResponseHandler.

Exceptions

java.lang.IllegalStateException

This exception is thrown if PageContext has not been specified.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

java.lang.IllegalArgumentException

This exception is thrown if the length is negative.

Examples

None.

sendVideo(OrdVideo)

Format

public void sendVideo(oracle.ord.im.OrdVideo media)

Description

Retrieves a video clip from an OrdVideo object and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers. This method calls the JspWriter clear( ) method to clear the output buffer of the page prior to delivering the video clip. Therefore, the page must use the buffered output model, which is the default.

This method extends the OrdHttpResponseHandler sendVideo(OrdVideo) method. See sendVideo(OrdVideo) in OrdHttpResponseHandler Class for information about this method in the base class.

Parameters

media
An object of type oracle.ord.im.OrdVideo.

Return Value

This method overrides the sendVideo(OrdVideo) method in class OrdHttpResponseHandler.

Exceptions

java.lang.IllegalStateException

This exception is thrown if PageContext has not been specified.

OrdHttpResponseException

This exception is thrown if the source type is not recognized.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

setPageContext(PageContext)

Format

public void setPageContext(javax.servlet.jsp.PageContext pageContext)

Description

Specifies the PageContext object for this response handler. You must call this method before calling any of the send methods if you did not specify the PageContext object in the constructor.

Parameters

pageContext
An object of type PageContext.

Return Value

None.

Exceptions

None.

Examples

None.

OrdHttpResponseHandler Class

This section presents reference information on the methods of the oracle.ord.im.OrdHttpResponseHandler class.

The OrdHttpResponseHandler class facilitates the retrieval of media data from Oracle Database, and its delivery to a browser or other HTTP client from a Java servlet.

An *inter*Media Java object, such as an OrdImage object, is not dependent on the JDBC statement or result set from which it was obtained. However, an *inter*Media Java object is dependent on the JDBC connection it is using and on which the SQL statement was executed, or from which the result set was obtained. Therefore, having obtained an *inter*Media Java object from the database, an application must not release the JDBC connection before delivering the media data to the browser.

This class contains the following field:

- public static final int DEFAULT_BUFFER_SIZE

  The OrdHttpResponseHandler class uses a default buffer size of 32768 to retrieve LOB data from the database and deliver it to the client. You can override this value with the setBufferSize( ) method.

The following example shows how to use the OrdHttpResponseHandler class to retrieve an image from a database and deliver it to a browser:

```
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.getORAData(1, OrdImage.getORADataFactory(
));
     OrdHttpResponseHandler handler = new OrdHttpResponseHandler(request,
          response);
     handler.sendImage(media);
}
else{
     response.setStatus(response.SC_NOT_FOUND);
}
rset.close( );
stmt.close( );
```

A Note on the Use of Charsets Other Than ISO-8859-1 (Latin-1)

If you wish to retrieve from an OrdDoc object a text-based document with a character set (charset) other than ISO-8859-1 (also called Latin-1), and deliver that document to a browser, your application must specify the charset name in the HTTP Content-Type header.

If the charset specification is included in the MIME type attribute in the OrdDoc object, then your application needs to call only the sendDoc( ) method to retrieve the document and deliver it to the browser. For example, an HTML page that is written in Japanese might be stored in the OrdDoc object with a MIME type of `text/html; charset=Shift_JIS`. In this case, calling the sendDoc( ) method will send the appropriate Content-Type header, allowing the browser to display the page correctly.

However, if the MIME type in the OrdDoc object does not include the charset specification, then you must call one of the sendResponse( ) methods and specify the MIME type explicitly. For example, if the MIME type of an HTML page written in Japanese is stored in the OrdDoc object as `text/html`, and the charset name is specified in a separate column, then the application must append the charset specification to the MIME type before calling a sendResponse( ) method. For example:

```
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.getORAData(1, OrdDoc.getORADataFactory( ));
     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( )

Format

public OrdHttpResponseHandler( )

Description

Creates an OrdHttpResponseHandler object to handle the response to a multimedia retrieval request. The application must subsequently specify the HttpServletResponse object by calling the setServletResponse( ) method, and can optionally specify the HttpServletRequest object by calling the setServletRequest( ) method.

Parameters

None.

Return Value

None.

Exceptions

None.

Examples

None.

OrdHttpResponseHandler(HttpServletRequest, HttpServletResponse)

Format

```
public OrdHttpResponseHandler(javax.servlet.http.HttpServletRequest request,
                              javax.servlet.http.HttpServletResponse response)
```

Description

Creates an OrdHttpResponseHandler object to handle the response to a multimedia retrieval request and specifies the HttpServletRequest and HttpServletResponse objects for the response handler.

Parameters

request
An object of type HttpServletRequest.

response
An object of type HttpServletResponse.

Return Value

None.

Exceptions

None.

Examples

None.

sendAudio(OrdAudio)

Format

public void sendAudio(oracle.ord.im.OrdAudio media)

Description

Retrieves an audio clip from an OrdAudio object and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers.

Parameters

media
An object of type oracle.ord.im.OrdAudio.

Return Value

None.

Exceptions

java.lang.IllegalStateException

This exception is thrown if HttpServletRequest or HttpServletResponse has not been specified.

OrdHttpResponseException

This exception is thrown if the source type is not recognized.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendDoc(OrdDoc)

Format

public void sendDoc(oracle.ord.im.OrdDoc media)

Description

Retrieves media data from an OrdDoc object and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers.

Parameters

media
An object of type oracle.ord.im.OrdDoc.

Return Value

None.

Exceptions

java.lang.IllegalStateException

This exception is thrown if HttpServletRequest or HttpServletResponse has not been specified.

OrdHttpResponseException

This exception is thrown if the source type is not recognized.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendImage(OrdImage)

Format

public void sendImage(oracle.ord.im.OrdImage media)

Description

Retrieves an image from an OrdImage object and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers.

Parameters

media
An object of type oracle.ord.im.OrdImage.

Return Value

None.

Exceptions

java.lang.IllegalStateException

This exception is thrown if HttpServletRequest or HttpServletResponse has not been specified.

OrdHttpResponseException

This exception is thrown if the source type is not recognized.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendResponse( )

Format

public void sendResponse( )

Description

Retrieves a media object from one of the *inter*Media objects (OrdImage, OrdAudio, OrdVideo, or OrdDoc) and delivers it to the browser. The media object to be delivered is determined by the last setMedia( ) method.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers.

Parameters

None.

Return Value

None.

Exceptions

java.io.IOException

This exception is thrown if an error occurs reading the media data.

java.lang.IllegalStateException

This exception is thrown if HttpServletRequest or HttpServletResponse has not been specified.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

OrdHttpResponseException

This exception is thrown if the source type is not recognized.

Examples

None.

sendResponse(String, int, BFILE, Timestamp)

Format

```
public void sendResponse(String contentType, int length, oracle.sql.BFILE bfile,
                         java.sql.Timestamp lastModified)
```

Description

Builds the HTTP response header, then retrieves the contents of the BFILE from the database and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers.

Parameters

contentType
A string that specifies the MIME type of the content.

length
An integer that specifies the length of the data.

bfile
An object of type oracle.sql.BFILE from which the media data is retrieved.

lastModified
A java.sql.Timestamp object that specifies the date and time when the data was last modified, or null if no last modified date and time are available.

Return Value

None.

Exceptions

java.lang.IllegalStateException

This exception is thrown if HttpServletRequest or HttpServletResponse has not been specified.

java.lang.IllegalArgumentException

This exception is thrown if the length is negative.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendResponse(String, int, BLOB, Timestamp)

Format

```
public void sendResponse(String contentType, int length, oracle.sql.BLOB blob,
                         java.sql.Timestamp lastModified)
```

Description

Builds the HTTP response header, then retrieves the contents of the BLOB from the database and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers.

Parameters

contentType
A string that specifies the MIME type of the content.

length
An integer that specifies the length of the data.

blob
An object of type oracle.sql.BLOB from which the media data is retrieved.

lastModified
A java.sql.Timestamp object that specifies the date and time when the data was last modified, or null if no last modified date and time are available.

Return Value

None.

Exceptions

java.lang.IllegalStateException

This exception is thrown if HttpServletRequest or HttpServletResponse has not been specified.

java.lang.IllegalArgumentException

This exception is thrown if the length is negative.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendResponse(String, int, InputStream, Timestamp)

Format

```
public void sendResponse(String contentType, int length, java.io.InputStream in,
                         java.sql.Timestamp lastModified)
```

Description

Builds the HTTP response header, then retrieves the contents of the InputStream object and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers.

Parameters

contentType
A String that specifies the MIME type of the content.

length
An integer that specifies the length of the data.

in
An InputStream object from which the media data is retrieved.

lastModified
A java.sql.Timestamp object that specifies the date and time when the data was last modified, or null if no last modified date and time are available.

Return Value

None.

Exceptions

java.lang.IllegalStateException

This exception is thrown if HttpServletRequest or HttpServletResponse has not been specified.

java.lang.IllegalArgumentException

This exception is thrown if the length is negative.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendResponseBody(int, BFILE)

Format

public void sendResponseBody(int length, oracle.sql.BFILE bfile)

Description

Retrieves the content of a BFILE from the database and delivers it as the response body to the browser. The caller is responsible for building the HTTP header.

Parameters

length
An integer that specifies the length of the data.

bfile
An object of type oracle.sql.BFILE from which the media data is retrieved.

Return Value

None.

Exceptions

java.lang.IllegalStateException

This exception is thrown if HttpServletResponse has not been specified.

java.lang.IllegalArgumentException

This exception is thrown if the length is negative.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendResponseBody(int, BLOB)

Format

public void sendResponseBody(int length, oracle.sql.BLOB blob)

Description

Retrieves the content of a BLOB from the database and delivers it as the response body to the browser. The caller is responsible for building the HTTP header.

Parameters

length
An integer that specifies the length of the data.

blob
An object of type oracle.sql.BLOB from which the media data is retrieved.

Return Value

None.

Exceptions

java.lang.IllegalStateException

This exception is thrown if HttpServletResponse has not been specified.

java.lang.IllegalArgumentException

This exception is thrown if the length is negative.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendResponseBody(int, InputStream)

Format

public void sendResponseBody(int length, java.io.InputStream in)

Description

Retrieves the content of the InputStream object and delivers it to the client. The caller is responsible for building the HTTP header.

Parameters

length
An integer that specifies the length of the data.

in
An InputStream object from which the media data is retrieved.

Return Value

None.

Exceptions

java.lang.IllegalStateException

This exception is thrown if HttpServletResponse has not been specified.

java.lang.IllegalArgumentException

This exception is thrown if the length is negative.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

sendVideo(OrdVideo)

Format

public void sendVideo(oracle.ord.im.OrdVideo media)

Description

Retrieves a video clip from an OrdVideo object and delivers it to the browser.

This method supports browser content caching by supporting the If-Modified-Since and Last-Modified headers.

Parameters

media
An object of type oracle.ord.im.OrdVideo.

Return Value

None.

Exceptions

java.lang.IllegalStateException

This exception is thrown if HttpServletRequest or HttpServletResponse has not been specified.

OrdHttpResponseException

This exception is thrown if the source type is not recognized.

javax.servlet.ServletException

This exception is thrown if an error occurs accessing the binary output stream.

java.sql.SQLException

This exception is thrown if an error occurs obtaining an InputStream object to read the media data.

java.io.IOException

This exception is thrown if an error occurs reading the media data.

Examples

None.

setBufferSize(int)

Format

public void setBufferSize(int bufferSize)

Description

Sets the buffer size for LOB read and HTTP response write operations.

Parameters

bufferSize
An integer that specifies the buffer size.

Return Value

None.

Exceptions

java.lang.IllegalArgumentException

This exception is thrown if the buffer size is negative or zero.

Examples

None.

setEncodeHtml(boolean)

Format

public void setEncodeHtml(boolean doEncode)

Description

Enables the encoding of special characters (`! " % & ' ( ) ; < >`) from the response output stream with MIME type text or html. By default, there is no encoding.

Parameters

doEncode
A Boolean value indicating whether or not to encode the response output stream.

Return Value

None.

Exceptions

None.

Examples

None.

setHeader(String, String)

Format

public void setHeader(String name, String value)

Description

Sets the HTTP response header with the String value. If the header is already set, the new value overwrites the previous value.

Parameters

name
The name of the header.

value
The value of the header.

Return Value

None.

Exceptions

None.

Examples

None.

setHeader(String, long)

Format

public void setHeader(String name, long date)

Description

Sets the HTTP response header with the date value. If the header is already set, the new value overwrites the previous value.

Parameters

name
The name of the header.

date
The date value of the header.

Return Value

None.

Exceptions

None.

Examples

None.

setHeader(String, int)

Format

public void setHeader(java.lang.String name, int value)

Description

Sets the HTTP response header with the integer value. If the header is already set, the new value overwrites the previous value.

Parameters

name
The name of the header.

value
The integer value of the header.

Return Value

None.

Exceptions

None.

Examples

None.

setMedia(OrdAudio)

Format

public void setMedia(OrdAudio media)

Description

Sets the media object to be delivered. To deliver the data in the response body, this method must be called before calling the sendResponse( ) method.

Parameters

media
An object of type oracle.ord.im.OrdAudio.

Return Value

None.

Exceptions

None.

Examples

None.

setMedia(OrdDoc)

Format

public void setMedia(OrdDoc media)

Description

Sets the media object to be delivered. To deliver the data in the response body, this method must be called before calling the sendResponse( ) method.

Parameters

media
An object of type oracle.ord.im.OrdDoc.

Return Value

None.

Exceptions

None.

Examples

None.

setMedia(OrdImage)

Format

public void setMedia(OrdImage media)

Description

Sets the media object to be delivered. To deliver the data in the response body, this method must be called before calling the sendResponse( ) method.

Parameters

media
An object of type oracle.ord.im.OrdImage.

Return Value

None.

Exceptions

None.

Examples

None.

setMedia(OrdVideo)

Format

public void setMedia(OrdVideo media)

Description

Sets the media object to be delivered. To deliver the data in the response body, this method must be called before calling the sendResponse( ) method.

Parameters

media
An object of type oracle.ord.im.OrdVideo.

Return Value

None.

Exceptions

None.

Examples

None.

setServletRequest(HttpServletRequest)

Format

public void setServletRequest(javax.servlet.http.HttpServletRequest request)

Description

Specifies the HttpServletRequest object for this response handler. You must call this method if you did not specify the HttpServletRequest object in the constructor and you want to call any of the send methods other than the sendResponseBody methods. You do not need to call this method if you call only the sendResponseBody methods.

Parameters

request
An object of type HttpServletRequest.

Return Value

None.

Exceptions

None.

Examples

None.

setServletResponse(HttpServletResponse)

Format

public void setServletResponse(javax.servlet.http.HttpServletResponse response)

Description

Specifies the HttpServletResponse object for this response handler. You must call this method before calling any of the send methods if you did not specify the HttpServletResponse object in the constructor.

Parameters

response
An object of type HttpServletResponse.

Return Value

None.

Exceptions

None.

Examples

None.

OrdHttpUploadFile Class

This section presents reference information on the methods of the oracle.ord.im.OrdHttpUploadFile class.

Form-based file uploading using HTML forms encodes form data and uploaded files in POST requests using the multipart/form-data format. The OrdHttpUploadFile class is used to represent an uploaded file that has been parsed by the OrdHttpUploadFormData class (see OrdHttpUploadFormData Class for more information on the OrdHttpUploadFormData class). The OrdHttpUploadFile class provides methods to obtain information about the uploaded file, to access the contents of the file directly, and to facilitate loading the contents into an *inter*Media object in a database.

Every input field of type FILE in an HTML form will produce a parameter of type OrdHttpUploadFile, whether or not a user enters a valid file name into such a field. Depending on the requirements, applications can test the length of the file name, the length of the content, or both to determine if a valid file name was entered by a user and if the file was successfully uploaded by the browser. For example, if a user does not enter a file name, the getOriginalFileName( ) method will return the length of the file name as zero. However, if a user enters either an invalid file name or the name of an empty (zero-length) file, the getOriginalFileName( ) method will return the length of the file name, which will not be zero, and the getContentLength( ) method will return the content length of the file as zero.

getContentLength( )

Format

public int getContentLength( )

Description

Returns the length of the uploaded media file. If you enter an invalid file name, the name of a nonexistent file, or the name of an empty file, the length returned is zero.

Parameters

None.

Return Value

This method returns the length of the uploaded file.

Exceptions

None.

Examples

None.

getInputStream( )

Format

public java.io.InputStream getInputStream( )

Description

Returns an InputStream object that can be used to read uploaded data directly. Applications should close the stream with the close( ) method when finished.

Parameters

None.

Return Value

This method returns an InputStream object, from which to read the contents.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the uploaded file is no longer available (for example: because it has been released).

java.io.IOException

This exception is thrown if an error occurs opening the temporary file.

Examples

None.

getMimeType( )

Format

public String getMimeType( )

Description

Returns the MIME type of the media file, as determined by the browser when the file is uploaded.

Some browsers return a default MIME type even if you do not supply a file name; therefore, the application should check the file name or content length to ensure the file was uploaded successfully.

Parameters

None.

Return Value

This method returns the MIME type of the file, as a String.

Exceptions

None.

Examples

None.

getOriginalFileName( )

Format

public String getOriginalFileName( )

Description

Returns the original file name, as provided by the browser. If you do not supply a file name, an empty String is returned.

Parameters

None.

Return Value

This method returns the file name, as a String.

Exceptions

None.

Examples

None.

getSimpleFileName( )

Format

public String getSimpleFileName( )

Description

Returns the simple file name (that is, the name of the file and the extension). If you do not supply a file name, an empty String is returned.

Parameters

None.

Return Value

This method returns the simple file name, as a String.

Exceptions

None.

Examples

None.

loadAudio(OrdAudio)

Format

public void loadAudio(oracle.ord.im.OrdAudio media)

Description

Loads the uploaded file into an OrdAudio Java object and sets the properties based on the audio data. This method loads the audio data into the database and calls the OrdAudio setProperties( ) method to set the properties, such as the MIME type. This method does not use any existing format plug-in context information and does not set any comments when setting the properties. To use this method, the application fetches an initialized OrdAudio object from the database, calls this method to load the audio data into the database, and then updates the OrdAudio object in the database.

If the call to the setProperties( ) method fails because the audio format is not recognized, this method sets the following properties:

- MIME type (to the value specified by the browser)
- update time (to the current date and time)

Parameters

media
An object of type oracle.ord.im.OrdAudio into which the audio data will be loaded.

Return Value

None.

Exceptions

java.io.IOException

This exception is thrown if an error occurs while reading or writing the media data.

java.sql.SQLException

This exception is thrown if an unrecognized error occurs while storing the media data.

java.lang.IllegalStateException

This exception is thrown if the uploaded file is no longer available (for example: because it has been released).

Examples

None.

loadAudio(OrdAudio, byte[ ][ ], boolean)

Format

public void loadAudio(oracle.ord.im.OrdAudio media, byte[ ][ ] ctx, boolean setComments)

Description

Loads the uploaded file into an OrdAudio Java object and sets the properties using an application-supplied, format plug-in context. This method loads the audio data into the database and calls the OrdAudio setProperties( ) method to set the properties, such as the MIME type. The application provides the format plug-in context information and chooses whether or not to set the comments in the OrdAudio object. To use this method, the application fetches an initialized OrdAudio object from the database, calls this method to load the audio data into the database, and then updates the OrdAudio object in the database.

If the call to the setProperties( ) method fails because the audio format is not recognized, this method sets the following properties:

- MIME type (to the value specified by the browser)
- update time (to the current date and time)

Parameters

media
An object of type oracle.ord.im.OrdAudio into which the audio data will be loaded.

ctx
The format plug-in context information.

setComments
A Boolean value indicating whether or not to set the comments in the OrdAudio object.

Return Value

None.

Exceptions

java.io.IOException

This exception is thrown if an error occurs while reading or writing the media data.

java.sql.SQLException

This exception is thrown if an unrecognized error occurs while storing the media data.

java.lang.IllegalStateException

This exception is thrown if the uploaded file is no longer available (for example: because it has been released).

Examples

None.

loadBlob(BLOB)

Format

public void loadBlob(oracle.sql.BLOB blob)

Description

Loads the uploaded media file into a BLOB.

Parameters

blob
An object of type oracle.sql.BLOB into which the data will be loaded.

Return Value

None.

Exceptions

java.io.IOException

This exception is thrown if an error occurs while reading or writing the media data.

java.sql.SQLException

This exception is thrown if an unrecognized error occurs while storing the media data.

java.lang.IllegalStateException

This exception is thrown if the uploaded file is no longer available (for example: because it has been released).

Examples

None.

loadDoc(OrdDoc)

Format

public void loadDoc(oracle.ord.im.OrdDoc media)

Description

Loads the uploaded file into an OrdDoc Java object and sets the properties based on the contents of the document. This method loads the document into the database and calls the OrdDoc setProperties( ) method to set the properties, such as the MIME type. This method does not use any existing format plug-in context information and does not set any comments when setting the properties. To use this method, the application fetches an initialized OrdDoc object from the database, calls this method to load the document into the database, and then updates the OrdDoc object in the database.

If the call to the setProperties( ) method fails because the document format is not recognized, this method sets the following properties:

- MIME type (to the value specified by the browser)
- content length (to the length of the uploaded file)
- update time (to the current date and time)

Parameters

media
An object of type oracle.ord.im.OrdDoc into which the document will be loaded.

Return Value

None.

Exceptions

java.io.IOException

This exception is thrown if an error occurs while reading or writing the media data.

java.sql.SQLException

This exception is thrown if an unrecognized error occurs while storing the media data.

java.lang.IllegalStateException

This exception is thrown if the uploaded file is no longer available (for example: because it has been released).

Examples

None.

loadDoc(OrdDoc, byte[ ][ ], boolean)

Format

public void loadDoc(oracle.ord.im.OrdDoc media, byte[ ][ ] ctx, boolean setComments)

Description

Loads the uploaded file into an OrdDoc Java object and sets the properties using an application-supplied, format plug-in context. This method loads the document into the database and calls the OrdDoc setProperties( ) method to set the properties, such as the MIME type. The application provides the format plug-in context information and chooses whether or not to set the comments in the OrdDoc object. To use this method, the application fetches an initialized OrdDoc object from the database, calls this method to load the document into the database, and then updates the OrdDoc object in the database.

If the call to the setProperties( ) method fails because the document format is not recognized, this method sets the following properties:

- MIME type (to the value specified by the browser)
- content length (to the length of the uploaded file)
- update time (to the current date and time)

Parameters

media
An object of type oracle.ord.im.OrdDoc object which the document will be loaded.

ctx
The format plug-in context information.

setComments
A Boolean value indicating whether or not to set the comments in the object.

Return Value

None.

Exceptions

java.io.IOException

This exception is thrown if an error occurs while reading or writing the media data.

java.sql.SQLException

This exception is thrown if an unrecognized error occurs while storing the media data.

java.lang.IllegalStateException

This exception is thrown if the uploaded file is no longer available (for example: because it has been released).

Examples

None.

loadImage(OrdImage)

Format

public void loadImage(oracle.ord.im.OrdImage media)

Description

Loads the uploaded file into an OrdImage Java object and sets the properties based on the image contents. This method loads the image content into the database and calls the OrdImage setProperties( ) method to set the image's properties, such as the MIME type, length, height, and width. To use this method, the application fetches an initialized OrdImage object from the database, calls this method to load the image content into the database, and then updates the OrdImage object in the database.

If the call to the setProperties( ) method fails because the image format is not recognized, this method sets the following properties:

- MIME type (to the value specified by the browser)
- content length (to the length of the uploaded file)
- update time (to the current date and time)

Parameters

media
An object of type oracle.ord.im.OrdImage into which the image data will be loaded.

Return Value

None.

Exceptions

java.io.IOException

This exception is thrown if an error occurs while reading or writing the media data.

java.sql.SQLException

This exception is thrown if an unrecognized error occurs while storing the media data.

java.lang.IllegalStateException

This exception is thrown if the uploaded file is no longer available (for example: because it has been released).

Examples

None.

loadImage(OrdImage, String)

Format

public void loadImage(oracle.ord.im.OrdImage media, String cmd)

Description

Loads the uploaded file into an OrdImage Java object and sets the properties according to an application-supplied command string. To use this method, the application fetches an initialized OrdImage object from the database, calls this method to load the image content into the database, and then updates the OrdImage object in the database.

Parameters

media
An object of type oracle.ord.im.OrdImage into which the uploaded image data will be loaded.

cmd
A String that specifies the properties to be set.

Return Value

None.

Exceptions

java.io.IOException

This exception is thrown if an error occurs while reading or writing the media data.

java.sql.SQLException

This exception is thrown if an unrecognized error occurs while storing the media data.

java.lang.IllegalStateException

This exception is thrown if the uploaded file is no longer available (for example: because it has been released).

Examples

None.

loadVideo(OrdVideo)

Format

public void loadVideo(oracle.ord.im.OrdVideo media)

Description

Loads the uploaded file into an OrdVideo Java object and sets the properties based on the video data. This method loads the video data into the database and calls the OrdVideo setProperties( ) method to set the properties, such as the MIME type. This method does not use any existing format plug-in context information and does not set any comments when setting the properties. To use this method, the application fetches an initialized OrdVideo object from the database, calls this method to load the video data into the database, and then updates the OrdVideo object in the database.

If the call to the setProperties( ) method fails because the video format is not recognized, this method sets the following properties:

- MIME type (to the value specified by the browser)
- update time (to the current date and time)

Parameters

media
An object of type oracle.ord.im.OrdVideo into which the uploaded video data will be loaded.

Return Value

None.

Exceptions

java.io.IOException

This exception is thrown if an error occurs while reading or writing the media data.

java.sql.SQLException

This exception is thrown if an unrecognized error occurs while storing the media data.

java.lang.IllegalStateException

This exception is thrown if the uploaded file is no longer available (for example: because it has been released).

Examples

None.

loadVideo(OrdVideo, byte[ ][ ], boolean)

Format

public void loadVideo(oracle.ord.im.OrdVideo media, byte[ ][ ] ctx, boolean setComments)

Description

Loads the uploaded file into an OrdVideo Java object and sets the properties using an application-supplied, format plug-in context. This method loads the video data into the database and calls the OrdVideo setProperties( ) method to set the properties, such as the MIME type. The application provides the format plug-in context information and chooses whether or not to set the comments in the OrdVideo object. To use this method, the application fetches an initialized OrdVideo object from the database, calls this method to load the video data into the database, and then updates the OrdVideo object in the database.

If the call to the setProperties( ) method fails because the video format is not recognized, this method sets the following properties:

- MIME type (to the value specified by the browser)
- update time (to the current date and time)

Parameters

media
An object of type oracle.ord.im.OrdVideo into which the uploaded video data will be loaded.

ctx
The format plug-in context information.

setComments
A Boolean value indicating whether or not to set the comments in the object.

Return Value

None.

Exceptions

java.io.IOException

This exception is thrown if an error occurs while reading or writing the media data.

java.sql.SQLException

This exception is thrown if an unrecognized error occurs while storing the media data.

java.lang.IllegalStateException

This exception is thrown if the uploaded file is no longer available (for example: because it has been released).

Examples

None.

release( )

Format

public void release( )

Description

Releases all resources held by an OrdHttpUploadFile object. Specifically, this method releases the memory used to hold the contents of an uploaded file or deletes the temporary file used to hold the contents of the uploaded file. An application can optimize memory usage by calling this method to release any allocated memory, making it a candidate for garbage collection, after the application has finished processing an uploaded file.

Parameters

None.

Return Value

None.

Exceptions

None.

Examples

None.

OrdHttpUploadFormData Class

This section presents reference information on the methods of the oracle.ord.im.OrdHttpUploadFormData class.

File uploading using HTML forms encodes form data and uploaded files in POST requests using the multipart/form-data format. The OrdHttpUploadFormData class facilitates the processing of such requests by parsing the POST data and making the contents of regular form fields and uploaded files readily accessible to a Java servlet or JSP page. The OrdHttpUploadFormData class provides methods to access text-based form field parameters that are identical to the getParameter( ), getParameterValues( ), and getParameterNames( ) methods provided by the ServletRequest class. Access to uploaded files is provided by a similar set of methods, namely getFileParameter( ), getFileParameterValues( ), and getFileParameterNames( ). The OrdHttpUploadFile objects returned by the getFileParameter( ) and getFileParameterValues( ) methods provide simple access to the MIME type, length, and contents of each uploaded file.

For more information on the OrdHttpUploadFile class, see OrdHttpUploadFile Class.

The following is an example of how to use the OrdHttpUploadFormData class:

```
// 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.getORAData(1, OrdImage.getORADataFactory( ));

// 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.setORAData(1, media);
stmt.setString(2 id);
stmt.execute( );
stmt.close( );

// Commit the changes.
//
conn.commit( );
```

A Note on the Handling of Query String Parameters and Text-Based HTML Form Field Parameters

Every parameter in the optional query string of a request produces a corresponding parameter of type String, whether or not any data is associated with the parameter name. Likewise, every text-based input field in an HTML form also produces a corresponding parameter of type String, whether or not any data is entered into a field. When processing query string parameters and text-based input fields,

applications can test the length of the corresponding String object to determine if any data is present.

The parseFormData( ) method merges all query string and form field parameters into a single set of ordered parameters, where the query string parameters are processed first, followed by the form field parameters. Thus, query string parameters take precedence over form field parameters. For example, if a request is made with a query string of arg=hello&arg=world and the values 'greetings' and 'everyone' are entered into two HTML form fields named 'arg,' then the resulting parameter set would include the following entry: arg=(hello, world, greetings, everyone).

A Note on the Handling of FILE-Type HTML Form Field Parameters

Every input field of type FILE in an HTML form produces a corresponding parameter of type OrdUploadFile, whether or not a valid file name is entered into the field. When processing a field of type FILE, applications can test either the length of the file name, the length of content, or a combination of the two to determine if a valid file name was entered by a user, and if the file was successfully uploaded by the browser. See OrdHttpUploadFile Class for more information.

A Note on the Use of Non-Western European Languages

The Microsoft Internet Explorer browser lets data be entered into an HTML form using a character set encoding that is different from that being used to view the form. For example, it is possible to copy Japanese Shift_JIS character set data from one browser window and paste it into a form being viewed using the Western European (ISO) character set in a different browser window. In this situation, Internet Explorer can sometimes transmit the form data twice in such a way that the multipart/form-data parser cannot detect the duplicated data. Furthermore, the non-Western European language form data is sometimes sent as a Unicode escape sequence, sometimes in its raw binary form, and sometimes duplicated using both formats in different portions of the POST data.

Although this same problem does not exist with the Netscape browser, care must still be taken to ensure that the correct character set is being used. For example, although it is possible to copy Japanese Shift_JIS character set data from one browser window and paste it into a form being viewed using the Western European (ISO) character set in a different browser window, when the data is pasted into the form field, the two bytes that comprise each Japanese Shift_JIS character are stored as two individual Western European (ISO) characters.

Therefore, care must be taken to view an HTML form using the correct character set, no matter which Web browser is used. For example, the HTML META tag can be used to specify the character set as follows:

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=Shift_JIS">
```

enableParameterTranslation(String)

Format

public void enableParameterTranslation(java.lang.String encoding)

Description

Enables the translation of all HTML form parameter names and all text-based HTML form parameter values using the specified character encoding when parsing the body of a multipart/form-data POST request.

Character encoding of request parameters is not well defined in the HTTP specification. Most servlet containers interpret them using the servlet default encoding, ISO-8859-1. Therefore, to provide an API that is compatible with such servlet containers, by default, the OrdHttpUploadFormData class also interprets multipart/form-data request parameters using the default encoding, ISO-8859-1.

Applications that process requests and responses using other character encodings can, prior to calling the parseFormData( ) method, call the enableParameterTranslation( ) method to specify the character encoding to be used to translate the names of all HTML form parameters, and the values of all text-based HTML form parameters when parsing the body of a multipart/form-data POST request.

Notes:

- Query string parameters that accompany multipart/form-data POST requests are *not* translated prior to being merged into the list of multipart/form-data parameters. This is because there is no way to determine if the underlying servlet container or JSP engine has decoded the query string or translated the parameter names and values already. Therefore, the application is responsible for translating any multibyte query string parameter names or values in the case where the underlying servlet container or JSP engine does not perform the translation.
- The contents of uploaded files are never translated; nor is the associated content type attribute, which is always represented using the ISO-8859-1 character encoding. However, the file name attribute of an uploaded file is translated.
- Query string parameters in GET requests and query string and POST data parameters in application/x-www-form-urlencoded POST requests are never translated.
- To correctly handle the translation of HTML form parameter names and values, applications must call the enableParameterTranslation( ) method for multipart/form-data POST requests, even if the servlet container or JSP engine translates parameter names and values for GET requests and application/x-www-form-urlencoded POST requests.
- Do not call the enableParameterTranslation( ) method if the application contains code that handles the translation of parameter names and values.
- The enableParameterTranslation( ) method *must* be called *before* calling the parseFormData( ) method.

Calling the enableParameterTranslation( ) method with a character encoding other than ISO-8859-1 affects the following methods when called for multipart/form-data POST requests:

- getParameter( ): parameter name argument and returned parameter value
- getParameterValues( ): parameter name argument and returned parameter values

- getParameterNames( ): returned parameter names
- getFileParameter( ): parameter name argument only
- getFileParameterValues( ): parameter name argument only
- getFileParameterNames( ): returned parameter names

For GET requests and application/x-www-form-urlencoded POST requests, calls to the getParameter( ), getParameterValues( ), and getParameterNames( ) methods are passed directly to the underlying servlet container or JSP engine. Please consult the servlet container or JSP engine documentation for information regarding any parameter translation functions that might be supported by the servlet container or JSP engine.

Parameters

encoding
A String that specifies the character encoding.

Return Value

None.

Exceptions

None.

Examples

None.

getFileParameter(String)

Format

public OrdHttpUploadFile getFileParameter(String parameterName)

Description

Returns information about an uploaded file identified by parameter name, as an OrdHttpUploadFile object.

Every input field of type FILE in an HTML form will produce a parameter of type OrdHttpUploadFile, whether or not you enter a valid file name into the field.

Parameters

parameterName
The name of the uploaded file parameter, as a String.

Return Value

This method returns the uploaded file parameter, as an OrdHttpUploadFile object, or null if the parameter does not exist.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the ServletRequest object has not been specified, if the multipart form data has not been parsed, or if the upload request has been released.

Examples

None.

getFileParameterNames( )

Format

public java.util.Enumeration getFileParameterNames( )

Description

Returns an Enumeration of the names of all input fields of type FILE in an HTML form, or an empty Enumeration if there are no input fields of type FILE.

Every input field of type FILE in an HTML form will produce a parameter of type OrdHttpUploadFile, whether or not you enter a valid file name into the field.

Parameters

None.

Return Value

This method returns a list of uploaded file parameter names, as an Enumeration of Strings.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the ServletRequest object has not been specified, if the multipart form data has not been parsed, or if the upload request has been released.

Examples

None.

getFileParameterValues(String)

Format

public OrdHttpUploadFile[ ] getFileParameterValues(String parameterName)

Description

Returns an array of OrdHttpUploadFile objects that represents all files uploaded using the specified parameter name. Every input field of type FILE in an HTML form will produce a parameter of type OrdHttpUploadFile, whether or not you enter a valid file name in the field.

Parameters

parameterName
The name of the uploaded file parameter, as a String.

Return Value

This method returns the uploaded file parameters, as an array of OrdHttpUploadFile objects, or null if the parameter does not exist.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the ServletRequest object has not been specified, if the multipart form data has not been parsed, or if the upload request has been released.

Examples

None.

getParameter(String)

Format

public String getParameter(String parameterName)

Description

Returns the value of the first query string parameter or text-based form field parameter with the specified name, or null if the parameter does not exist. The query string parameters of the request are merged with the text-based form field parameters by the parseFormData( ) method.

This method calls the getParameterName( ) method in the ServletRequest class if the request is not a multipart/form-data upload request.

Parameters

parameterName
The name of the parameter, as a String.

Return Value

This method returns the value of the specified parameter, as a String, or null if the parameter does not exist.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the ServletRequest object has not been specified, if the multipart form data has not been parsed, or if the upload request has been released.

Examples

None.

getParameterNames( )

Format

public java.util.Enumeration getParameterNames( )

Description

Returns an Enumeration of all the query string parameter names and all the text-based form field data parameter names in the request, or an empty Enumeration if there are no query string parameters and no text-based form field parameters. The query string parameters of the request are merged with the text-based form field parameters by the parseFormData( ) method.

This method calls the getParameterNames( ) method in the ServletRequest class if the request is not a multipart/form-data upload request.

Parameters

None.

Return Value

This method returns a list of parameter names, as an Enumeration of Strings.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the ServletRequest object has not been specified, if the multipart form data has not been parsed, or if the upload request has been released.

Examples

None.

getParameterValues(String)

Format

public String[ ] getParameterValues(String parameterName)

Description

Returns an array of String objects containing the values of all the query string parameters and text-based form field data parameters with the specified parameter name, or null if the parameter does not exist. The query string parameters of the request are merged with the text-based form field parameters by the parseFormData( ) method.

This method calls the getParameterValues( ) method in the ServletRequest class if the request is not a multipart/form-data upload request.

Parameters

parameterName
The name of the parameter, as a String.

Return Value

This method returns the parameter value, as a String, or null if the parameter does not exist.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the ServletRequest object has not been specified, if the multipart form data has not been parsed, or if the upload request has been released.

Examples

None.

isUploadRequest( )

Format

public boolean isUploadRequest( )

Description

Tests if the request was encoded using the multipart/form-data encoding format.

Parameters

None.

Return Value

This method returns true if the request body was encoded using the multipart/form-data encoding format; false otherwise.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the ServletRequest object has not been specified.

Examples

None.

OrdHttpUploadFormData( )

Format

public OrdHttpUploadFormData( )

Description

Creates an OrdHttpFormData object to parse a multipart/form-data request. Subsequently, the application must specify the ServletRequest object.

Parameters

None.

Return Value

None.

Exceptions

None.

Examples

None.

OrdHttpUploadFormData(ServletRequest)

Format

public OrdHttpUploadFormData(javax.servlet.ServletRequest request)

Description

Creates an OrdHttpUploadFormData object to parse a multipart/form-data request.

Parameters

request
An object of type ServletRequest.

Return Value

None.

Exceptions

None.

Examples

None.

parseFormData( )

Format

public void parseFormData( )

Description

Parses the body of a POST request that is encoded using the multipart/form-data encoding. If the request is not an upload request, this method does nothing.

Parameters

None.

Return Value

None.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the ServletRequest object has not been specified.

java.io.IOException

This exception is thrown if an error occurs reading the request body or writing a temporary file.

OrdHttpUploadException

This exception is thrown if an error occurs parsing the multipart/form-data message.

Examples

None.

release( )

Format

public void release( )

Description

Releases all resources held by an OrdHttpUploadFormData object, including temporary files used to hold the contents of uploaded files. An application that enables the use of temporary files must call this method.

Parameters

None.

Return Value

None.

Exceptions

None.

Examples

None.

setMaxMemory(int, String)

Format

public void setMaxMemory(int maxMemory, String tempFileDir)

Description

Specifies the maximum amount of memory that the contents of uploaded files can consume before the contents are stored in a temporary directory.

By default, the contents of uploaded files are held in memory until stored in a database by the application. If users upload large files, such as large video clips, then it may be desirable to limit the amount of memory consumed, and to store temporarily the contents of such files on disk, before the contents are written to a database.

Note: Applications that use this mechanism must ensure that any temporary files are deleted when no longer required by using the release( ) method. See the release( ) method for more information.

Parameters

maxMemory
An integer that specifies the maximum amount of memory to be consumed by all uploaded files in a request before the contents of the uploaded files are stored in temporary files.

tempFileDir
A String that specifies the temporary file directory where you will store temporary files. This parameter is optional if the java.io.tmpdir system property is present.

Return Value

None.

Exceptions

java.lang.IllegalArgumentException

This exception is thrown if the maxMemory parameter is negative, or if the tempFileDir parameter was specified as null and the java.io.tmpdir system property is not present.

Examples

None.

setServletRequest(ServletRequest)

Format

public void setServletRequest(javax.servlet.ServletRequest request)

Description

Specifies the ServletRequest object for the request. The ServletRequest object must be specified either in the constructor, or by calling this method before parsing the request.

Parameters

request
An object of type ServletRequest.

Return Value

None.

Exceptions

None.

Examples

None.

OrdMultipartFilter Class

This section presents reference information on the methods of the oracle.ord.im.OrdMultipartFilter class.

The OrdMultipartFilter class implements the javax.servlet.Filter interface in servlet 2.3. It filters the form-based, file upload servlet request (with the content type multipart/form-data) and wraps the request object using the OrdMultipartWrapper object that parses the request content. For any servlet container that supports servlet 2.3, this filter combined with the OrdMultipartWrapper object provide transparent access to the parameters and files in the servlet request with multipart/form-data encoding.

destroy( )

Format

public void destroy( )

Description

Implements the javax.servlet.Filter interface destroy( ) method that destroys the filter.

Parameters

None.

Return Value

None.

Exceptions

None.

Examples

None.

doFilter(ServletRequest, ServletResponse, FilterChain)

Format

public void doFilter(javax.servlet.ServletRequest request, javax.servlet.ServletResponse response, javax.servlet.FilterChain chain)

Description

Implements the javax.servlet.Filter interface doFilter( ) method that handles the servlet request with multipart/form-data encoding. This method filters the servlet request with multipart/form-data content and wraps it with the OrdMultipartWrapper object. The OrdMultipartWrapper object then parses the content and provides access to the text-based form field parameters and uploaded files.

Parameters

request
The servlet request.

response
The servlet response.

chain
The filters that must be processed.

Return Value

None.

Exceptions

java.io.IOException

This exception is thrown if an error occurs during the I/O operation.

javax.servlet.ServletException

This exception is thrown if an error occurs in the servlet.

Examples

None.

init(FilterConfig)

Format

public void init(javax.servlet.FilterConfig config)

Description

Implements the javax.servlet.Filter interface init( ) method that initializes the filter. The FilterConfig object can be set with initial parameters (for the tempDir and maxMemory attributes), to specify the maximum amount of memory that the contents of uploaded files can consume before the contents are stored in a temporary directory.

Parameters

config
An object of type javax.servlet.FilterConfig.

Return Value

None.

Exceptions

javax.servlet.ServletException

This exception is thrown if an error occurs in the servlet.

Examples

None.

OrdMultipartWrapper Class

This section presents reference information on the methods of the oracle.ord.im.OrdMultipartWrapper class.

The OrdMultipartWrapper class wraps the HttpServletRequest object and provides access to the contents in the HTTP request that is encoded using multipart/form-data encoding. This class overrides some of the methods in the HttpServletRequestWrapper class to provide access to the text-based form field parameters. This class also defines new methods to provide access to the uploaded files.

getFileParameter(String)

Format

public OrdHttpUploadFile getFileParameter(java.lang.String parameterName)

Description

Gets information about an uploaded file identified by the file's parameter name as an OrdHttpUploadFile object. Input fields of type FILE in an HTML form produce a parameter of type OrdUploadFile, whether or not valid file names are entered into input fields of this type.

Parameters

parameterName
The name of the uploaded file parameter, as a String.

Return Value

This method returns the uploaded file parameter, as an OrdHttpUploadFile object, or null if the parameter does not exist.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the ServletRequest object has not been specified, the multipart form data has not been parsed, or the upload request has been released.

Examples

None.

getFileParameterNames( )

Format

public java.util.Enumeration getFileParameterNames( )

Description

Gets an Enumeration of the names of all the input fields of type FILE in an HTML form. Input fields of type FILE in an HTML form produce a parameter of type OrdUploadFile, whether or not valid file names are entered into input fields of this type. This method returns an empty Enumeration if there are no input fields of type FILE.

Parameters

None.

Return Value

This method returns a list of uploaded file parameter names, as an Enumeration of Strings.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the ServletRequest object has not been specified, the multipart form data has not been parsed, or the upload request has been released.

Examples

None.

getFileParameterValues(String)

Format

public OrdHttpUploadFile[ ] getFileParameterValues(java.lang.String parameterName)

Description

Gets an array of OrdHttpUploadFile objects that represents all the files uploaded using the specified parameter name. Input fields of type FILE in an HTML form produce a parameter of type OrdUploadFile, whether or not valid file names are entered into input fields of this type.

Parameters

parameterName
The name of the uploaded file parameter, as a String.

Return Value

This method returns uploaded file parameters, as an array of OrdHttpUploadFile objects, or null if the parameter does not exist.

Exceptions

java.lang.IllegalStateException

This exception is thrown if the ServletRequest object has not been specified, the multipart form data has not been parsed, or the upload request has been released.

Examples

None.

getParameter(String)

Format

public java.lang.String getParameter(java.lang.String name)

Description

Overrides the getParameter( ) method in the HttpServletRequestWrapper class (class javax.servlet.ServletRequestWrapper).

Parameters

name
The name of the parameter, as a String.

Return Value

None.

Exceptions

None.

Examples

None.

getParameterMap( )

Format

public java.util.Map getParameterMap( )

Description

Overrides the getParameterMap( ) method in the HttpServletRequestWrapper class (class javax.servlet.ServletRequestWrapper).

Parameters

None.

Return Value

None.

Exceptions

None.

Examples

None.

getParameterValues(String)

Format

public java.lang.String[ ] getParameterValues(java.lang.String name)

Description

Overrides the getParameterValues( ) method in the HttpServletRequestWrapper class (class javax.servlet.ServletRequestWrapper).

Parameters

name
The name of the parameter, as a String.

Return Value

None.

Exceptions

None.

Examples

None.

release( )

Format

public void release( )

Description

Releases the resources owned by an OrdMultipartWrapper object.

Parameters

None.

Return Value

None.

Exceptions

None.

Examples

None.

A

Deprecated Methods

The following methods have been deprecated:

- 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( )
- 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)
- public static void imCompatibilityInit(oracle.jdbc.driver.OracleConnection con)

Note: Oracle does not recommend using these deprecated methods. Although they are available in the current release, they will not be enhanced and they may be removed in a future release.

Index

I

J

L

M

O

P

R

S

T

W