OracleEOAdaptor Framework

Java API Reference

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

OracleEOAdaptor Framework

Package: com.apple.yellow.oracleeoadaptor

Introduction

The OracleEOAdaptor framework is a set of classes that allow your programs to connect to an Oracle server. These classes provide Oracle-specific method implementations for the EOAccess framework's EOAdaptor, EOAdaptorChannel, EOAdaptorContext, and EOSQLExpression abstract classes.

The following table lists the classes in the OracleEOAdaptor Framework and provides a brief description of each class.

Class	Description
OracleAdaptor	Represents a single connection to a Oracle database server, and is responsible for keeping login and model information, performing Oracle-specific formatting of SQL expressions, and reporting errors.
OracleChannel	Represents an independent communication channel to the database server its OracleAdaptor is connected to.
OracleContext	Represents a single transaction scope on the database server to which its adaptor object is connected.
OracleSQLExpression	Defines how to build SQL statements for OracleChannels.

The Connection Dictionary

The connection dictionary contains items needed to connect to an Oracle server, such as the server name and database (it's common to omit the user name and password from the connection dictionary, and prompt users to enter those values in a login panel). The keys of this dictionary identify the information the server expects, and the values of those keys are the values that the adaptor uses when trying to connect to the server.

The Oracle adaptor defines string constants for use as connection dictionary keys:

- `ServerIdKey`
- `HostMachineKey`
- `UserNameKey`
- `PasswordKey`
- `ConnectionStringKey`
- `NlsLangKey`

The OracleAdaptor attempts to connect with a connection string of the form "userName/password@serverId". If all the values except the one for `ServerIdKey` are absent, then OracleAdaptor attempts to connect with just the value for `ServerIdKey`.

The `ConnectionStringKey` and `NlsLangKey` are optional. The value for `ConnectionStringKey` is a string to be used to login to the database server. If the `ConnectionStringKey` is present in the connection dictionary, the other logon keys (`ServerIdKey`, `HostMachineKey`, `UserNameKey`, and `PasswordKey`) are ignored and the value for the `ConnectionStringKey` is used to connect to the database.

The value for `NlsLangKey` is used to set the Oracle NLS_LANG environment variable. NLS_LANG declares to the Oracle server the character set being used by the client, as well as the language in which you want server error messages to appear. The format is as follows:

```
language_territory. characterSet
```

For example, supplying the value `japanese_japan.jeuc` for the `NlsLangKey` tells the server that the language is Japanese, the territory is Japan, and the character set is jeuc. See your Oracle documentation for a complete list of types available for this field.

To add the `NlsLangKey` and a value to your connection dictionary, you can manually edit your model file. For example:

```
connectionDictionary = {
```

```
    password = tiger;
    serverId = sjOracle;
    userName = scott;
    NLS_LANG = american_america.us7ascii;
};
```

Subsequently changing the connection dictionary in your model file using the Set Adaptor Info command in EOModeler has no effect on these keys and their values-they are preserved unless you edit the file to remove them.

The default character set for Japanese systems is jeuc. If you are using a non-Japanese system, the default is whatever Oracle provides. You only need to add the `NlsLangKey` to your connection dictionary if you are using a character set other than your system's default.

Note: Enterprise Objects Framework uses Mac OS X Server encoding to represent string data, and it passes strings to the database without converting them to the database character set. If you require that the data passed to your server is in an encoding other than Mac OS X Server encoding, you need to subclass NSString.

Locking

All adaptors use the database server's native locking facilities to lock rows on the server. The Oracle adaptor locks a row by using the SELECT... FOR UPDATE... statement. This occurs when:

- You send the adaptor channel a `selectAttributes` message with `true` specified as the value for the `lock` keyword.
- You explicitly lock an object's row with the EODatabaseContext's `lockObjectWithGlobalID` message.
- You set pessimistic locking at the database level and fetch objects.

Data Type Mapping

Every adaptor provides a mapping between each server data type and the Java type to which a database value will be coerced when it's fetched from the database. The following table lists the mapping used by OracleAdaptor.

Oracle Data Type	Java Data Type
VARCHAR2	String
NUMBER	BigDecimal
LONG	String
DATE	NSGregorianDate
RAW	NSData
LONG RAW	NSData
CHAR	String
MLSLABEL	String
REFCURSOR	OracleChannel

The type mapping methods—`externalTypesWithModel`, `internalTypeForExternalTypeInModel`, and `isValidQualifierTypeInModel`—allow for an adaptor to supplement its set of type mappings with additional mappings for user-defined database types. OracleAdaptor does not make use of the model argument if one is provided.

Prototype Attributes

The OracleAdaptor Framework provides the following set of prototype attributes:

Name	External Type	Value Class Name	Other Attributes
binaryID	RAW	NSData	width = 12
city	VARCHAR2	String	columnName = CITY width = 50
date	DATE	NSGregorianDate	columnName = “”
longText	LONG	String	
money	NUMBER	BigNumber	columnName = “”
phoneNumber	VARCHAR2	String	columnName = PHONE width = 20
rawImage	LONG RAW	NSData	columnName = RAW_IMAGE
state	VARCHAR2	String	columnName = STATE width = 2
streetAddress	VARCHAR2	String	columnName = STREET_ADDRESS width = 100
tiffImage	LONG RAW	NSImage	adaptorValueConversionMethodName = TIFFRepresentation columnName = PHOTO valueFactoryMethodName = “imageWithData:”
uniqueID	NUMBER	Number	columnName = “” valueType = i
zipCode	VARCHAR2	String	columnName = ZIP width = 10

Handling Errors

OracleChannel provides a method for handling errors: `raiseOracleError`. This method is invoked whenever the channel encounters an error reported by the Oracle server.

Generating Primary Keys

Each adaptor provides a database-specific implementation of the method `primaryKeyForNewRowWithEntity` for generating primary keys. The OracleChannel's implementation uses sequence objects to provide primary key values. The statement used to create the sequence is:

```
create sequence table_SEQ
```

where table is the name of the table for which the adaptor provides primary key values. The adaptor sets the sequence start value to the corresponding table's maximum primary key value plus one.

To use OracleChannel's database-specific primary key generation mechanism, be sure that your database accommodates the adaptor's scheme. To modify your database so that it supports the adaptor's mechanism for generating primary keys, use EOModeler. For more information on this topic, see *Enterprise Objects Framework Developer's Guide*.

Bind Variables

The OracleAdaptor uses bind variables. A bind variable is a placeholder used in an SQL statement that is replaced with an actual value after the database server determines an execution plan. You use the following OracleSQLExpression methods to operate on bind variables:

- bindVariableDictionaryForAttribute
- mustUseBindVariableForAttribute
- shouldUseBindVariableForAttribute

OracleAdaptor

Inherits from: EOAdaptor : NSObject

Package: com.apple.yellow.oracleeoadaptorjava

Class Description

An OracleAdaptor represents a single connection to an Oracle database server, and is responsible for keeping login and model information, performing Oracle-specific formatting of SQL expressions, and reporting errors.

The OracleAdaptor class has these restrictions: You can't have nested transactions, and the adaptor doesn't support full outer joins.

Constants

OracleAdaptor defines the following string constants for use as connection dictionary keys.

Constant	Corresponding value in the connection dictionary
`ServerIdKey`	The server ID. Used as the string to connect to the database if values for `HostMachineKey`, `UserNameKey`, and `PasswordKey` are not present in the database.
`HostMachineKey`	The name of the host machine. If this key is not present, the string used to log in to the database is of the form "userName/password@serverId".
`UserNameKey`	The name of the user to log in as.
`PasswordKey`	The user's password.
`ConnectionStringKey`	The connection string used to log in to a database server. If this key is present, `ServerIdKey`, `HostMachineKey`, `UserNameKey`, and `PasswordKey` are ignored.
`NlsLangKey`	The setting to NLS_LANG, which is used to specify the language and character set for server connections. On J systems this option defaults to japanese_japan.jeuc.

See "The Connection Dictionary" (page 4) in the OracleEOAdaptor framework introduction for more information on the connection dictionary and its entries.

OracleAdaptor also defines a string constant for use as a key in an exception's userInfo dictionary (see `raiseOracleError` in the OracleChannel class specification).

Constant	Corresponding value in an exception's userInfo dictionary
`OracleErrorKey`	The Oracle OCI client library error code.

Instance Methods

connectionKeys

```
public NSArray connectionKeys()
```

Returns an NSArray containing the keys in the receiver's connection dictionary. You can use this method to prompt the user to supply values for the connection dictionary.

fetchedValueForDateValue

```
public NSGregorianDate fetchedValueForDateValue(
   NSGregorianDate value,
   EOAttribute attribute)
```

Returns an NSGregorianDate based on date whose millisecond value is set to 0.

fetchedValueForNumberValue

```
public Number fetchedValueForNumberValue(
   Number value,
   EOAttribute attribute)
```

Returns a Number based on *numberValue* that has been rounded according to the precision and scale specified for attribute.

oracleConnectionString

```
public String oracleConnectionString()
```

Returns the user name, password, host machine, and server id as a string suitable to be supplied as an argument to `orlon()`.

OracleChannel

Inherits from: EOAdaptorChannel : NSObject

Package: com.apple.yellow.oracleeoadaptorjava

Class Description

An OracleChannel represents an independent communication channel to the database server its OracleAdaptor is connected to. All of an OracleChannel's operations take place within the context of transactions controlled or tracked by its OracleContext. An OracleContext can manage multiple OracleChannels, and a channel is associated with only one context.

The features OracleChannel adds to EOAdaptorChannel are as follows:

- Oracle-specific error handling
- The ability to configure the fetch buffer
- The ability to read a default list of table names from the database

Method Types

Finding table names

`oracleTableNamesSQL`

`setOracleTableNamesSQL`

`describeTableNames`

Accessing the fetch buffer length

`fetchBufferLength`

`setFetchBufferLength`

Error handling

`raiseOracleError`

Static Methods

oracleTableNamesSQL

```
public static String oracleTableNamesSQL()
```

Returns the SQL statement that will be executed when building a default model.

setOracleTableNamesSQL

```
public static void setOracleTableNamesSQL(String sql)
```

Sets to *sql* the SQL statement that will be used to return a list of table names from the database. By default, this list is the result of the SQL statement:

```
SELECT TABLE_NAME FROM USER_TABLES ORDER BY TABLE_NAME
```

This setting is used by all OracleChannels in an application. You can specify a different SQL statement using the defaults write command, for example:

```
% defaults write NSGlobalDomain OracleTableNamesSQL "SELECT TABLE_NAME FROM..."
```

Once you use `setOracleTableNamesSQL` to specify a setting, it supersedes values set with the `defaults write` command.

Instance Methods

describeTableNames

```
public NSArray describeTableNames()
```

Overrides the EOAdaptorChannel implementation to return an array of the names of all the tables owned by the current user. Uses the SQL returned by `oracleTableNamesSQL`.

fetchBufferLength

```
public int fetchBufferLength()
```

Returns the size, in bytes, of the fetch buffer. The larger the buffer, the more rows can be returned for each round trip to the server.

raiseOracleError

```
public void raiseOracleError()
```

Examines Oracle structures for error flags and raises an exception if one is found. The userInfo of the exception contains the Oracle error code in an entry with the key `OracleErrorKey`.

setFetchBufferLength

```
public void setFetchBufferLength(int length)
```

Sets to *`length`* the size, in bytes, of the fetch buffer. The larger the buffer, the more rows can be returned for each round trip to the server.

OracleContext

Inherits from: EOAdaptorContext : NSObject

Package: com.apple.yellow.oracleeoadaptorjava

Class Description

An OracleContext represents a single transaction scope on the database server to which its adaptor object is connected. If the server supports multiple concurrent transaction sessions, the adaptor may have several adaptor contexts. An OracleContext may in turn have several OracleChannels, which handle actual access to the data on the server.

Method Types

Managing a connection to the server

`connect`

`disconnect`

`isConnected`

Tracking fetches

`fetchesInProgress`

Instance Methods

connect

```
public void connect()
```

Opens a connection to the database server. OracleChannel sends this message to OracleContext when it (OracleChannel) is about to open a channel to the server.

disconnect

```
public void disconnect()
```

Closes a connection to the database server. OracleChannel sends this message to OracleContext when it (OracleChannel) has just closed a channel to the server.

fetchesInProgress

```
public int fetchesInProgress()
```

Returns the number of fetches the receiver has in progress.

isConnected

```
public boolean isConnected()
```

Returns `true` if the receiver has an open connection to the database, `false` otherwise.

OracleSQLExpression

Inherits from: EOSQLExpression : NSObject

Package: com.apple.yellow.oracleeoadaptorjava

Class Description

OracleSQLExpression defines how to build SQL statements for OracleChannels.

Static Methods

serverTypeIdForName

```
public static int serverTypeIdForName(String typeName)
```

Returns the Oracle type code (such as OraVARCHAR2 or OraNumber) for *typeName* (such as "VARCHAR2" or "NUMBER").

setUseNoWaitLocks

```
public static void setUseNoWaitLocks(boolean flag)
```

Sets according to *flag* whether the lock clause of the OracleSQLExpression is "FOR UPDATE" (block until the row is available) or "FOR UPDATE NOWAIT" (return an error immediately if an attempt to lock a row would block). By default OracleSQLExpression uses the clause "FOR UPDATE"—that is, by default it does not use NOWAIT locks. This behavior is also controllable through the `EOOracleUseNoWaitLocks` user default.

setUseQuotedExternalNames

```
public static void setUseQuotedExternalNames(boolean flag)
```

Sets according to *flag* whether the OracleSQLExpression expects external (database) names to be enclosed in quotation marks. This is useful if the database has table or column names that are either reserved words or that are not all uppercase. The default is NO.

This behavior can also be controlled through the `EOOracleUseQuotedExternalNames` user default.

useNoWaitLocks

```
public static boolean useNoWaitLocks()
```

Returns `true` to indicate that the OracleSQLExpression uses NOWAIT locks, `false` otherwise. The default is `false`.

useQuotedExternalNames

```
public static boolean useQuotedExternalNames()
```

Returns YES to indicate that the OracleSQLExpression uses quoted external (database) names, NO otherwise.

Instance Methods

lockClause

```
public String lockClause()
```

Overrides the EOSQLExpression method `lockClause` to return the SQL string used in a SELECT statement to lock selected rows. Queries the user default EOOracleUseNoWaitLocks. If this default is not set or if it is set to `false`, this method returns the string "FOR UPDATE". If the default is set to `true`, this method returns "FOR UPDATE NOWAIT".

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software.

Line art was created using Adobe™ Illustrator and Adobe Photoshop.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Adobe Letter Gothic.