SQL Anywhere - Mobile Link
文書バージョン: 17 – 2016-05-11

Mobile Link - Java API リファレンス

目次

1 Mobile Link サーバ Java API リファレンス

Mobile Link サーバ Java API のトピックでは、インタフェースとクラス、これらに関連するメソッド、コンストラクタについて説明します。これらのクラスを使用するには、*%SQLANY17%*¥java¥ にある mlscript.jar アセンブリを参照してください。

パッケージ

```
com.sap.ml.script
```

> **注記**
>
> 主な **SQL Anywhere** マニュアルをお探しですか。マニュアルをローカルにインストールした場合は、Windows のスタートメニューを使用してアクセスするか (Microsoft Windows)、C:¥Program Files¥SQL Anywhere 17¥Documentation にナビゲートします。
>
> また、DocCommentXchange の Web で、主な SQL Anywhere API リファレンスマニュアルにアクセスすることもできます。http://dcx.sap.com

このセクションの内容:

サーバで同期の進行を妨げるエラー状態が存在することを示すために発行されます。

ShutdownListener インタフェース [54 ページ]
サーバのシャットダウンを取得するリスナインタフェースです。

SpatialUtilities クラス [56 ページ]
空間値を操作する静的メソッドのコレクション。

TimestampWithTimeZone クラス [59 ページ]
タイムゾーンを取得および設定するメソッドでの *java.sql.Timestamp*。

UpdateResultSet インタフェース [67 ページ]
指定した行の更新前イメージ値 (古い値) と更新後イメージ値 (新しい値) にアクセスするための特別なメソッドを含む結果セットオブジェクトです。

UploadData インタフェース [70 ページ]
ダイレクトローハンドリングで使用するアップロード操作をカプセル化します。

UploadedTableData インタフェース [73 ページ]
ダイレクトローハンドリングアップロードで使用するテーブル操作をカプセル化します。

1.1 DBConnectionContext インタフェース

現在のデータベース接続に関する情報を取得します。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface DBConnectionContext
```

メンバー

DBConnectionContext のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public java.sql.Connection	getConnection() [8 ページ]	統合データベースへの既存の接続を返します。

変更子とタイプ	メソッド	説明
public DownloadData	getDownloadData() [9 ページ]	現在の同期に対する DownloadData を返します。
public NetworkData	getNetworkData() [10 ページ]	同期用のネットワークストリームの情報を返します。
public Properties	getProperties() [11 ページ]	スクリプトバージョンに基づいて、この接続のプロパティを返します。
public String	getRemoteID() [11 ページ]	この接続で現在同期中のデータベースのリモート ID を返します。
public ServerContext	getServerContext() [12 ページ]	この同期サーバの ServerContext を返します。
public String	getVersion() [13 ページ]	この接続のバージョン文字列を返します。

備考

このインタフェースは、スクリプトを含むクラスのコンストラクタに渡されます。コンテキストがバックグラウンドスレッドに必要な場合や、接続期間を超えて必要な場合は、*ServerContext* インタフェースを使用してください。

注記

DBConnectionContext インスタンスは、Java コードに呼び出すスレッド以外で使用しないでください。

例

次の例は、同期スクリプトで使用するクラスレベルの *DBConnectionContext* インスタンスを作成する方法を示します。*DBConnectionContext.getConnection* メソッドは、統合データベースとの現在の接続を表す *DBConnection* インスタンスを取得します。

```
import com.sap.ml.script;
public class OrderProcessor {
   DBConnectionContext _cc;
   public OrderProcessor( DBConnectionContext cc ) {
       _cc = cc;
   }
   // The method used for the handle_DownloadData event.
   public void HandleEvent() {
       DBConnection my_connection = _cc.GetConnection();
       // ...
   }
   // ...
}
```

このセクションの内容:

現在の同期に対する DownloadData を返します。

getNetworkData() メソッド [10 ページ]
同期用のネットワークストリームの情報を返します。

getProperties() メソッド [11 ページ]
スクリプトバージョンに基づいて、この接続のプロパティを返します。

getRemoteID() メソッド [11 ページ]
この接続で現在同期中のデータベースのリモート ID を返します。

getServerContext() メソッド [12 ページ]
この Mobile Link サーバの ServerContext を返します。

getVersion() メソッド [13 ページ]
この接続のバージョン文字列を返します。

関連情報

ServerContext インタフェース [39 ページ]

1.1.1 getConnection() メソッド

統合データベースへの既存の接続を返します。

構文

```
public java.sql.Connection getConnection () throws SQLException
```

戻り値

JDBC 接続として存在する既存の接続。

例外

java.sql.SQLException 既存の接続を JDBC 接続としてバインドするときにエラーが発生するとスローされます。

備考

この接続は、この同期用に同期サーバが SQL スクリプトの実行時に使用するものと同じです。

この接続は、同期サーバのこの接続での使用に影響する方法でコミット、クローズ、または変更しないでください。返される接続は、基本となる接続の期間中にのみ有効です。

注記

接続に対して end_connection イベントが呼び出された後は、その接続を使用しないでください。

関連情報

makeConnection() メソッド [48 ページ]
DBConnectionContext インタフェース [6 ページ]

1.1.2 getDownloadData() メソッド

現在の同期に対する DownloadData を返します。

構文

```
public DownloadData getDownloadData ()
```

戻り値

現在の同期の DownloadData。同期がダウンロードされてない場合は Null です。

備考

ダイレクトローハンドリング用のダウンロードを作成する場合は、DownloadData クラスを使用してください。

例

次の例は、*DBConnectionContext.getDownloadData* メソッドを使用して、現在の同期に対する *DownloadData* インスタンスを取得する方法を示します。この例は、*_cc* という *DBConnectionContext* インスタンスがあることを前提としています。

```
// The method used for the handle_DownloadData event.
```

```
public void HandleDownload() throws SQLException {
   // Get the DownloadData for the current synchronization
   DownloadData my_dd = _cc.getDownloadData();
   // ...
}
// ...
```

関連情報

DownloadData インタフェース [13 ページ]

1.1.3 getNetworkData() メソッド

同期用のネットワークストリームの情報を返します。

構文

```
public NetworkData getNetworkData ()
```

戻り値

要求に使用されたネットワークストリームの情報、コレクションが有効でない場合は NULL

備考

この方法は、クライアント側証明書と HTTP ヘッダを使用するエンタープライズ内の別のサーバを認証する際に便利です。

ネットワークストリームのデータの収集を有効にするには、-x スイッチに collect_network_data=1 を追加します。このオプションにより、同期ごとのデータの保存時に、メモリのオーバヘッドが増えます。

関連情報

NetworkData インタフェース [33 ページ]

1.1.4 getProperties() メソッド

スクリプトバージョンに基づいて、この接続のプロパティを返します。

構文

```
public Properties getProperties ()
```

戻り値

この接続のプロパティ

備考

プロパティは、*ml_property* テーブルに格納されます。

java.util.Properties の詳細については、Java Software Development Kit マニュアルを参照してください。

例

次の例は、*DBConnectionContext* のプロパティを出力する方法を示します。この例は、*_cc* という *DBConnectionContext* インスタンスがあることを前提としています。

```
// The method used to output the connection properties.
public void outputProperties() {
   // Output the properties for the current synchronization
   java.util.Properties properties = _cc.getProperties();
   System.out.println(properties.toString());
}
```

1.1.5 getRemoteID() メソッド

この接続で現在同期中のデータベースのリモート ID を返します。

構文

```
public String getRemoteID ()
```

戻り値

リモート ID。

例

次の例は、*DBConnectionContext* のリモート ID を出力する方法を示します。この例は、*_cc* という *DBConnectionContext* インスタンスがあることを前提としています。

```
// The method used to output the remote ID.
public void outputRemoteID() {
   // output the Remote ID for the current synchronization
   String remoteID = _cc.getRemoteID();
   System.out.println(remoteID);
}
```

1.1.6 getServerContext() メソッド

この Mobile Link サーバの ServerContext を返します。

構文

```
public ServerContext getServerContext ()
```

戻り値

Mobile Link サーバコンテキスト。

例

次の例は、*DBConnectionContext* の *ServerContext* インスタンスを取得する方法を示します。この例は、*_cc* という *DBConnectionContext* インスタンスがあることを前提としています。

```
// A method that uses a ServerContext instance to shut down the server
public void shutDownServer() {
   ServerContext context = _cc.getServerContext();
   context.shutdown();
}
```

関連情報

ServerContext インタフェース [39 ページ]

1.1.7 getVersion() メソッド

この接続のバージョン文字列を返します。

構文

```
public String getVersion ()
```

戻り値

スクリプトバージョン。

例

次の例は、スクリプトバージョンを取得して、その値に基づいて決定を行う方法を示します。この例は、*_cc* という *DBConnectionContext* インスタンスがあることを前提としています。

```
// A method that uses the script version
public void handleEvent() {
   // ...
   String version = _cc.getVersion();
   if (version.equals("My Version 1")) {
       // ...
   } else if (version.equals("My Version 2")) {
       // ...
   }
}
// ...
```

1.2 DownloadData インタフェース

ダイレクトローハンドリングで使用するダウンロードデータ操作をカプセル化します。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface DownloadData
```

メンバー

DownloadData のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public DownloadTableData	getDownloadTableByName(String) [15 ページ]	現在の同期に使用する名前付きダウンロードテーブルを取得します。
public DownloadTableData[]	getDownloadTables() [16 ページ]	現在の同期のダウンロードのすべてのテーブルを含む配列を取得します。

備考

DownloadData のインスタンスを取得するには、*DBConnectionContext.getDownloadData* メソッドを使用します。

DownloadTableData のインスタンスを返すには、*getDownloadTables* メソッドと *getDownloadTableByName* メソッドを使用します。

ダウンロードデータは *DBConnectionContext* を使用して取得できます。begin_synchronization イベントの前または end_download イベントの後にダウンロードデータにアクセスすることはできません。また、*DownloadData* にアップロード専用同期でアクセスすることもできません。

例

次の例は、*DBConnectionContext.getDownloadData* メソッドを使用して、現在の同期に対する *DownloadData* インスタンスを取得する方法を示します。

```
DBConnectionContext _cc;
// Your class constructor.
public OrderProcessor(DBConnectionContext cc) {
   _cc = cc;
}
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.getDownloadData();
   // ...
}
```

このセクションの内容:

関連情報

getDownloadData() メソッド [9 ページ]
DownloadTableData インタフェース [17 ページ]

1.2.1 getDownloadTableByName(String) メソッド

現在の同期に使用する名前付きダウンロードテーブルを取得します。

構文

```
public DownloadTableData getDownloadTableByName (String table_name)
```

パラメータ

table_name ダウンロードデータの取得先テーブルの名前

戻り値

指定されたテーブルを表す *DownloadTableData* インスタンス。指定された名前のテーブルが現在の同期で存在しない場合は NULL。

例

次の例では *getDownloadTableByName* メソッドを使用して、*remoteOrders* テーブルの *DownloadTableData* インスタンスを返します。この例は、*_cc* という *DBConnectionContext* インスタンスがあることを前提としています。

```
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.getDownloadData();
   // Get the DownloadTableData for the remoteOrders table.
   DownloadTableData my_download_table =
my_dd.getDownloadTableByName("remoteOrders");
   // ...
}
```

関連情報

DownloadData インタフェース [13 ページ]

DownloadTableData インタフェース [17 ページ]
DBConnectionContext インタフェース [6 ページ]

1.2.2 getDownloadTables() メソッド

現在の同期のダウンロードのすべてのテーブルを含む配列を取得します。

構文

```
public DownloadTableData[] getDownloadTables ()
```

戻り値

現在の同期の *DownloadTableData* オブジェクトの配列。配列内でのテーブルの順序は、リモートのアップロード順と同じです。

備考

このテーブルに対して実行された操作はリモートデータベースに送信されます。

例

次の例では *DownloadData.getDownloadTables* メソッドを使用して、現在の同期の *DownloadTableData* オブジェクトの配列を取得します。この例は、*_cc* という *DBConnectionContext* インスタンスがあることを前提としています。

```
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.getDownloadData();
   // Get an array of tables to set download operations.
   DownloadTableData[] download_tables = my_dd.getDownloadTables();
   // Get the first table in the DownloadTableData array.
   DownloadTableData my_download_table = download_tables[0];
   // ...
}
```

関連情報

DownloadData インタフェース [13 ページ]
DownloadTableData インタフェース [17 ページ]

DBConnectionContext インタフェース [6 ページ]

1.3 DownloadTableData インタフェース

1 つのダウンロードテーブルの情報を同期用にカプセル化します。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface DownloadTableData
```

メンバー

DownloadTableData のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public java.sql.PreparedStatement	getDeletePreparedStatement() [19 ページ]	ユーザが削除操作をダウンロードに追加できるようにする java.sql.PreparedStatement インスタンスを返します。
public java.sql.Timestamp	getLastDownloadTime() [21 ページ]	このテーブルの最終ダウンロード時刻を返します。
public java.sql.ResultSetMetaData	getMetaData() [22 ページ]	DownloadTableData インスタンスのメタデータを取得します。
public String	getName() [23 ページ]	DownloadTableData インスタンスのテーブル名を返します。
public java.sql.PreparedStatement	getUpsertPreparedStatement() [24 ページ]	ユーザがアップサート (更新または挿入) 操作を同期のダウンロードに追加できるようにする java.sql.PreparedStatement インスタンスを返します。

備考

このインタフェースを使用して、クライアントにダウンロードされるデータ操作を設定します。

DownloadData インタフェースを使用して、現在の同期の *DownloadTableData* インスタンスを取得できます。*getUpsertPreparedStatement* と *getDeletePreparedStatement* メソッドを使用して、それぞれ更新/挿入操作と削除操作を行う Java 準備文を取得できます。

テーブルをリモートでトランケートするため、すべてのプライマリキーを NULL に設定して削除文を実行できます。

java.sql.PreparedStatement.executeUpdate メソッドはダウンロードの操作を登録します。*java.sql.PreparedStatement* の詳細については、Java Software Development Kit マニュアルを参照してください。

注記

挿入と更新用の準備文のすべてのカラム値を設定してください。削除操作の場合は、プライマリキー値を設定します。削除とアップサート準備文を両方同時に開いておくことはできません。

例

この例では、次の SQL 文を使用して作成された同期クライアントデータベースに *remoteOrders* というテーブルがあることを前提にしています。

```
CREATE TABLE remoteOrders (
   pk INT NOT NULL,
   col1 VARCHAR(200),
   PRIMARY KEY (pk)
);
```

次の例では *DownloadData.getDownloadTableByName* メソッドを使用して、*remoteOrders* テーブルを表す *DownloadTableData* インスタンスを返します。この例は、*_cc* という *DBConnectionContext* インスタンスがあることを前提としています。

```
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.getDownloadData();
   // Get the DownloadTableData for the remoteOrders table.
   DownloadTableData td = my_dd.getDownloadTableByName("remoteOrders");
   // User defined-methods to set download operations.
   setDownloadInserts(td);
   setDownloadDeletes(td);
   // ...
}
```

この例では、*SetDownloadInserts* メソッドは *GetUpsertCommand* を使用して、挿入または更新するローのコマンドを取得します。*IDbCommand* は、リモートデータベースに挿入する値の設定先となるパラメータを保持します。

```
void setDownloadInserts(DownloadTableData td) {
   java.sql.PreparedStatement insert_ps = td.getUpsertPreparedStatement();
   // The following method calls are the same as the following SQL statement:
   // INSERT INTO remoteOrders(pk, col1) values(2300, "truck");
   insert_ps.setInt(1, 2300);
   insert_ps.setString(2, "truck");
   int update_result = insert_ps.executeUpdate();
   if (update_result == 0) {
       // Insert was filtered because it was uploaded
       // in the same synchronization.
```

```
    }
    else {
        // Insert was not filtered.
    }
}
```

setDownloadDeletes メソッドは *DownloadTableData.getDeletePreparedStatement* を使用して、削除するローの準備文を取得します。*java.sql.PreparedStatement.setInt* メソッドは、リモートデータベースで削除するローのプライマリキー値を設定し、*java.sql.PreparedStatement.executeUpdate* メソッドはダウンロードするロー値を登録します。

```
void setDownloadDeletes(DownloadTableData td) {
   java.sql.PreparedStatement delete_ps = td.getDeletePreparedStatement();
   // The following method calls are the same as the following SQL statement:
   // DELETE FROM remoteOrders  where pk=2300;
   delete_ps.setInt(1, 2300);
   delete_ps.executeUpdate();
}
```

このセクションの内容:

関連情報

DownloadData インタフェース [13 ページ]

1.3.1 getDeletePreparedStatement() メソッド

ユーザが削除操作をダウンロードに追加できるようにする java.sql.PreparedStatement インスタンスを返します。

構文

```
public java.sql.PreparedStatement getDeletePreparedStatement () throws
java.sql.SQLException
```

戻り値

削除操作をダウンロードに追加するための *java.sql.PreparedStatement* インスタンス

例外

java.sql.SQLException 削除用の *java.sql.PreparedStatement* インスタンスの取り出し時に問題が発生した場合に発行されます。

備考

この準備文は *DownloadTableData* インスタンスに適用され、テーブルの各プライマリキーカラムに対するパラメータを含んでいます。

ダウンロードに削除操作を含めるには、*java.sql.PreparedStatement* ですべてのカラムを設定してから、*java.sql.PreparedStatement.executeUpdate* メソッドを呼び出します。

テーブルをリモートデータベースでトランケートするため、すべてのパラメータを NULL に設定します。

注記

ダウンロード削除操作対象のすべてのプライマリキー値を設定するか、トランケート操作対象のすべてのプライマリキー値を NULL に設定してください。

例

次の例では、*setDownloadDeletes* メソッドは *getDeletePreparedStatement* を使用して、削除するローの準備文を取得します。*java.sql.PreparedStatement.setInt* メソッドは、リモートデータベースで削除するローのプライマリキー値を設定し、*java.sql.PreparedStatement.executeUpdate* メソッドはダウンロード内のロー値を設定します。

```
void setDownloadDeletes(DownloadTableData td) {
   java.sql.PreparedStatement delete_ps = td.getDeletePreparedStatement();
   // This is the same as executing the following SQL statement:
   // DELETE FROM remoteOrders  where pk=2300;
   delete_ps.setInt(1, 2300);
   delete_ps.executeUpdate();
   delete_ps.close();
}
```

関連情報

DownloadTableData インタフェース [17 ページ]

1.3.2 getLastDownloadTime() メソッド

このテーブルの最終ダウンロード時刻を返します。

構文

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

戻り値

このダウンロードテーブルの最終ダウンロード時刻

備考

これは、テーブルごとのダウンロードイベントの多くで渡される最終ダウンロード時刻と同じです。

最終ダウンロード時刻は、特定の同期に対してテーブルダウンロードデータを生成する場合に便利です。

例

次の例では、前回のダウンロード時刻を使用した挿入によって、ダウンロードのテーブルにデータを移植する方法を示します。この例は、*_cc* という *DBConnectionContext* インスタンスがあることを前提としています。

```
// The method used for the handle_DownloadData event.
public void handleDownload() throws SQLException {
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.getDownloadData();
   // Get the DownloadTableData for the remoteOrders table.
   DownloadTableData td = my_dd.getDownloadTableByName("remoteOrders");
   // Get the inserts given a last download time.
   ResultSet inserts_rs  = makeInsertsFromTimestamp(td.getLastDownloadTime());
   // Fill the DownloadTableData using the inserts resultset.
   setDownloadInsertsFromRS(td, inserts_rs);
   inserts_rs.close();
   // ...
}
```

関連情報

DownloadTableData インタフェース [17 ページ]

1.3.3 getMetaData() メソッド

DownloadTableData インスタンスのメタデータを取得します。

構文

```
public java.sql.ResultSetMetaData getMetaData ()
```

戻り値

DownloadTableData インスタンスのメタデータ

備考

メタデータは標準 *java.sql.ResultSetMetaData* オブジェクトです。

メタデータにカラム名情報を含める場合は、クライアントで、アップロードとともにカラム名が送信されるように指定します。

java.sql.ResultSetMetaData の詳細については、Java Software Development Kit マニュアルを参照してください。

例

次の例は、*DownloadTableData* インスタンスのクエリで使用されるカラムの数を取得する方法を示します。

```
import java.sql.ResultSetMetaData;
// The method used to return the number of columns in a DownloadTableData
instance query
public int getNumColumns(DownloadTableData td) {
   ResultSetMetaData rsmd = td.getMetaData();
   return rsmd.getColumnCount();
}
```

関連情報

DownloadTableData インタフェース [17 ページ]

1.3.4 getName() メソッド

DownloadTableData インスタンスのテーブル名を返します。

構文

```
public String getName ()
```

戻り値

DownloadTableData インスタンスのテーブル名

備考

getMetaData メソッドによって返される *java.sql.ResultSetMetaData* インスタンスを使用して、テーブル名にアクセスすることもできます。

例

次の例は、*DownloadTableData* インスタンスのテーブル名を出力する方法を示します。この例は、*_cc* という *DBConnectionContext* インスタンスがあることを前提としています。

```
// The method used for the handle_DownloadData event
public void handleDownload() throws SQLException {
// Get the DownloadData for the current synchronization.
DownloadData my_dd = _cc.getDownloadData();
// Get the DownloadTableData for the remoteOrders table.
DownloadTableData td = my_dd.getDownloadTableByName("remoteOrders");
// Print the table name to standard output (remoteOrders)
System.out.println(td.getName());
// User defined-methods to set download operations.
setDownloadInserts(td);
setDownloadDeletes(td);
// ...
}
```

関連情報

DownloadTableData インタフェース [17 ページ]
getMetaData() メソッド [22 ページ]

1.3.5 getUpsertPreparedStatement() メソッド

ユーザがアップサート (更新または挿入) 操作を同期のダウンロードに追加できるようにする java.sql.PreparedStatement インスタンスを返します。

構文

```
public java.sql.PreparedStatement getUpsertPreparedStatement () throws
java.sql.SQLException
```

戻り値

アップサート操作をダウンロードに追加するための *java.sql.PreparedStatement* インスタンス

例外

java.sql.SQLException アップサート用の *java.sql.PreparedStatement* インスタンスの取り出し時に問題が発生した場合に発行されます。

備考

この準備文は DownloadTableData インスタンスに適用され、テーブルの各カラムに対するパラメータを含んでいます。

ダウンロードに挿入または更新操作を含めるには、*java.sql.PreparedStatement* ですべてのカラム値を設定してから、*java.sql.PreparedStatement.executeUpdate* メソッドを呼び出します。準備文で *java.sql.PreparedStatement.executeUpdate* を呼び出すと、挿入または更新操作がフィルタされた場合には 0 が返され、フィルタされなかった場合には 1 が返されます。同じ同期でアップロードされた場合、操作はフィルタされます。

注記

ダウンロード挿入と更新操作のすべてのカラム値を設定してください。

例

次の例では、*setDownloadInserts* メソッドは *getUpsertPreparedStatement* を使用して、挿入または更新するローの準備文を取得します。*java.sql.PreparedStatement.setInt* メソッドと *PreparedStatement.setString* メソッドはカラム値を設定し、*PreparedStatement.executeUpdate* メソッドはダウンロード内のロー値を設定します。

```
void setDownloadInserts(DownloadTableData td) {
   java.sql.PreparedStatement insert_ps = td.getUpsertPreparedStatement();
   // This is the same as executing the following SQL statement:
   // INSERT INTO remoteOrders(pk, col1) VALUES (2300, "truck");
   insert_ps.setInt(1, 2300);
```

```
    insert_ps.setString(2, "truck");
    int update_result = insert_ps.executeUpdate();
    if (update_result == 0) {
        // Insert was filtered because it was uploaded
        // in the same synchronization.
    }
    else {
        // Insert was not filtered.
    }
    insert_ps.close();
}
```

関連情報

DownloadTableData インタフェース [17 ページ]

1.4 InOutInteger インタフェース

SQL スクリプトに渡される in/out パラメータの機能を有効にするために、メソッドに渡されます。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface InOutInteger
```

メンバー

OutInteger のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public int	getValue() [26 ページ]	この整数パラメータの値を返します。
public void	setValue(int) [27 ページ]	この整数パラメータの値を設定します。

例

次の同期システムプロシージャコールは、スクリプトバージョン *ver1* を同期するときに、*handleError* という Java メソッドを *handle_error* 接続イベント用のスクリプトとして登録します。

```
CALL ml_add_java_connection_script(
   'ver1',
   'handle_error',
   'ExamplePackage.ExampleClass.handleError'
)
```

次に示すサンプルは、*handleError* という Java メソッドです。このメソッドは、渡されたデータに基づいてエラーを処理します。また、エラーの結果生じるエラーコードも判別します。

```
public String handleError(
   ianywhere.ml.script.InOutInteger actionCode,
   int errorCode,
   String errorMessage,
   String user,
   String table)
{
int new_ac;
if (user == null) {
   new_ac = handleNonSyncError(errorCode, errorMessage);
} else if (table == null) {
   new_ac = handleConnectionError(errorCode, errorMessage, user);
}
else {
   new_ac = handleTableError(errorCode, errorMessage, user, table);
}
// Keep the most serious action code.
if (actionCode.getValue() < new_ac) {
   actionCode.setValue(new_ac);
}
}
```

このセクションの内容:

1.4.1 getValue() メソッド

この整数パラメータの値を返します。

構文

```
public int getValue ()
```

戻り値

この整数の値。

関連情報

InOutInteger インタフェース [25 ページ]

1.4.2 setValue(int) メソッド

この整数パラメータの値を設定します。

構文

```
public void setValue (int new_value)
```

パラメータ

new_value この整数が取る値

関連情報

InOutInteger インタフェース [25 ページ]

1.5 InOutString インタフェース

SQL スクリプトに渡される in/out パラメータの機能を有効にするために、メソッドに渡されます。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface InOutString
```

メンバー

InOutString のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public String	getValue() [29 ページ]	この文字列パラメータの値を返します。
public void	setValue(String) [29 ページ]	この文字列パラメータの値を設定します。

例

次の同期システムプロシージャコールは、スクリプトバージョン *ver1* を同期するときに、*modifyUser* という Java メソッドを *modify_user* 接続イベント用のスクリプトとして登録します。

```
CALL ml_add_java_connection_script(
    'ver1',
    'modify_user',
    'ExamplePackage.ExampleClass.modifyUser'
)
```

次に示すサンプルは、*modifyUser* という Java メソッドです。このメソッドは、データベースからユーザ ID を取得し、それを使用してユーザ名を設定します。

```
public String modifyUser(InOutString io_user_name) throws SQLException {
    Statement uid_select = curConn.createStatement();
    ResultSet uid_result = uid_select.executeQuery(
        "SELECT rep_id FROM SalesRep WHERE name = '"
            + io_user_name.getValue() + "' "
    );
    uid_result.next();
    io_user_name.setValue(java.lang.Integer.toString(uid_result.getInt(1));
    uid_result.close();
    uid_select.close();
    return (null);
}
```

このセクションの内容:

1.5.1 getValue() メソッド

この文字列パラメータの値を返します。

構文

```
public String getValue ()
```

戻り値

この文字列パラメータの値

関連情報

InOutString インタフェース [27 ページ]

1.5.2 setValue(String) メソッド

この文字列パラメータの値を設定します。

構文

```
public void setValue (String new_value)
```

パラメータ

new_value この文字列が取る値

関連情報

InOutString インタフェース [27 ページ]

1.6 LogListener インタフェース

ログに出力されるメッセージを取得するために使用されます。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface LogListener
```

メンバー

LogListener のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public void	messageLogged(ServerContext, LogMessage) [30 ページ]	メッセージがログに出力されたときに呼び出されます。

このセクションの内容:

1.6.1 messageLogged(ServerContext, LogMessage) メソッド

メッセージがログに出力されたときに呼び出されます。

構文

```
public void messageLogged (
    ServerContext sc,
    LogMessage message
)
```

パラメータ

sc メッセージを出力しているサーバのコンテキスト
message 同期サーバログに送信された *LogMessage*

1.7 LogMessage クラス

ログメッセージに関連付けられたデータを保持します。

パッケージ

```
com.sap.ml.script
```

構文

```
public class LogMessage
```

メンバー

LogMessage のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

変数

変更子とタイプ	変数	説明
public static final int	ERROR	ログメッセージがエラーであることを示します。
public static final int	WARNING	ログメッセージが警告であることを示します。
public static final int	INFO	メッセージログに情報が表示されることを示します。

メソッド

変更子とタイプ	メソッド	説明
public String	getText() [32 ページ]	このメッセージに対応するテキストにアクセスします。
public int	getType() [32 ページ]	このメッセージタイプにアクセスします。
public String	getUser() [33 ページ]	このメッセージに対応するユーザ名にアクセスします。

このセクションの内容:

1.7.1 getText() メソッド

このメッセージに対応するテキストにアクセスします。

構文

```
public String getText ()
```

戻り値

このメッセージの本文

1.7.2 getType() メソッド

このメッセージタイプにアクセスします。

構文

```
public int getType ()
```

戻り値

このメッセージのタイプ (*LogMessage.ERROR*、*LogMessage.INFO*、または *LogMessage.WARNING*)。

1.7.3 getUser() メソッド

このメッセージに対応するユーザ名にアクセスします。

構文

```
public String getUser ()
```

戻り値

このメッセージに対応するユーザメッセージに対応するユーザがない場合には、この値が NULL になる場合があります。

1.8 NetworkData インタフェース

同期用のネットワークストリームの情報を含みます。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface NetworkData
```

メンバー

NetworkData のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public java.security.cert.CertPath	getCertificateChain() [36 ページ]	クライアントから送信されたすべての証明書を含む java.security.cert.CertPath オブジェクトを返します。

変更子とタイプ	メソッド	説明
public Map< String, List< String > >	getHTTPHeaders() [36 ページ]	ヘッダ名をヘッダ値のリストにマップするマップオブジェクトを返します。
public String	getHTTPHeaderValue(String) [37 ページ]	指定した名前でサーバが受信した最終ヘッダ値を返します。
public List< String >	getHTTPHeaderValues(String) [37 ページ]	指定した名前と関連付けられている、サーバが受信したすべてのヘッダ値を返します。
public boolean	isEndToEndEncrypted() [38 ページ]	同期がエンドツーエンドで暗号化されるか確認します。
public boolean	isHTTP() [38 ページ]	同期で HTTP または HTTPS のどちらが使用されるかを確認します。
public boolean	isTLS() [39 ページ]	同期で TLS が使用されるかを確認します。

備考

このインタフェースは、クライアント側証明書と HTTP ヘッダを使用するエンタープライズ内の別のサーバを認証する際に便利です。

ネットワークストリームのデータの収集を有効にするには、-x スイッチに collect_network_data=1 を追加します。このオプションにより、同期ごとのデータの保存時に、メモリのオーバヘッドが増えます。TLS または HTTPS をクライアント側証明書で使用する場合に、trusted_certificates=<certificate file> を追加すると、サーバは TLS ハンドシェイクの間に証明書を送信するようクライアントに要求します。これにより、時間とネットワークコストがかかるようになります。

NetworkData オブジェクトは、DBConnectionContext クラスの getNetworkData メソッドを呼び出して取得できます。HTTP または HTTPS を使用する場合、これには、認証スクリプトが呼び出される前にサーバが受信した、最終の HTTP 要求のヘッダデータが含まれます。

例

次に、DBConnectionContext オブジェクトから NetworkData オブジェクトを取得し、データを出力する例を示します。

```
public class OrderProcessor {
   DBConnectionContext _cc;
   public OrderProcessor( DBConnectionContext cc ) {
       _cc = cc;
   }
   // The method used for the authenticate_user event.
   public void AuthUser() {
       NetworkData nd = _cc.getNetworkData();
       if( nd != null ) {
           if( nd.isHTTP() ) {
               System.out.println( "http" );
               String user_agent = nd.getHTTPHeaderValue( "user-agent" );
               System.out.println( "  user-agent: " + user_agent.substring( 0,
user_agent.indexOf( '/' ) ) );
           } else {
               System.out.println( "no http" );
           }
           if( nd.isTLS() ) {
               System.out.println( "tls" );
               CertPath certs = nd.getCertificateChain();
```

```
                if( certs != null ) {
                    System.out.println( "  client-side cert:" );
                    int n = 1;
                    for( Certificate c : certs.getCertificates() ) {
                        System.out.println( "    cert " + n++ );
                        X509Certificate c509 = (X509Certificate) c;
                        System.out.println( "      Subject: " +
c509.getSubjectX500Principal().getName() );
                        System.out.println( "      Issuer: " +
c509.getIssuerX500Principal().getName() );
                    }
                } else {
                    System.out.println( "  no client cert" );
                }
            } else {
                System.out.println( "no tls" );
            }
            if( nd.isEndToEndEncrypted() ) {
                System.out.println( "e2ee" );
            } else {
                System.out.println( "no e2ee" );
            }
        } else {
            System.out.println( "NULL networkdata" );
        }
    }
}
```

このセクションの内容:

関連情報

DBConnectionContext インタフェース [6 ページ]

1.8.1 getCertificateChain() メソッド

クライアントから送信されたすべての証明書を含む java.security.cert.CertPath オブジェクトを返します。

構文

```
public java.security.cert.CertPath getCertificateChain ()
```

戻り値

クライアントを識別する X509 証明書を含む CertPath。そのような証明書が提供されなかった場合は NULL。

備考

すべての証明書は java.security.cert.X509Certificate オブジェクトです。

このメソッドによって NULL 以外の値が返されるのは、isTLS メソッドが true であり、クライアントが「identity」ストリームパラメータを使用して証明書を提供し、サーバ上で trusted_certificates オプションが設定される場合のみです。NULL 以外の CertPath 値には、自己署名証明書からピアの証明書の順の証明書が含まれます。

1.8.2 getHTTPHeaders() メソッド

ヘッダ名をヘッダ値のリストにマップするマップオブジェクトを返します。

構文

```
public Map< String, List< String > > getHTTPHeaders ()
```

戻り値

サーバが受信したすべてのヘッダを含むマップ。

関連情報

getHTTPHeaderValue(String) メソッド [37 ページ]

getHTTPHeaderValues(String) メソッド [37 ページ]

1.8.3 getHTTPHeaderValue(String) メソッド

指定した名前でサーバが受信した最終ヘッダ値を返します。

構文

```
public String getHTTPHeaderValue (String name)
```

パラメータ

name 値を返すヘッダの名前

戻り値

指定したヘッダ名に関連付けられている最終ヘッダ値。

関連情報

getHTTPHeaderValues(String) メソッド [37 ページ]
getHTTPHeaders() メソッド [36 ページ]

1.8.4 getHTTPHeaderValues(String) メソッド

指定した名前と関連付けられている、サーバが受信したすべてのヘッダ値を返します。

構文

```
public List< String > getHTTPHeaderValues (String name)
```

パラメータ

name 値を返すヘッダの名前

戻り値

指定したヘッダ名に関連付けられているヘッダ値。

関連情報

getHTTPHeaderValue(String) メソッド [37 ページ]
getHTTPHeaders() メソッド [36 ページ]

1.8.5 isEndToEndEncrypted() メソッド

同期がエンドツーエンドで暗号化されるか確認します。

構文

```
public boolean isEndToEndEncrypted ()
```

戻り値

同期がエンドツーエンドで暗号化されている場合は true、それ以外の場合は false を返します。

1.8.6 isHTTP() メソッド

同期で HTTP または HTTPS のどちらが使用されるかを確認します。

構文

```
public boolean isHTTP ()
```

戻り値

同期で HTTP または HTTPS が使用される場合は true、それ以外の場合は false を返します。

1.8.7 isTLS() メソッド

同期で TLS が使用されるかを確認します。

構文

```
public boolean isTLS ()
```

戻り値

同期で TLS が使用される場合は true、それ以外の場合は false を返します。

1.9 ServerContext インタフェース

同期サーバの継続期間中に存在する、すべてのコンテキストのインスタンス化。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface ServerContext
```

メンバー

ServerContext のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public void	addErrorListener(LogListener) [42 ページ]	エラーが出力されたときに通知を受信するために、指定した LogListener オブジェクトを追加します。
public void	addInfoListener(LogListener) [42 ページ]	情報が出力されたときに通知を受信するリスナオブジェクトのリストから、指定した LogListener オブジェクトを追加します。
public void	addShutdownListener(ShutdownListener) [44 ページ]	サーバコンテキストが破壊される前に通知を受信する指定の ShutdownListener オブジェクトを追加します。
public void	addWarningListener(LogListener) [44 ページ]	警告が出力されたときに通知を受信するために、指定した LogListener オブジェクトを追加します。
public Properties	getProperties(String, String) [45 ページ]	指定されたコンポーネントとプロパティセットに関連する一連のプロパティを返します。
public Properties	getPropertiesByVersion(String) [46 ページ]	スクリプトバージョンに関連する一連のプロパティを返します。
public Iterator	getPropertySetNames(String) [47 ページ]	指定したコンポーネントのプロパティセット名のリストを返します。
public Object[]	getStartClassInstances() [48 ページ]	サーバ起動時に構築された起動クラスの配列を取得します。
public java.sql.Connection	makeConnection() [48 ページ]	新しいサーバ接続を開いて、返します。
public void	removeErrorListener(LogListener) [49 ページ]	エラーが出力されたときに通知を受信するリスナオブジェクトのリストから、指定した LogListener オブジェクトを削除します。
public void	removeInfoListener(LogListener) [49 ページ]	情報が出力されたときに通知を受信するリスナオブジェクトのリストから、指定した LogListener オブジェクトを削除します。
public void	removeShutdownListener(ShutdownListener) [50 ページ]	この ServerContext が破棄される前に通知を受信するリスナオブジェクトのリストから、指定した ShutdownListener オブジェクトを削除します。
public void	removeWarningListener(LogListener) [50 ページ]	警告が出力されたときに通知を受信するリスナオブジェクトのリストから、指定した LogListener オブジェクトを削除します。
public void	shutdown() [51 ページ]	サーバを強制的に停止します。

備考

このコンテキストを静的なデータとして保持し、バックグラウンドスレッドで使用できます。同期サーバによって起動される Java VM の継続期間中は有効です。

ServerContext インスタンスにアクセスするには、*DBConnectionContext.getServerContext* メソッドを使用します。

このセクションの内容:

1.9.1 addErrorListener(LogListener) メソッド

エラーが出力されたときに通知を受信するために、指定した LogListener オブジェクトを追加します。

構文

```
public void addErrorListener (LogListener ll)
```

パラメータ

ll エラーが通知される LogListener オブジェクト。

備考

エラーが出力されたときは、*LogListener.messageLogged(ServerContext, LogMessage)* メソッドが呼び出されます。

関連情報

messageLogged(ServerContext, LogMessage) メソッド [30 ページ]
LogMessage クラス [31 ページ]

1.9.2 addInfoListener(LogListener) メソッド

情報が出力されたときに通知を受信するリスナオブジェクトのリストから、指定した LogListener オブジェクトを追加します。

構文

```
public void addInfoListener (LogListener ll)
```

パラメータ

ll 情報が通知される *LogListener* オブジェクト。

備考

LogListener.messageLogged メソッドが呼び出されます。

例

次のコードは *MyLogListener* オブジェクトを登録して、情報メッセージの通知を受信します。

```
// ServerContext serv_context;
serv_context.addInfoListener(new MyLogListener(ll_out_file));
// The following code shows an example of processing those messages:
class MyLogListener implements LogListener {
   FileOutputStream _out_file;
   public TestLogListener(FileOutputStream out_file) {
       _out_file = out_file;
   }
   public void messageLogged(ServerContext sc, LogMessage msg) {
       String type;
       String user;
       try {
           if (msg.getType() == LogMessage.ERROR) {
               type = "ERROR";
           } else if (msg.getType() == LogMessage.WARNING) {
               type = "WARNING";
           } else if (msg.getType() == LogMessage.INFO) {
               type = "INFO";
           } else {
               type = "UNKNOWN!!!";
           }
           user = msg.getUser();
           if (user == null) {
               user = "NULL";
           }
           _out_file.write(("Caught msg type="
               + type
               + " user=" + user
               + " text=" +msg.getText()
               + "¥n").getBytes()
           );
           _out_file.flush();
       }
       catch(Exception e) {
// if we print the exception from processing an info message,
// we may recurse indefinitely
if (msg.getType() != LogMessage.ERROR) {
   // Print some error output to the synchronization server log.
   e.printStackTrace();
}
       }
   }
}
```

関連情報

messageLogged(ServerContext, LogMessage) メソッド [30 ページ]

1.9.3 addShutdownListener(ShutdownListener) メソッド

サーバコンテキストが破壊される前に通知を受信する指定の ShutdownListener オブジェクトを追加します。

構文

```
public void addShutdownListener (ShutdownListener sl)
```

パラメータ

sl シャットダウン時に通知される ShutdownListener オブジェクト。

備考

シャットダウン時に、*ShutdownListener.shutdownPerformed(ServerContext)* メソッドが呼び出されます。

関連情報

ShutdownListener インタフェース [54 ページ]
shutdownPerformed(ServerContext) メソッド [55 ページ]

1.9.4 addWarningListener(LogListener) メソッド

警告が出力されたときに通知を受信するために、指定した LogListener オブジェクトを追加します。

構文

```
public void addWarningListener (LogListener ll)
```

パラメータ

ll 警告が通知される LogListener オブジェクト。

備考

LogListener.messageLogged(ServerContext, LogMessage) メソッドが呼び出されます。

関連情報

LogMessage クラス [31 ページ]
messageLogged(ServerContext, LogMessage) メソッド [30 ページ]

1.9.5 getProperties(String, String) メソッド

指定されたコンポーネントとプロパティセットに関連する一連のプロパティを返します。

構文

```
public Properties getProperties (
    String component,
    String set
)
```

パラメータ

component コンポーネント
set プロパティセット

戻り値

一連のプロパティ。空の場合があります。

備考

これらのプロパティは、*ml_property* システムテーブルに格納されます。

例

次の例では、*ServerContext* インスタンスのすべてのプロパティをリストする方法を示します。

```
import java.util.*;
ServerContext serverContext;
PrintStream out
Properties prop = serverContext.getProperties();
prop.list(out);
```

1.9.6 getPropertiesByVersion(String) メソッド

スクリプトバージョンに関連する一連のプロパティを返します。

構文

```
public Properties getPropertiesByVersion (String script_version)
```

パラメータ

script_version 関連するプロパティを返すスクリプトバージョン

戻り値

指定したスクリプトバージョンに関連する一連のプロパティ

備考

これらは、*ml_property* システムテーブルに格納されます。*component_name* が *ScriptVersion* の場合、スクリプトバージョンは *property_set_name* カラムに格納されます。

例

次の例では、指定したスクリプトバージョンに関連付けられた *ServerContext* インスタンスのすべてのプロパティをリストする方法を示します。

```
import java.util.*;
ServerContext serverContext;
PrintStream out
Properties prop = serverContext.getPropertiesByVersion("MyScriptVersion");
```

```
prop.list(out);
```

1.9.7 getPropertySetNames(String) メソッド

指定したコンポーネントのプロパティセット名のリストを返します。

構文

```
public Iterator getPropertySetNames (String component)
```

パラメータ

component プロパティ名をリストするコンポーネントの名前。

戻り値

指定したコンポーネントのプロパティセット名のリスト

備考

これらのプロパティは、*ml_property* システムテーブルに格納されます。

例

次の例では、指定したコンポーネントに関連付けられた *ServerContext* インスタンスのすべてのプロパティをリストする方法を示します。

```
import java.util.*;
ServerContext serverContext;
PrintStream out
Properties prop = serverContext.getPropertySetNames("Component Name");
prop.list(out);
```

1.9.8 getStartClassInstances() メソッド

サーバ起動時に構築された起動クラスの配列を取得します。

構文

```
public Object[] getStartClassInstances ()
```

戻り値

サーバ起動時に構築された起動クラスの配列。起動クラスがない場合は、長さが 0 の配列。

例

次の例は、*getStartClassInstances* メソッドの使用方法を示します。

```
Object objs[] = sc.getStartClassInstances();
int i;
for (i=0; i < objs.length; i += 1) {
   if (objs[i] instanceof MyClass) {
       // Use class.
   }
}
```

1.9.9 makeConnection() メソッド

新しいサーバ接続を開いて、返します。

構文

```
public java.sql.Connection makeConnection () throws SQLException
```

戻り値

新しく作成されたサーバ接続。

例外

java.sql.SQLException 新しい接続を開く間にエラーが発生した場合に発行されます。

備考

この接続は、ユーザの Java コードが所有します。ユーザがコミットし、閉じる必要があります。

サーバコンテキストにアクセスするには、現在の接続の *DBConnectionContext* の *DBConnectionContext.getServerContext* メソッドを使用します。

> **注記**
>
> 接続を開く負荷が大きくなる可能性があります。このメソッドの呼び出し数が最小になるように、ロジックを書いてください。

1.9.10 removeErrorListener(LogListener) メソッド

エラーが出力されたときに通知を受信するリスナオブジェクトのリストから、指定した LogListener オブジェクトを削除します。

構文

```
public void removeErrorListener (LogListener ll)
```

パラメータ

ll 通知が中止されるリスナオブジェクト。

例

次のコードは、エラーリスナオブジェクトのリストから *LogListener* を削除します。

```
ServerContext serverContext;
LogListener myErrorListener
serverContext.removeErrorListener(myErrorListener);
```

1.9.11 removeInfoListener(LogListener) メソッド

情報が出力されたときに通知を受信するリスナオブジェクトのリストから、指定した LogListener オブジェクトを削除します。

構文

```
public void removeInfoListener (LogListener ll)
```

パラメータ

ll 通知が中止されるリスナオブジェクト。

例

次のコードは、情報リスナオブジェクトのリストから *LogListener* を削除します。

```
ServerContext serverContext;
LogListener myInfoListener
serverContext.removeInfoListener(myInfoListener);
```

1.9.12 removeShutdownListener(ShutdownListener) メソッド

この ServerContext が破棄される前に通知を受信するリスナオブジェクトのリストから、指定した ShutdownListener オブジェクトを削除します。

構文

```
public void removeShutdownListener (ShutdownListener sl)
```

パラメータ

sl 通知が中止されるリスナオブジェクト。

例

次のコードは、*ServerContext* が破壊される前に通知を受信するリスナオブジェクトのリストから、*ShutdownListener* オブジェクトを削除します。

```
ServerContext serverContext;
ShutdownListener myShutdownListener
serverContext.removeShutdownListener(myShutdownListener);
```

1.9.13 removeWarningListener(LogListener) メソッド

警告が出力されたときに通知を受信するリスナオブジェクトのリストから、指定した LogListener オブジェクトを削除します。

構文

```
public void removeWarningListener (LogListener ll)
```

パラメータ

ll 通知が中止されるリスナオブジェクト。

例

次のコードは、警告リスナオブジェクトのリストから *LogListener* を削除します。

```
ServerContext serverContext;
LogListener myWarningListener
serverContext.removeWarningListener(myWarningListener);
```

1.9.14 shutdown() メソッド

サーバを強制的に停止します。

構文

```
public void shutdown ()
```

備考

登録された *ShutdownListener* オブジェクトで、*shutdownPerformed* メソッドが呼び出されます。

例

次のコードは、サーバを強制的に停止します。

```
ServerContext serverContext;
serverContext.shutdown();
```

関連情報

shutdownPerformed(ServerContext) メソッド [55 ページ]

1.10 ServerException クラス

サーバで同期の進行を妨げるエラー状態が存在することを示すために発行されます。

パッケージ

```
com.sap.ml.script
```

構文

```
public class ServerException
```

メンバー

ServerException のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

コンストラクタ

変更子とタイプ	コンストラクタ	説明
public	ServerException [53 ページ]	ServerException を構成します。

備考

この例外が発行されると、Mobile Link サーバはシャットダウンされます。

例

次の例では、致命的な問題が発生したときに *ServerException* をスローし、Mobile Link サーバをシャットダウンする方法を示します。

```
public void handleUpload(UploadData ud)
   throws SQLException, IOException, ServerException
{
UploadedTableData tables[] = ud.getUploadedTables();
if (tables == null) {
   throw new ServerException("Failed to read uploaded tables");
}
for (int i = 0; i < tables.length; i++) {
   UploadedTableData currentTable = tables[i];
   println("table " + java.lang.Integer.toString(i)
       + " name: " + currentTable.getName());
   // Print out delete result set.
   println("Deletes");
   printRSInfo(currentTable.getDeletes());
```

```
    // Print out insert result set.
    println("Inserts");
    printRSInfo(currentTable.getInserts());
    // print out update result set
    println("Updates");
    printUpdateRSInfo(currentTable.getUpdates());
}
}
```

このセクションの内容:

1.10.1 ServerException コンストラクタ

ServerException を構成します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public	ServerException() [53 ページ]	詳細メッセージのない ServerException を構成します。
public	ServerException(String) [54 ページ]	指定した詳細メッセージを含む ServerException を構成します。

このセクションの内容:

1.10.1.1 ServerException() コンストラクタ

詳細メッセージのない ServerException を構成します。

構文

```
public ServerException ()
```

1.10.1.2 ServerException(String) コンストラクタ

指定した詳細メッセージを含む ServerException を構成します。

構文

```
public ServerException (String s)
```

パラメータ

s 詳細メッセージ。

1.11 ShutdownListener インタフェース

サーバのシャットダウンを取得するリスナインタフェースです。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface ShutdownListener
```

メンバー

ShutdownListener のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public void	shutdownPerformed(ServerContext) [55 ページ]	サーバのシャットダウンによって ServerContext が破壊される前に起動されます。

備考

このインタフェースを使用して、同期サーバが終了する前に、スレッド、接続、およびその他のリソースがすべてクリーンアップされるようにします。

例

次に、*ServerContext* インスタンスの *ShutdownListener* オブジェクトをインストールする方法の例を示します。

```
class MyShutdownListener implements ShutdownListener {
   FileOutputStream _outFile;
   public MyShutdownListener(FileOutputStream outFile) {
       _outFile = outFile;
   }
   public void shutdownPerformed(ServerContext sc) {
       // Add shutdown code
       try {
           _outFile.write(("Shutting Down" + "¥n").getBytes());
           _outFile.flush();
       }
       catch(Exception e) {
           // Print some error output to the synchronization server log.
           e.printStackTrace();
       }
       // ...
   }
}
```

次のコードは、*MyShutdownListener* オブジェクトを登録します。クラスコンストラクタや同期スクリプトなど、*ServerContext* にアクセスできる任意の場所からこのコードを呼び出してください。

```
ServerContext serv_context;
FileOutputStream outFile;
serv_context.addShutdownListener(new MyShutdownListener(outFile));
```

このセクションの内容:

1.11.1 shutdownPerformed(ServerContext) メソッド

サーバのシャットダウンによって ServerContext が破壊される前に起動されます。

構文

```
public void shutdownPerformed (ServerContext sc)
```

パラメータ

sc シャットダウンされるサーバのコンテキスト。

1.12 SpatialUtilities クラス

空間値を操作する静的メソッドのコレクション。

パッケージ

```
com.sap.ml.script
```

構文

```
public class SpatialUtilities
```

メンバー

SpatialUtilities のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public static byte[]	createSpatialValue(int, byte[]) [57 ページ]	ダウンロード用にフォーマットされた空間値を含む新しいバイト配列を返します。
public static byte[]	getBytes(byte[]) [57 ページ]	指定したバイト配列と同じ空間データを含む新しいバイト配列を返しますが、SRID は削除されています。
public static int	getSRID(byte[]) [58 ページ]	指定した空間値の SRID を返します。
public static void	setSRID(byte[], int) [58 ページ]	指定した SRID を、指定したバイト配列の最初の 4 バイトに格納します。

このセクションの内容:

createSpatialValue(int, byte[]) メソッド [57 ページ]
ダウンロード用にフォーマットされた空間値を含む新しいバイト配列を返します。

getBytes(byte[]) メソッド [57 ページ]
指定したバイト配列と同じ空間データを含む新しいバイト配列を返しますが、SRID は削除されています。

getSRID(byte[]) メソッド [58 ページ]
指定した空間値の SRID を返します。

setSRID(byte[], int) メソッド [58 ページ]
指定した SRID を、指定したバイト配列の最初の 4 バイトに格納します。

1.12.1 createSpatialValue(int, byte[]) メソッド

ダウンロード用にフォーマットされた空間値を含む新しいバイト配列を返します。

構文

```
public static byte[] createSpatialValue (
    int srid,
    byte[] spatial_value
)
```

パラメータ

srid SRID。
spatial_value 空間データ。

戻り値

ダウンロード用にフォーマットされた空間値。

備考

最初の 4 バイトにはリトルエンディアンで指定した SRID が含まれ、残りのバイトには指定したバイト配列に渡された空間データが含まれます。

1.12.2 getBytes(byte[]) メソッド

指定したバイト配列と同じ空間データを含む新しいバイト配列を返しますが、SRID は削除されています。

構文

```
public static byte[] getBytes (byte[] spatial_value)
```

パラメータ

spatial_value SRID を削除する必要がある空間値。

戻り値

新しいバイト配列

1.12.3 getSRID(byte[]) メソッド

指定した空間値の SRID を返します。

構文

```
public static int getSRID (byte[] spatial_value)
```

パラメータ

spatial_value アップロードされた値。最初の 4 バイトには、リトルエンディアンでエンコードされた SRID が含まれている必要があります。

戻り値

SRID。

1.12.4 setSRID(byte[], int) メソッド

指定した SRID を、指定したバイト配列の最初の 4 バイトに格納します。

構文

```
public static void setSRID (
    byte[] spatial_value,
    int srid
)
```

パラメータ

spatial_value SRID を格納する配列。
srid 格納する SRID。

1.13 TimestampWithTimeZone クラス

タイムゾーンを取得および設定するメソッドでの *java.sql.Timestamp*。

パッケージ

```
com.sap.ml.script
```

構文

```
public class TimestampWithTimeZone
```

メンバー

TimestampWithTimeZone のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

変数

変更子とタイプ	変数	説明
protected static Pattern	_timestamp_pattern	
protected static Pattern	_timezone_pattern	
protected static Pattern	_time_pattern	
protected static Pattern	_date_pattern	

コンストラクタ

変更子とタイプ	コンストラクタ	説明
public	TimestampWithTimeZone(int, int, int, int, int, int, int, int, int) [61 ページ]	指定された年、月、日、時間、分、秒、ナノ、タイムゾーンの時間、タイムゾーンの分を含む新しい TimestampWithTimeZone を構成します。

メソッド

変更子とタイプ	メソッド	説明
public boolean	equals [62 ページ]	o がこのメソッドの Timestamp 部分に等しく、o がこのメソッドと同じタイムゾーンオフセットを持つ TimestampWithTimeZone であるか、o が TimestampWithTimeZone ではなくタイムゾーンオフセットが 00:00 である場合は、true を返します。
public int	getTimeZoneOffsetHours() [64 ページ]	タイムゾーンオフセットの時間の部分を取得します。
public int	getTimeZoneOffsetMinutes() [64 ページ]	タイムゾーンオフセットの分の部分を取得します。
public void	setTimeZoneOffsetHours(int) [65 ページ]	タイムゾーンオフセットの時間の部分を設定します。
public void	setTimeZoneOffsetMinutes(int) [65 ページ]	タイムゾーンオフセットの分の部分を設定します。
public String	toString() [66 ページ]	このタイムスタンプを表す文字列を yyyy-mm-dd hh:mm:ss.ffffff Shh:mm の形式で返します。S は時間フィールドの符号です。
public static TimestampWithTimeZone	toTimestampWithTimeZone(Timestamp) [66 ページ]	指定された Timestamp を TimestampWithTimeZone に変換します。
public static TimestampWithTimeZone	valueOf(String) [67 ページ]	String を TimestampWithTimeZone 値に変換します。

備考

ダイレクトロー API を使用して TIMESTAMP WITH TIME ZONE カラムのタイムゾーンオフセットを指定するときに使用します。ダイレクトロー API 以外の JDBC ドライバの *PreparedStatement* と *ResultSet* では、これが通常の *Timestamp* として処理されます。

このセクションの内容:

TimestampWithTimeZone(int, int, int, int, int, int, int, int, int) コンストラクタ [61 ページ]
: 指定された年、月、日、時間、分、秒、ナノ、タイムゾーンの時間、タイムゾーンの分を含む新しい TimestampWithTimeZone を構成します。

equals メソッド [62 ページ]
: o がこのメソッドの Timestamp 部分に等しく、o がこのメソッドと同じタイムゾーンオフセットを持つ TimestampWithTimeZone であるか、o が TimestampWithTimeZone ではなくタイムゾーンオフセットが 00:00 である場合は、true を返します。

getTimeZoneOffsetHours() メソッド [64 ページ]
: タイムゾーンオフセットの時間の部分を取得します。

getTimeZoneOffsetMinutes() メソッド [64 ページ]
: タイムゾーンオフセットの分の部分を取得します。

setTimeZoneOffsetHours(int) メソッド [65 ページ]

タイムゾーンオフセットの時間の部分を設定します。

setTimeZoneOffsetMinutes(int) メソッド [65 ページ]
タイムゾーンオフセットの分の部分を設定します。

toString() メソッド [66 ページ]
このタイムスタンプを表す文字列を yyyy-mm-dd hh:mm:ss.ffffff Shh:mm の形式で返します。S は時間フィールドの符号です。

toTimestampWithTimeZone(Timestamp) メソッド [66 ページ]
指定された Timestamp を TimestampWithTimeZone に変換します。

valueOf(String) メソッド [67 ページ]
String を TimestampWithTimeZone 値に変換します。

1.13.1 TimestampWithTimeZone(int, int, int, int, int, int, int, int, int) コンストラクタ

指定された年、月、日、時間、分、秒、ナノ、タイムゾーンの時間、タイムゾーンの分を含む新しい TimestampWithTimeZone を構成します。

構文

```
public TimestampWithTimeZone (
    int year,
    int month,
    int date,
    int hour,
    int minute,
    int second,
    int nano,
    int tz_hour,
    int tz_minute
)
```

パラメータ

year 年から 1900 を差し引いた値。
month 0 ～ 11 の範囲の整数。
date 1 ～ 31 の範囲の整数。
hour 0 ～ 23 の範囲の整数。
minute 0 ～ 59 の範囲の整数。
second 0 ～ 59 の範囲の整数。
nano 0 ～ 999,999,999 の範囲の整数。
tz_hour -14 ～ 14 の範囲の整数。
tz_minute 0 ～ 59 の範囲の整数。

例外

java.lang.IllegalArgumentException *tz_minute* が 0 ～ 59 の範囲にないか、または *tz_hour* が適切な範囲内でない場合にスローされます。

1.13.2 equals メソッド

o がこのメソッドの Timestamp 部分に等しく、o がこのメソッドと同じタイムゾーンオフセットを持つ TimestampWithTimeZone であるか、o が TimestampWithTimeZone ではなくタイムゾーンオフセットが 00:00 である場合は、true を返します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public boolean	equals(Object) [63 ページ]	o がこのメソッドの Timestamp 部分に等しく、o がこのメソッドと同じタイムゾーンオフセットを持つ TimestampWithTimeZone であるか、o が TimestampWithTimeZone ではなくタイムゾーンオフセットが 00:00 である場合は、true を返します。
public boolean	equals(Timestamp) [63 ページ]	o がこのメソッドの Timestamp 部分に等しく、o がこのメソッドと同じタイムゾーンオフセットを持つ TimestampWithTimeZone であるか、o が TimestampWithTimeZone ではなくタイムゾーンオフセットが 00:00 である場合は、true を返します。

このセクションの内容:

1.13.2.1 equals(Object) メソッド

o がこのメソッドの Timestamp 部分に等しく、o がこのメソッドと同じタイムゾーンオフセットを持つ TimestampWithTimeZone であるか、o が TimestampWithTimeZone ではなくタイムゾーンオフセットが 00:00 である場合は、true を返します。

構文

```
public boolean equals (Object o)
```

パラメータ

o 比較元のオブジェクト。

戻り値

o がこれに等しい場合は true、そうでない場合は false。

関連情報

equals(Timestamp) メソッド [63 ページ]

1.13.2.2 equals(Timestamp) メソッド

o がこのメソッドの Timestamp 部分に等しく、o がこのメソッドと同じタイムゾーンオフセットを持つ TimestampWithTimeZone であるか、o が TimestampWithTimeZone ではなくタイムゾーンオフセットが 00:00 である場合は、true を返します。

構文

```
public boolean equals (Timestamp o)
```

パラメータ

o 比較元のオブジェクト。

戻り値

o がこれに等しい場合は true、そうでない場合は false。

関連情報

equals(Object) メソッド [63 ページ]

1.13.3 getTimeZoneOffsetHours() メソッド

タイムゾーンオフセットの時間の部分を取得します。

構文

```
public int getTimeZoneOffsetHours ()
```

戻り値

タイムゾーンオフセットの時間の部分。

1.13.4 getTimeZoneOffsetMinutes() メソッド

タイムゾーンオフセットの分の部分を取得します。

構文

```
public int getTimeZoneOffsetMinutes ()
```

戻り値

タイムゾーンオフセットの分の部分。

1.13.5 setTimeZoneOffsetHours(int) メソッド

タイムゾーンオフセットの時間の部分を設定します。

構文

```
public void setTimeZoneOffsetHours (int tz_hour)
```

パラメータ

tz_hour タイムゾーンオフセットの新しい時間の部分。

例外

java.lang.IllegalArgumentException *tz_hour* が -14:00 ～ +14:00 の範囲にない場合にスローされます。

1.13.6 setTimeZoneOffsetMinutes(int) メソッド

タイムゾーンオフセットの分の部分を設定します。

構文

```
public void setTimeZoneOffsetMinutes (int tz_minute)
```

パラメータ

tz_minute タイムゾーンオフセットの新しい分の部分。

例外

java.lang.IllegalArgumentException *tz_minute* が 0:00 ～ 59:00 の範囲にない場合にスローされます。

1.13.7 toString() メソッド

このタイムスタンプを表す文字列を yyyy-mm-dd hh:mm:ss.ffffff Shh:mm の形式で返します。S は時間フィールドの符号です。

構文

```
public String toString ()
```

戻り値

このタイムスタンプを表す文字列。

1.13.8 toTimestampWithTimeZone(Timestamp) メソッド

指定された Timestamp を TimestampWithTimeZone に変換します。

構文

```
public static TimestampWithTimeZone toTimestampWithTimeZone (Timestamp ts)
```

パラメータ

ts 変換するタイムスタンプ。

戻り値

ts が *TimestampWithTimeZone* のインスタンスの場合は、*ts* を返します。それ以外の場合は、タイムゾーンオフセットが 00:00 である *ts* と同等の新しい *TimestampWithTimeZone* を構築します。

1.13.9 valueOf(String) メソッド

String を TimestampWithTimeZone 値に変換します。

構文

```
public static TimestampWithTimeZone valueOf (String val)
```

パラメータ

val *yyyy-mm-dd hh:mm:ss[.f...][[-+]hh:mm]* 形式によるタイムゾーンでのタイムスタンプ小数点部分とタイムゾーンの部分は省略できます。タイムゾーンを指定する場合、符号は省略できます。

戻り値

新しい TimestampWithTimeZone。

例外

java.lang.IllegalArgumentException *val* が正しいフォーマットでない場合。

1.14 UpdateResultSet インタフェース

指定した行の更新前イメージ値 (古い値) と更新後イメージ値 (新しい値) にアクセスするための特別なメソッドを含む結果セットオブジェクトです。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface UpdateResultSet extends
```

メンバー

UpdateResultSet のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public void	setNewRowValues() [68 ページ]	新しいカラム値 (更新後のロー) を返すように、この結果セットのモードを設定します。
public void	setOldRowValues() [69 ページ]	古いカラム値 (更新前のロー) を返すように、この結果セットのモードを設定します。

備考

1 つのテーブルの 1 つのアップロードトランザクションのための更新操作を保持できます。

ResultSet のモードを変更することで、新しいローにも古いローにもアクセスできます。

UpdateResultSet のインスタンスを取得するには、*DownloadTableData.getUpdates* メソッドを使用します。

UpdateResultSet は *java.sql.ResultSet* を拡張して、*setNewRowValues* メソッドと *setOldRowValues* メソッドを追加しています。それ以外の場合は、通常の *ResultSet* と同様に使用できます。

java.sql.ResultSet の詳細については、Java Software Development Kit マニュアルを参照してください。

このセクションの内容:

関連情報

getUpdates() メソッド [79 ページ]

1.14.1 setNewRowValues() メソッド

新しいカラム値 (更新後のロー) を返すように、この結果セットのモードを設定します。

構文

```
public void setNewRowValues ()
```

備考

結果セットは、リモートクライアントデータベース内の最新の更新値を表します。

これがデフォルトモードです。

例

次のコードは、新しいカラム値を返すように、*UpdateResultSet* のモードを設定する方法を示します。

```
UpdateResultSet results
results.setNewRowValues();
```

1.14.2 setOldRowValues() メソッド

古いカラム値 (更新前のロー) を返すように、この結果セットのモードを設定します。

構文

```
public void setOldRowValues ()
```

備考

このモードでは、*UpdateResultSet* は、最後の同期中にクライアントが取得した古いカラム値を表します。

例

次のコードは、古いカラム値を返すように、*UpdateResultSet* のモードを設定する方法を示します。

```
UpdateResultSet results
results.setOldRowValues();
```

1.15 UploadData インタフェース

ダイレクトローハンドリングで使用するアップロード操作をカプセル化します。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface UploadData
```

メンバー

UploadData のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public UploadedTableData	getUploadedTableByName(String) [71 ページ]	指定されたテーブルを表す UploadedTableData インスタンスを返します。
public UploadedTableData[]	getUploadedTables() [72 ページ]	現在の同期の UploadedTableData オブジェクトの配列を返します。

備考

単一のアップロードトランザクションを表す *UploadData* インスタンスが、handle_UploadData イベントに渡されます。

注記

ダイレクトローハンドリングのアップロード操作は、handle_UploadData イベントに対して登録したメソッドで処理してください。登録されたメソッドを呼び出した後、*UploadData* は破棄されます。後続のイベントで使用するために新しい *UploadData* インスタンスを作成しないでください。

UploadedTableData インスタンスを取得するには、*getUploadedTables* メソッドまたは *getUploadedTableByName* メソッドを使用します。

リモートデータベースがトランザクションアップロードを使用している場合を除き、同期の *UploadData* は 1 つです。

このセクションの内容:

関連情報

UploadedTableData インタフェース [73 ページ]

1.15.1 getUploadedTableByName(String) メソッド

指定されたテーブルを表す UploadedTableData インスタンスを返します。

構文

```
public UploadedTableData getUploadedTableByName (String table_name)
```

パラメータ

table_name アップロードデータの取得先アップロードテーブルの名前

戻り値

指定されたテーブルを表す *UploadedTableData* インスタンス。指定された名前のテーブルが現在の同期で存在しない場合は NULL。

例

handle_UploadData イベントに対して *HandleUpload* というメソッドを使用するものとします。次の例では *GetUploadedTableByName* メソッドを使用して、*remoteOrderstable* の *UploadedTableData* インスタンスを返します。

```
public void handleUpload(UploadData ud)
   throws SQLException, IOException
{
   UploadedTableData uploaded_t1 = ud.GetUploadedTableByName("remoteOrders");
   //...
}
```

関連情報

UploadedTableData インタフェース [73 ページ]

1.15.2 getUploadedTables() メソッド

現在の同期の UploadedTableData オブジェクトの配列を返します。

構文

```
public UploadedTableData[] getUploadedTables ()
```

戻り値

現在の同期の *UploadedTableData* オブジェクトの配列配列内でのテーブルの順序は、クライアントのアップロード順と同じです。

備考

テーブルの配列内での順序は、同期サーバによる SQL のローハンドリングでの順序と同じで、参照整合性違反を防ぐ最適な順序になります。データソースがリレーショナルデータベースの場合は、このテーブル順序を使用してください。

例

handle_UploadData イベントに対して *HandleUpload* というメソッドを使用するものとします。次の例では *getUploadedTables* メソッドを使用して、現在のアップロードトランザクションの *UploadedTableData* インスタンスを返します。

```
public void handleUpload(UploadData ud)
    throws SQLException, IOException
{
    UploadedTableData tables[] = ud.getUploadedTables();
    //...
}
```

関連情報

UploadedTableData インタフェース [73 ページ]

1.16 UploadedTableData インタフェース

ダイレクトローハンドリングアップロードで使用するテーブル操作をカプセル化します。

パッケージ

```
com.sap.ml.script
```

構文

```
public interface UploadedTableData
```

メンバー

UploadedTableData のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public java.sql.ResultSet	getDeletes() [75 ページ]	同期クライアントによってアップロードされた削除操作を表す java.sql.ResultSet オブジェクトを返します。
public java.sql.ResultSet	getInserts() [76 ページ]	同期クライアントによってアップロードされた挿入操作を表す java.sql.ResultSet オブジェクトを返します。
public java.sql.ResultSetMetaData	getMetaData() [77 ページ]	UploadedTableData インスタンスのメタデータを取得します。
public String	getName() [78 ページ]	UploadedTableData インスタンスのテーブル名を返します。
public UpdateResultSet	getUpdates() [79 ページ]	同期クライアントによってアップロードされた更新操作を表す UpdateResultSet オブジェクトを返します。

備考

UploadedTableData インスタンスを使用して、単一アップロードトランザクションに対するテーブルの挿入、更新、削除操作を取得できます。*getInserts*、*getUpdates*、*getDeletes* メソッドを使用して、標準 JDBC *java.sql.ResultSet* オブジェクトを返します。

java.sql.ResultSet と *java.sql.ResultSetMetaData* の詳細については、Java Software Development Kit マニュアルを参照してください。

getMetaData メソッドを使用するか、*getInserts*、*getUpdates*、*getDeletes* によって返された結果セットを使用してテーブルメタデータにアクセスできます。削除の結果セットには、テーブルのプライマリキーカラムのみが含まれています。

例

次のコードでは、アップロードされた削除を取得し、それぞれの最初のカラムを表示します。

```
void printFirstColOfDeletes( UploadedTableData tab_data )
{
   ResultSet deletes = tab_data.getDeletes();
   while( deletes.next() ) {
       java.lang.System.out.println( deletes.getString( 1 ) );
   }
   deletes.close();
}
```

次のコードでは、それぞれの更新の 3 番目のカラムの古い値と新しい値を表示します。

```
void printThirdColOfUpdates( UploadedTableData tab_data )
{
   ResultSet updates = tab_data.getUpdates();
   while( updates.next() ) {
       updates.setOldRowValues();
       java.lang.System.out.println( "old row col: " + updates.getString( 3 ) );
       updates.setNewRowValues();
       java.lang.System.out.println( "new row col: " + updates.getString( 3 ) );
   }
   updates.close();
}
```

このセクションの内容:

関連情報

UploadData インタフェース [70 ページ]

1.16.1 getDeletes() メソッド

同期クライアントによってアップロードされた削除操作を表す java.sql.ResultSet オブジェクトを返します。

構文

```
public java.sql.ResultSet getDeletes ()
```

戻り値

同期クライアントによってアップロードされた削除操作を表す java.sql.ResultSet オブジェクト

備考

結果セットには、削除されたローのプライマリキー値が含まれています。

例

リモートクライアントには *remoteOrders* というテーブルがあるものとします。次の例は *DownloadTableData.getDeletes* メソッドを使用して、削除されたローの結果セットを取得します。この場合、削除結果セットには 1 つのプライマリキーカラムが含まれています。

```
import ianywhere.ml.script.*;
import java.sql.*;
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut)
   throws SQLException, IOException
{
   // Get an UploadedTableData for the remoteOrders table.
   UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");
   // Get deletes uploaded by the synchronization client.
   java.sql.ResultSet delete_rs = remoteOrdersTable.getDeletes();
    while (delete_rs.next()) {
        // Get primary key values for deleted rows.
        int deleted_id = delete_rs.getInt(1);
        // ...
   }
   delete_rs.close();
}
```

1.16.2 getInserts() メソッド

同期クライアントによってアップロードされた挿入操作を表す java.sql.ResultSet オブジェクトを返します。

構文

```
public java.sql.ResultSet getInserts ()
```

戻り値

同期クライアントによってアップロードされた挿入操作を表す *java.sql.ResultSet* オブジェクト

備考

各挿入は結果セットの 1 つのローで表されています。

例

リモートクライアントには *remoteOrders* というテーブルがあるものとします。次の例は *DownloadTableData.getInserts* メソッドを使用して、挿入されたローの結果セットを取得します。このコードは、現在のアップロードトランザクションの各ローに対する発注額を取得します。

```
import ianywhere.ml.script.*;
import java.sql.*;
// The method used for the handle_UploadData event
public void HandleUpload(UploadData ut)
   throws SQLException, IOException
{
   // Get an UploadedTableData instance representing the remoteOrders table.
   UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");
   // Get inserts uploaded by the synchronization client.
   java.sql.ResultSet rs = remoteOrdersTable.getInserts();
   while (rs.next()) {
       // get the uploaded order_amount
       double order_amount = rs.getDouble("order_amount");
       // ...
   }
   rs.close();
}
```

1.16.3 getMetaData() メソッド

UploadedTableData インスタンスのメタデータを取得します。

構文

```
public java.sql.ResultSetMetaData getMetaData ()
```

戻り値

UploadedTableData インスタンスのメタデータ

備考

メタデータは標準 *java.sql.ResultSetMetaData* インスタンスです。

ResultSetMetaData にカラム名情報を含める場合は、カラム名を送信するためのクライアント拡張オプションを指定してください。

java.sql.ResultSetMetaData の詳細については、Java Software Development Kit マニュアルを参照してください。

例

次の例は、*remoteOrders* というアップロードされたテーブルの *java.sql.ResultSetMetaData* インスタンスを取得します。このコードは *ResultSetMetaData.getColumnCount* と *getColumnLabel* メソッドを使用して、カラム名のリストをコンパイルします。

```
import ianywhere.ml.script.*;
import java.sql.*;
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut) {
   throws SQLException, IOException
{
   // Get an UploadedTableData instance representing the remoteOrders table.
   UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");
   // get inserts uploaded by the synchronization client
   java.sql.ResultSet rs = remoteOrdersTable.getInserts();
   // Obtain the result set metadata.
   java.sql.ResultSetMetaData md = rs.getMetaData();
   String columnHeading = "";
   // Compile a list of column names.
   for (int c=1; c <= md.getColumnCount(); c += 1) {
       columnHeading += md.getColumnLabel( c );
       if (c < md.getColumnCount()) {
           columnHeading += ", ";
       }
   }
   //...
}
```

この場合、*HandleUpload* というメソッドが handle_UploadData 同期イベントを処理します。

1.16.4 getName() メソッド

UploadedTableData インスタンスのテーブル名を返します。

構文

```
public String getName ()
```

戻り値

UploadedTableData インスタンスのテーブル名

備考

getMetaData メソッドによって返される *java.sql.ResultSetMetaData* インスタンスを使用して、テーブル名にアクセスすることもできます。

例

次の例は、単一アップロードトランザクションのアップロードされた各テーブルの名前を取得します。

```
import ianywhere.ml.script.*;
import java.sql.*;
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ud) {
   throws SQLException, IOException
{
   int i;
   // Get UploadedTableData instances.
   UploadedTableData tables[] = ud.getUploadedTables();
   for (i=0; i<tables.length; i+=1) {
       // Get the table name.
       String table_name = tables[i].getName();
       // ...
   }
}
```

関連情報

getMetaData() メソッド [77 ページ]

1.16.5 getUpdates() メソッド

同期クライアントによってアップロードされた更新操作を表す UpdateResultSet オブジェクトを返します。

構文

```
public UpdateResultSet getUpdates ()
```

戻り値

同期クライアントによってアップロードされた更新操作を表す *UpdateResultSet* オブジェクト

備考

各更新は、すべてのカラム値を含む 1 つのローで表されています。*UpdateResultSet* は *java.sql.ResultSet* を拡張して、同期での競合検出用の特別なメソッドを追加しています。

例

リモートクライアントには *remoteOrders* というテーブルがあるものとします。次の例は *getUpdates* メソッドを使用して、更新されたローの結果セットを取得します。このコードは各ローの発注額を取得します。

```
import ianywhere.ml.script.*;
import java.sql.*;
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut)
   throws SQLException, IOException
{
   // Get an UploadedTableData instance representing the remoteOrders table.
   UploadedTableData remoteOrdersTable =
ut.getUploadedTableByName("remoteOrders");
   // Get inserts uploaded by the synchronization client.
   java.sql.ResultSet rs = remoteOrdersTable.getUpdates();
   while (rs.next()) {
       // Get the uploaded order_amount.
       double order_amount = rs.getDouble("order_amount");
       // ...
   }
   rs.close();
}
```

関連情報

UpdateResultSet インタフェース [67 ページ]

2 このマニュアルの印刷、再生、および再配布

次の条件に従うかぎり、このマニュアルの全部または一部を使用、印刷、再生、配布することができます。

1. ここに示したものとそれ以外のすべての版権と商標の表示をすべてのコピーに含めること。
2. マニュアルに変更を加えないこと。
3. SAP 以外の人間がマニュアルの著者または情報源であるかのように示す一切の行為をしないこと。

ここに記載された情報は事前の通知なしに変更されることがあります。

重要免責事項および法的情報

コードサンプル

この文書に含まれるソフトウェアコード及び / 又はコードライン / 文字列 (「コード」) はすべてサンプルとしてのみ提供されるものであり、本稼動システム環境で使用することが目的ではありません。「コード」は、特定のコードの構文及び表現規則を分かりやすく説明及び視覚化することのみを目的としています。SAP は、この文書に記載される「コード」の正確性及び完全性の保証を行いません。更に、SAP は、「コード」の使用により発生したエラー又は損害が SAP の故意又は重大な過失が原因で発生させたものでない限り、そのエラー又は損害に対して一切責任を負いません。

アクセシビリティ

この SAP 文書に含まれる情報は、公開日現在のアクセシビリティ基準に関する SAP の最新の見解を表明するものであり、ソフトウェア製品のアクセシビリティ機能の確実な提供方法に関する拘束力のあるガイドラインとして意図されるものではありません。SAP は、この文書に関する一切の責任を明確に放棄するものです。ただし、この免責事項は、SAP の意図的な違法行為または重大な過失による場合は、適用されません。さらに、この文書により SAP の直接的または間接的な契約上の義務が発生することは一切ありません。

ジェンダーニュートラルな表現

SAP 文書では、可能な限りジェンダーニュートラルな表現を使用しています。文脈により、文書の読者は「あなた」と直接的な呼ばれ方をされたり、ジェンダーニュートラルな名詞 (例: 「販売員」又は「勤務日数」) で表現されます。ただし、男女両方を指すとき、三人称単数形の使用が避けられない又はジェンダーニュートラルな名詞が存在しない場合、SAP はその名詞又は代名詞の男性形を使用する権利を有します。これは、文書を分かりやすくするためです。

インターネットハイパーリンク

SAP 文書にはインターネットへのハイパーリンクが含まれる場合があります。これらのハイパーリンクは、関連情報を見いだすヒントを提供することが目的です。SAP は、この関連情報の可用性や正確性又はこの情報が特定の目的に役立つことの保証は行いません。SAP は、関連情報の使用により発生した損害が、SAP の重大な過失又は意図的な違法行為が原因で発生したものでない限り、その損害に対して一切責任を負いません。すべてのリンクは、透明性を目的に分類されています (http://help.sap.com/disclaimer を参照)。

go.sap.com/registration/contact.html