Package Summary		Page
edu.ou.weinmann.repsi.controller.script	Provides the command line interface to the REPSI tool.	1
edu.ou.weinmann.repsi.model.calibration	Provides the calibration functionality.	2
edu.ou.weinmann.repsi.model.database	Provides the functionality to modify schema or instance of a relational database system.	5
edu.ou.weinmann.repsi.model.mapper	Provides the functionality to map the data from an object to the database.	9
edu.ou.weinmann.repsi.model.trial	Provides the trial run functionality.	22
edu.ou.weinmann.repsi.model.trial.metadata	Provides classes to manage the database metadata.	24
edu.ou.weinmann.repsi.model.trial.util	Provides the utility classes for the trial run component of the REPSI tool.	30
edu.ou.weinmann.repsi.model.util	Provides the utility classes for the model component of the REPSI tool.	35

Package edu.ou.weinmann.repsi.controller.script

Provides the command line interface to the REPSI tool.

See:
    Description

Class Summary		Page
Main	Command line interface to the REPSI tool.	1

Package edu.ou.weinmann.repsi.controller.script Description

Provides the command line interface to the REPSI tool.

Class Main

edu.ou.weinmann.repsi.controller.script

```
java.lang.Object
  └edu.ou.weinmann.repsi.controller.script.Main
```

final public class **Main**
extends Object

Command line interface to the REPSI tool. The interface is based on the Jakarta Commons CLI library which simplifies the handling of command line parameters. The main parameter is called `mode` and this parameter determines the functionality to be executed:

- `calibration` perform a calibration run.
- `master` modify the schema or instance of the master database.
- `result_cn` export the results of a calibration run into an Excel file.
- `result_r` export the results of a calibration run into a R compatible files.
- `result_tl` export the results of one or all trial runs into an Excel file.
- `test modify` the schema or instance of a test database.
- t`rial` perform a trial run.

Further details can be found in the document 'REPSI Tool User Manual'.

Author:
Walter Weinmann

Method Summary		Page
`static void`	`main(String[] args)` Command line interface to the REPSI tool.	2

Method Detail

main

```
public static void main(String[] args)
```

Command line interface to the REPSI tool. The interface is based on the Jakarta Commons CLI library which simplifies the handling of command line parameters. The main parameter is called `mode` and this parameter determines the functionality to be executed:

- `calibration` perform a calibration run.
- `master` modify the schema or instance of the master database.
- `result_cn` export the results of a calibration run into an Excel file.
- `result_r` export the results of a calibration run into a R compatible files.
- `result_tl` export the results of one or all trial runs into an Excel file.
- `test modify` the schema or instance of a test database.
- t`rial` perform a trial run.

Further details can be found in the document 'REPSI Tool User Manual'.
Parameters:
`args` - here not supported

Package edu.ou.weinmann.repsi.model.calibration

Provides the calibration functionality.

See:
Description

Class Summary		Page
Calibration	Manages the calibration functionality of the REPSI tool: • executes a calibration run with the method `System.nanoTime`, • executes a calibration run with a SQL test query pair, • extracts the results of all calibration runs from the database into an Excel file.	3

Package edu.ou.weinmann.repsi.model.calibration Description

Provides the calibration functionality.

Class Calibration

edu.ou.weinmann.repsi.model.calibration

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.calibration.Calibration
```

final public class **Calibration**
extends Object

Manages the calibration functionality of the REPSI tool:

- executes a calibration run with the method `System.nanoTime`,
- executes a calibration run with a SQL test query pair,
- extracts the results of all calibration runs from the database into an Excel file.

Author:
Walter Weinmann

Constructor Summary	Page
Calibration() Constructs a `Calibration` object.	3
Calibration(String parProprtiesFilename, boolean parPropertiesXml) Constructs a `Calibration` object.	4

Method Summary		Page
boolean	**calibrateQuery**(int parTestQueryPairId, int parCycles, int parDatabaseInstanceId, String parDescription, boolean parAlternating, boolean parConsecutive, int parFetchSize, boolean parIgnoreFirst, long parPrecision, boolean parVerbose) Calibrates the execution of a pair of SQL statements.	4
boolean	**calibrateSimpleMethod**(String parObject, int parCycles, int parDatabaseInstanceId, String parDescription, boolean parIgnoreFirst, long parPrecision, boolean parVerbose) Calibrates the execution of a simple Java method.	5
boolean	**calibrationDataToExcel**(String parFileName) Exports all the data of a callibration run from the master database into an Excel file.	4
boolean	**calibrationDataToR**(String parDirectoryNameIn) Exports all the data of a callibration run from the master database into an R style command files.	4
String	**getObject**() Returns the type of the `Calibration` object.	5
void	**setObject**(String parObject) Sets the type of the `Calibration` object.	5

Constructor Detail

Calibration

```
public Calibration()
```

Constructs a `Calibration` object. The name of the used properties file is taken from mthe class `edu.ou.weinmann.repsi.model.util.Global` and does not constitute an XML document.

Calibration

```
public Calibration(String parProprtiesFilename,
                   boolean parPropertiesXml)
```

Constructs a `Calibration` object.

Method Detail

calibrationDataToExcel

```
public boolean calibrationDataToExcel(String parFileName)
```

Exports all the data of a callibration run from the master database into an Excel file.
Parameters:
`parFileName` - The complete file name of the Excel file including the directory.
Returns:
`true` if the calibration data were exported without any error, and `false` otherwise.

calibrationDataToR

```
public boolean calibrationDataToR(String parDirectoryNameIn)
```

Exports all the data of a callibration run from the master database into an R style command files.
Parameters:
`parDirectoryNameIn` - The directory to store the R command files.
Returns:
`true` if the command files were created without any error, and `false` otherwise.

calibrateQuery

```
public boolean calibrateQuery(int parTestQueryPairId,
                              int parCycles,
                              int parDatabaseInstanceId,
                              String parDescription,
                              boolean parAlternating,
                              boolean parConsecutive,
                              int parFetchSize,
                              boolean parIgnoreFirst,
                              long parPrecision,
                              boolean parVerbose)
```

Calibrates the execution of a pair of SQL statements.
Parameters:
`parTestQueryPairId` - The identification of the test query pair to be calibrated.
`parCycles` - The number of cycles to be executed.
`parDatabaseInstanceId` - The identification of the used test database instance.
`parDescription` - The description of the calibration run.
`parAlternating` - Whether in every cycle should be performed alternatively the execution of the first query and then of the second query.
`parConsecutive` - Whether first all cycles should be done with the first query and afterwards the same number of cycles with the second query.
`parFetchSize` - The fetch size.
`parIgnoreFirst` - Whether the first measured response time (reading) should be ignored.

`parPrecision` - The exponent (base 10) of the required precision for the response time. Calibrates the execution of a simple Java method.
`parVerbose` - Whether to print a statistical overview.

Returns:
`true` if the processing ended without any error, and `false` otherwise.

---

calibrateSimpleMethod

```
public boolean calibrateSimpleMethod(String parObject,
                                     int parCycles,
                                     int parDatabaseInstanceId,
                                     String parDescription,
                                     boolean parIgnoreFirst,
                                     long parPrecision,
                                     boolean parVerbose)
```

Calibrates the execution of a simple Java method.

Parameters:
`parObject` - The specification of the Java method to be calibrated.
`parCycles` - The number of cycles to be executed.
`parDatabaseInstanceId` - The identification of the used test database instance.
`parDescription` - The description of the calibration run.
`parIgnoreFirst` - Whether the first measured response time (reading) should be ignored.
`parPrecision` - The exponent (base 10) of the required precision for the response time.
`parVerbose` - Whether to print a statistical overview.

Returns:
`true` if the processing ended without any error, and `false` otherwise.

---

getObject

```
public String getObject()
```

Returns the type of the `Calibration` object.

Returns:
the type of the `Calibration` object.

---

setObject

```
public void setObject(String parObject)
```

Sets the type of the `Calibration` object.

Parameters:
`parObject` - The type of the `Calibration` object.

Package edu.ou.weinmann.repsi.model.database

Provides the functionality to modify schema or instance of a relational database system.

See:
Description

Class Summary		*Page*
Database	Manages the functionality to modify the schema or instance of a relational database.	*6*

Package edu.ou.weinmann.repsi.model.database Description

Provides the functionality to modify schema or instance of a relational database system.

Class Database

edu.ou.weinmann.repsi.model.database

```
java.lang.Object
  └org.xml.sax.helpers.DefaultHandler
      └edu.ou.weinmann.repsi.model.database.Database
```

All Implemented Interfaces:

org.xml.sax.ContentHandler, org.xml.sax.DTDHandler, org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler

public class **Database**
extends org.xml.sax.helpers.DefaultHandler

Manages the functionality to modify the schema or instance of a relational database.

Author:

Walter Weinmann

Constructor Summary	Page
`Database()` Constructs a `Database` object.	7
`Database(String parProprtiesFilename, boolean parPropertiesXml)` Constructs a `Database` object.	7

Method Summary		Page
`final void`	`characters(char[] parCharacters, int parStart, int parLength)` Receive notification of character data inside an element.	7
`final void`	`endElement(String parUri, String parLocalName, String parQualifiedName)` Receive notification of the end of an element.	8
`final boolean`	`modifyMasterDatabase(String parFileName, String parFileType)` Modifies the master database schema or instance based on an Excel file, a flat file or an XML document containing SQL statements.	8
`final boolean`	`modifyTestDatabase(int parDatabaseInstanceId, String parFileName, String parFileType)` Modifies a test database schema or instance based on an Excel file, a flat file or an XML document containing SQL statements.	8
`final void`	`setSqlSyntaxSource(String parSqlSyntaxCode)` Sets the type of the SQL syntax version.	9
`final void`	`startElement(String parUri, String parLocalName, String parQualifiedName, org.xml.sax.Attributes parAttributes)` Receive notification of the start of an element.	9

Methods inherited from class org.xml.sax.helpers.DefaultHandler
`characters, endDocument, endElement, endPrefixMapping, error, fatalError, ignorableWhitespace, notationDecl, processingInstruction, resolveEntity, setDocumentLocator,`

`skippedEntity, startDocument, startElement, startPrefixMapping, unparsedEntityDecl, warning`

Methods inherited from interface org.xml.sax.EntityResolver
`resolveEntity`

Methods inherited from interface org.xml.sax.DTDHandler
`notationDecl, unparsedEntityDecl`

Methods inherited from interface org.xml.sax.ContentHandler
`characters, endDocument, endElement, endPrefixMapping, ignorableWhitespace, processingInstruction, setDocumentLocator, skippedEntity, startDocument, startElement, startPrefixMapping`

Methods inherited from interface org.xml.sax.ErrorHandler
`error, fatalError, warning`

Constructor Detail

Database

```
public Database()
```

Constructs a `Database` object. The name of the used properties file is taken from mthe class `edu.ou.weinmann.repsi.model.util.Global` and does not constitute an XML document.

Database

```
public Database(String parProprtiesFilename,
                boolean parPropertiesXml)
```

Constructs a `Database` object.

Method Detail

characters

```
public final void characters(char[] parCharacters,
                             int parStart,
                             int parLength)
                      throws org.xml.sax.SAXException
```

Receive notification of character data inside an element.
Specified by:
`characters` in interface `org.xml.sax.ContentHandler`
Overrides:
`characters` in class `org.xml.sax.helpers.DefaultHandler`
Parameters:
`parCharacters` - The characters from the element.
`parStart` - The start position in the character array.
`parLength` - The number of characters to use from the character array.
Throws:
`org.xml.sax.SAXException` - Any SAX exception, possibly wrapping another exception.

See Also:
`org.xml.sax.helpers.DefaultHandler.characters(char[], int, int)`

---

endElement

```
public final void endElement(String parUri,
                             String parLocalName,
                             String parQualifiedName)
                      throws org.xml.sax.SAXException
```

Receive notification of the end of an element.
Specified by:
`endElement` in interface `org.xml.sax.ContentHandler`
Overrides:
`endElement` in class `org.xml.sax.helpers.DefaultHandler`
Parameters:
`parUri` - The Namespace URI, or the empty string if the element has no Namespace URI or if Namespace processing is not being performed.
`parLocalName` - The local name (without prefix), or the empty string if Namespace processing is not being performed.
`parQualifiedName` - The qualified name (with prefix), or the empty string if qualified names are not available.
Throws:
`org.xml.sax.SAXException` - any SAX exception, possibly wrapping another exception
See Also:
`org.xml.sax.helpers.DefaultHandler.endElement(java.lang.String, java.lang.String, java.lang.String)`

---

modifyMasterDatabase

```
public final boolean modifyMasterDatabase(String parFileName,
                                          String parFileType)
```

Modifies the master database schema or instance based on an Excel file, a flat file or an XML document containing SQL statements. Reads the SQL statements from an Excel file, a flat file or an XML document and processes the contained data by considering the SQL syntax property related to the SQL statements in the input source and the SQL syntax property related to the master database which is defined in the file `configuration.properties`.
Parameters:
`parFileName` - The complete file name including the directory.
`parFileType` - The file type: xls if the input source consists of an Excel file, xml if the input source consists of an XML document and anything else if the imput source consists of a simple flat file.
Returns:
`true` if the processing of the file was completed without any error, and `false` otherwise.

---

modifyTestDatabase

```
public final boolean modifyTestDatabase(int parDatabaseInstanceId,
                                        String parFileName,
                                        String parFileType)
```

Modifies a test database schema or instance based on an Excel file, a flat file or an XML document containing SQL statements. Reads the SQL statements from an Excel file, a flat file or an XML document and processes the contained data by considering the SQL syntax property related to the SQL statements in the input source and the SQL syntax property related to the master database which is defined in the file `configuration.properties`.

Parameters:

`parDatabaseInstanceId` - The identification of the test database instance.
`parFileName` - The complete file name including the directory.
`parFileType` - The file type: xls if the input source consists of an Excel file, xml if the input source consists of an XML document and anything else if the imput source consists of a simple flat file.

Returns:

`true` if the processing of the file was completed without any error, and `false` otherwise.

---

setSqlSyntaxSource

```
public final void setSqlSyntaxSource(String parSqlSyntaxCode)
```

Sets the type of the SQL syntax version.

Parameters:

`parSqlSyntaxCode` - The type of the SQL syntax version.

---

startElement

```
public final void startElement(String parUri,
                               String parLocalName,
                               String parQualifiedName,
                               org.xml.sax.Attributes parAttributes)
                        throws org.xml.sax.SAXException
```

Receive notification of the start of an element.

Specified by:

`startElement` in interface `org.xml.sax.ContentHandler`

Overrides:

`startElement` in class `org.xml.sax.helpers.DefaultHandler`

Parameters:

`parUri` - The Namespace URI, or the empty string if the element has no Namespace URI or if Namespace processing is not being performed.
`parLocalName` - The local name (without prefix), or the empty string if Namespace processing is not being performed.
`parQualifiedName` - The qualified name (with prefix), or the empty string if qualified names are not available.
`parAttributes` - The attributes attached to the element. If there are no attributes, it shall be an empty Attributes object.

Throws:

`org.xml.sax.SAXException` - any SAX exception, possibly wrapping another exception

See Also:

`org.xml.sax.helpers.DefaultHandler.startElement(java.lang.String, java.lang.String, java.lang.String, org.xml.sax.Attributes)`

Package edu.ou.weinmann.repsi.model.mapper

Provides the functionality to map the data from an object to the database.

See:

Description

Class Summary		*Page*
CalibrationMapper	Maps the data from the calibration run to the database.	10
DatabaseInstanceMapper	Maps the data from the database instance to the database.	12

TrialRunActionMapper	Maps the data from the trial run actions to the database.	*13*
TrialRunMapper	Maps the data from the trial run to the database.	*17*
TrialRunProtocolMapper	Maps the data from the trial run protocol to the database.	*19*

Package edu.ou.weinmann.repsi.model.mapper Description

Provides the functionality to map the data from an object to the database.

Class CalibrationMapper

edu.ou.weinmann.repsi.model.mapper

```
java.lang.Object
 └edu.ou.weinmann.repsi.model.mapper.CalibrationMapper
```

public class **CalibrationMapper**
extends Object

Maps the data from the calibration run to the database.

Author:
Walter Weinmann

Constructor Summary	*Page*
CalibrationMapper(String parSQLSyntaxCodeTarget, String parStartTime) Constructs a `CalibrationMapper` object.	*11*

Method Summary		*Page*
final boolean	**closeConnection**() Close the database connection.	*11*
final boolean	**initialiseCalibration**(Map parColumnsDatabaseInstance) Creates the row in the database table `TMD_CALIBRATION`.	*11*
final boolean	**initialiseCalibrationStatistic**(Map parColumnsCalibrationStatistic) Creates the row in the database table `TMD_CALIBRATION_STATISTIC`.	*11*
final boolean	**setComparison**(String parEquals, String parMessage) Updates in the database the columns `COMPARISON_EQUALS` and `COMPARISON_MESSAGE`.	*11*
final boolean	**setDescription**(String parDescription) Updates in the database the columns `DESCRIPTION`.	*12*
final boolean	**setPatternSqlIdiomName**(String parPatternSqlIdiomName) Updates in the database the columns `PATTERN_SQL_IDIOM_NAME`.	*12*
final boolean	**setSqlSyntaxCodeTqp**(String parSqlSyntaxCodeTtqp) Updates in the database the columns `SQL_SYNTAX_CODE_TTQP`.	*12*
final boolean	**setStatus**() Updates in the database the columns `END_TIME` and `STATUS_CODE`.	*12*

Constructor Detail

CalibrationMapper

```
public CalibrationMapper(String parSQLSyntaxCodeTarget,
                         String parStartTime)
```

Constructs a `CalibrationMapper` object.

Method Detail

closeConnection

```
public final boolean closeConnection()
```

Close the database connection.

Returns:

`true` if the operation succeeeded and `false` otherwise.

---

initialiseCalibration

```
public final boolean initialiseCalibration(Map parColumnsDatabaseInstance)
```

Creates the row in the database table `TMD_CALIBRATION`.

Parameters:

`parColumnsDatabaseInstance` - The `Map` object containing the database columns.

Returns:

`true` if the operation succeeeded and `false` otherwise.

---

initialiseCalibrationStatistic

```
public final boolean initialiseCalibrationStatistic(Map parColumnsCalibrationStatistic)
```

Creates the row in the database table `TMD_CALIBRATION_STATISTIC`.

Parameters:

`parColumnsCalibrationStatistic` - The `Map` object containing the database columns.

Returns:

`true` if the operation succeeeded and `false` otherwise.

---

setComparison

```
public final boolean setComparison(String parEquals,
                                   String parMessage)
```

Updates in the database the columns `COMPARISON_EQUALS` and `COMPARISON_MESSAGE`.

Parameters:

`parEquals` - Y if both `ResultSet`s are equal, or N otherwise.

`parMessage` - The new message reasoning the inequality of both `ResultSet`.

Returns:

`true` if the operation succeeeded and `false` otherwise.

---

setDescription

```
public final boolean setDescription(String parDescription)
```

Updates in the database the columns `DESCRIPTION`.
Parameters:
`parDescription` - The new description.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

setPatternSqlIdiomName

```
public final boolean setPatternSqlIdiomName(String parPatternSqlIdiomName)
```

Updates in the database the columns `PATTERN_SQL_IDIOM_NAME`.
Parameters:
`parPatternSqlIdiomName` - The new pattern SQL idiom name.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

setSqlSyntaxCodeTqp

```
public final boolean setSqlSyntaxCodeTqp(String parSqlSyntaxCodeTtqp)
```

Updates in the database the columns `SQL_SYNTAX_CODE_TTQP`.
Parameters:
`parSqlSyntaxCodeTtqp` - The new SQL syntax code.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

setStatus

```
public final boolean setStatus()
```

Updates in the database the columns `END_TIME` and `STATUS_CODE`.
Returns:
`true` if the operation succeeeded and `false` otherwise.

Class DatabaseInstanceMapper

edu.ou.weinmann.repsi.model.mapper

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.mapper.DatabaseInstanceMapper
```

---

public class **DatabaseInstanceMapper**
extends Object

Maps the data from the database instance to the database.

Author:
Walter Weinmann

Constructor Summary	Page
DatabaseInstanceMapper(String parSQLSyntaxCodeTarget) Constructs a `DatabaseInstanceMapper` object.	13

Method Summary		Page
final boolean	**closeConnection**() Close the database connection.	13
final Map	**getDatabaseInstance**(int parDatabaseInstanceId) Returns a `Map` containing the relevant columns of the requested database instance.	13

Constructor Detail

DatabaseInstanceMapper

```
public DatabaseInstanceMapper(String parSQLSyntaxCodeTarget)
```

Constructs a `DatabaseInstanceMapper` object.

Method Detail

closeConnection

```
public final boolean closeConnection()
```

Close the database connection.
Returns:
`true` if the operation succeeeded and `false` otherwise.

getDatabaseInstance

```
public final Map getDatabaseInstance(int parDatabaseInstanceId)
```

Returns a `Map` containing the relevant columns of the requested database instance.
Parameters:
`parDatabaseInstanceId` - The identification of the database instance.
Returns:
a `Map` containing the relevant columns of the database instance.

Class TrialRunActionMapper

edu.ou.weinmann.repsi.model.mapper

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.mapper.TrialRunActionMapper
```

public class **TrialRunActionMapper**
extends Object

Maps the data from the trial run actions to the database.

Author:
Walter Weinmann

Constructor Summary	Page
TrialRunActionMapper(TrialRunProtocolMapper parTrialRunProtocol, String parSQLSyntaxCodeTarget, int parDatabaseInstanceId, int parTestSuiteId, String parStartTime) Constructs a `TrialRunActionMapper` object.	14

Method Summary		Page
final boolean	**closeConnection**() Close the database connection.	15
final boolean	**initialise**(Map parColumnsTestSuiteAction) Creates the row in the database table `TMD_TRIAL_RUN_ACTION`.	15
final boolean	**resetAppliedPatternSelectStmnt**() Rest the database column `APPLIED_PATTERN_SELECT_STMNT`.	15
final boolean	**resetUnappliedPatternSelectStmnt**() Rest the database column `UNAPPLIED_PATTERN_SELECT_STMNT`.	15
final boolean	**setAppliedEndAction**(Date parStartTime, Date parEndTime, long parDuration) Updates in the database the columns `APPLIED_DURATION_MICRO_SECOND`, `APPLIED_END_TIME`, and `APPLIED_START_TIME`.	15
final boolean	**setAppliedEndActionError**(String parErrorMessage) Updates in the database the columns `APPLIED_ERROR_MESSAGE` and `APPLIED_STATUS`.	16
final boolean	**setAppliedStartAction**(String parStatement, String parOrderBy) Updates in the database the columns `APPLIED_PATTERN_SELECT_STMNT` and `APPLIED_STATUS`.	16
final boolean	**setComparison**(String parEquals, String parMessage) Updates in the database the columns `COMPARISON_EQUALS` and `COMPARISON_MESSAGE`.	16
final void	**setSequenceNumberAction**(long parSequenceNumberAction) Sets the current action sequence number.	16
final boolean	**setTableName**(String parTableName) Updates in the database the column `TABLE_NAME`.	16
final boolean	**setUnappliedEndAction**(Date parStartTime, Date parEndTime, long parDuration) Updates in the database the columns `UNAPPLIED_DURATION_MICRO_SECOND`, `UNAPPLIED_END_TIME`, and `UNAPPLIED_START_TIME`.	17
final boolean	**setUnappliedEndActionError**(String parErrorMessage) Updates in the database the columns `UNAPPLIED_ERROR_MESSAGE` and `UNAPPLIED_STATUS`.	17
final boolean	**setUnappliedStartAction**(String parStatement, String parOrderBy) Updates in the database the columns `UNAPPLIED_PATTERN_SELECT_STMNT` and `UNAPPLIED_STATUS`.	17

Constructor Detail

TrialRunActionMapper

```
public TrialRunActionMapper(TrialRunProtocolMapper parTrialRunProtocol,
                            String parSQLSyntaxCodeTarget,
                            int parDatabaseInstanceId,
                            int parTestSuiteId,
                            String parStartTime)
```

Constructs a `TrialRunActionMapper` object.

Method Detail

closeConnection

```
public final boolean closeConnection()
```

Close the database connection.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

initialise

```
public final boolean initialise(Map parColumnsTestSuiteAction)
```

Creates the row in the database table `TMD_TRIAL_RUN_ACTION`.
Parameters:
`parColumnsTestSuiteAction` - The `Map` object containing the database columns.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

resetAppliedPatternSelectStmnt

```
public final boolean resetAppliedPatternSelectStmnt()
```

Rest the database column `APPLIED_PATTERN_SELECT_STMNT`.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

resetUnappliedPatternSelectStmnt

```
public final boolean resetUnappliedPatternSelectStmnt()
```

Rest the database column `UNAPPLIED_PATTERN_SELECT_STMNT`.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

setAppliedEndAction

```
public final boolean setAppliedEndAction(Date parStartTime,
                                         Date parEndTime,
                                         long parDuration)
```

Updates in the database the columns `APPLIED_DURATION_MICRO_SECOND`, `APPLIED_END_TIME`, and `APPLIED_START_TIME`.
Parameters:
`parStartTime` - The new start date and time.
`parEndTime` - The new end date and time.
`parDuration` - The new duration of the query execution.
Returns:
`true` if the operation succeeeded and `false` otherwise.

setAppliedEndActionError

```
public final boolean setAppliedEndActionError(String parErrorMessage)
```

Updates in the database the columns `APPLIED_ERROR_MESSAGE` and `APPLIED_STATUS`.

Parameters:

`parErrorMessage` - The new applied error message.

Returns:

`true` if the operation succeeeded and `false` otherwise.

setAppliedStartAction

```
public final boolean setAppliedStartAction(String parStatement,
                                           String parOrderBy)
```

Updates in the database the columns `APPLIED_PATTERN_SELECT_STMNT` and `APPLIED_STATUS`.

Parameters:

`parStatement` - The new applied `SQL` statement.

`parOrderBy` - Rhe new `ORDER BY` clause.

Returns:

`true` if the operation succeeeded and `false` otherwise.

setComparison

```
public final boolean setComparison(String parEquals,
                                   String parMessage)
```

Updates in the database the columns `COMPARISON_EQUALS` and `COMPARISON_MESSAGE`.

Parameters:

`parEquals` - Y if both `ResultSet`s are equal, or N otherwise.

`parMessage` - The new message reasoning the inequality of both `ResultSet`.

Returns:

`true` if the operation succeeeded and `false` otherwise.

setSequenceNumberAction

```
public final void setSequenceNumberAction(long parSequenceNumberAction)
```

Sets the current action sequence number.

Parameters:

`parSequenceNumberAction` - The current action sequence number.

setTableName

```
public final boolean setTableName(String parTableName)
```

Updates in the database the column `TABLE_NAME`.

Parameters:

`parTableName` - The new table name.

Returns:

`true` if the operation succeeeded and `false` otherwise.

setUnappliedEndAction

```
public final boolean setUnappliedEndAction(Date parStartTime,
                                           Date parEndTime,
                                           long parDuration)
```

Updates in the database the columns `UNAPPLIED_DURATION_MICRO_SECOND`, `UNAPPLIED_END_TIME`, and `UNAPPLIED_START_TIME`.

Parameters:

`parStartTime` - The new start date and time.
`parEndTime` - The new end date and time.
`parDuration` - The new duration of the query execution.

Returns:

`true` if the operation succeeeded and `false` otherwise.

setUnappliedEndActionError

```
public final boolean setUnappliedEndActionError(String parErrorMessage)
```

Updates in the database the columns `UNAPPLIED_ERROR_MESSAGE` and `UNAPPLIED_STATUS`.

Parameters:

`parErrorMessage` - The new unapplied error message.

Returns:

`true` if the operation succeeeded and `false` otherwise.

setUnappliedStartAction

```
public final boolean setUnappliedStartAction(String parStatement,
                                             String parOrderBy)
```

Updates in the database the columns `UNAPPLIED_PATTERN_SELECT_STMNT` and `UNAPPLIED_STATUS`.

Parameters:

`parStatement` - The new unapplied `SQL` statement.
`parOrderBy` - Rhe new `ORDER BY` clause.

Returns:

`true` if the operation succeeeded and `false` otherwise.

Class TrialRunMapper

edu.ou.weinmann.repsi.model.mapper

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.mapper.TrialRunMapper
```

public class **TrialRunMapper**
extends Object

Maps the data from the trial run to the database.

Author:

Walter Weinmann

Constructor Summary	*Page*
TrialRunMapper(TrialRunProtocolMapper parTrialRunProtocol, String parSQLSyntaxCodeTarget, int parDatabaseInstanceId, int parTestSuiteId, String parStartTime) Constructs a `TrialRunMapper` object.	*18*

Method Summary		*Page*
final boolean	**closeConnection**() Close the database connection.	*18*
final boolean	**initialise**(Map parColumnsDatabaseInstance, Map parColumnsTestSuite) Creates the row in the database table `TMD_TRIAL_RUN`.	*18*
final boolean	**setDescription**(String parDescription) Updates in the database the columns `DESCRIPTION`.	*19*
final boolean	**setStatus**(String parStatus) Updates in the database the columns `END_TIME` and `STATUS_CODE`.	*19*

Constructor Detail

TrialRunMapper

```
public TrialRunMapper(TrialRunProtocolMapper parTrialRunProtocol,
                      String parSQLSyntaxCodeTarget,
                      int parDatabaseInstanceId,
                      int parTestSuiteId,
                      String parStartTime)
```

Constructs a `TrialRunMapper` object.

Method Detail

closeConnection

```
public final boolean closeConnection()
```

Close the database connection.

Returns:

`true` if the operation succeeeded and `false` otherwise.

initialise

```
public final boolean initialise(Map parColumnsDatabaseInstance,
                                Map parColumnsTestSuite)
```

Creates the row in the database table `TMD_TRIAL_RUN`.

Parameters:

`parColumnsDatabaseInstance` - The `Map` object containing the columns of the database table `TMD_DATABASE_INSTANCE`.

`parColumnsTestSuite` - The `Map` object containing the columns of the database table `TMD_TEST_SUITE`.

Returns:

`true` if the operation succeeeded and `false` otherwise.

setDescription

`public final boolean setDescription(String parDescription)`

Updates in the database the columns `DESCRIPTION`.
Parameters:
`parDescription` - The new description.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

setStatus

`public final boolean setStatus(String parStatus)`

Updates in the database the columns `END_TIME` and `STATUS_CODE`.
Parameters:
`parStatus` - The new status.
Returns:
`true` if the operation succeeeded and `false` otherwise.

Class TrialRunProtocolMapper

edu.ou.weinmann.repsi.model.mapper

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.mapper.TrialRunProtocolMapper
```

---

public class **TrialRunProtocolMapper**
extends Object

Maps the data from the trial run protocol to the database.

Author:
Walter Weinmann

Constructor Summary	Page
TrialRunProtocolMapper(String parSQLSyntaxCodeTarget, int parDatabaseInstanceId, int parTestSuiteId, String parStartTime) Constructs a `TrialRunProtocolMapper` object.	20

Method Summary		Page
final boolean	**closeConnection**() Close the database connection.	20
final boolean	**createErrorProtocol**(String parMessage, boolean parAborted) Create an error protocol entry.	20
final boolean	**createProtocol**(String parMessage) Creates an protocol entry.	20
final boolean	**createProtocol**(String parMessagePart1, String parMessagePart2) Creates an protocol entry.	21
final boolean	**createProtocol**(String parMessagePart1, long parMessagePart2) Creates an protocol entry.	21

final int	`getNumberOfAborts()` Returns the current number of aborts reported.	*21*
final int	`getNumberOfErrors()` Returns the current number of errors reported.	*21*
final boolean	`isAborted()` Returns whether the processing was required to be terminated immedeately.	*21*
final void	`setSequenceNumberAction(long parSequenceNumberAction)` Sets the current action sequence number.	*21*

Constructor Detail

TrialRunProtocolMapper

```
public TrialRunProtocolMapper(String parSQLSyntaxCodeTarget,
                              int parDatabaseInstanceId,
                              int parTestSuiteId,
                              String parStartTime)
```

Constructs a `TrialRunProtocolMapper` object.

Method Detail

closeConnection

```
public final boolean closeConnection()
```

Close the database connection.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

createErrorProtocol

```
public final boolean createErrorProtocol(String parMessage,
                                         boolean parAborted)
```

Create an error protocol entry.
Parameters:
`parMessage` - The error message to be contained in the protocol.
`parAborted` - Whether the processing should be termintated immedeately.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

createProtocol

```
public final boolean createProtocol(String parMessage)
```

Creates an protocol entry.
Parameters:
`parMessage` - The error message to be contained in the protocol.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

createProtocol

```
public final boolean createProtocol(String parMessagePart1,
                                    long parMessagePart2)
```

Creates an protocol entry.
Parameters:
`parMessagePart1` - The left part of the information to be contained in the protocol.
`parMessagePart2` - The right part of the information to be contained in the protocol.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

createProtocol

```
public final boolean createProtocol(String parMessagePart1,
                                    String parMessagePart2)
```

Creates an protocol entry.
Parameters:
`parMessagePart1` - The left part of the information to be contained in the protocol.
`parMessagePart2` - The right part of the information to be contained in the protocol.
Returns:
`true` if the operation succeeeded and `false` otherwise.

---

getNumberOfAborts

```
public final int getNumberOfAborts()
```

Returns the current number of aborts reported.
Returns:
the current number of aborts reported.

---

getNumberOfErrors

```
public final int getNumberOfErrors()
```

Returns the current number of errors reported.
Returns:
the current number of errors reported.

---

isAborted

```
public final boolean isAborted()
```

Returns whether the processing was required to be terminated immedeately.
Returns:
`true` if the processing was required to be terminated immedeately.

---

setSequenceNumberAction

```
public final void setSequenceNumberAction(long parSequenceNumberAction)
```

Sets the current action sequence number.
Parameters:
`parSequenceNumberAction` - The current action sequence number.

Package edu.ou.weinmann.repsi.model.trial

Provides the trial run functionality.

See:
Description

Class Summary		Page
Trial	Manages the trial run functionality of the REPSI tool: • executes a trail run • extracts the results of trial runs from the database into an Excel file.	22

Package edu.ou.weinmann.repsi.model.trial Description

Provides the trial run functionality.

Class Trial

edu.ou.weinmann.repsi.model.trial

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.trial.Trial
```

public class **Trial**
extends Object

Manages the trial run functionality of the REPSI tool:

- executes a trail run
- extracts the results of trial runs from the database into an Excel file.

Author:
Walter Weinmann Manages the trial runs.
Walter Weinmann

Constructor Summary	Page
`Trial()` Constructs a `Trial` object.	23
`Trial(String parProprtiesFilename, boolean parPropertiesXml)` Constructs a `Trial` object.	23

Method Summary		Page
`final boolean`	`runTrial(int parDatabaseInstanceId, int parTestSuiteId, String parDescription, int parFetchSize, int parFrequency, long parPrecision)` Runs a trial and records the measured results.	23

`final boolean`	`trialDataToExcel(String parFileName, boolean parAllData)` Exports all the data of a trial run from the master database into an Excel file.	23

Constructor Detail

Trial

```
public Trial()
```

Constructs a `Trial` object. The name of the used properties file is taken from mthe class `edu.ou.weinmann.repsi.model.util.Global` and does not constitute an XML document.

Trial

```
public Trial(String parProprtiesFilename,
             boolean parPropertiesXml)
```

Constructs a `Trial` object.

Method Detail

runTrial

```
public final boolean runTrial(int parDatabaseInstanceId,
                              int parTestSuiteId,
                              String parDescription,
                              int parFetchSize,
                              int parFrequency,
                              long parPrecision)
```

Runs a trial and records the measured results.
Parameters:
`parDatabaseInstanceId` - The identification of the used test database instance.
`parTestSuiteId` - The identification of the used test test suite.
`parDescription` - The description of the calibration run.
`parFetchSize` - The fetch size.
`parFrequency` - The number of cycles to be executed.
`parPrecision` - The exponent (base 10) of the required precision for the response time. Calibrates the execution of a simple Java method.
Returns:
`true` if the processing ended without any error, and `false` otherwise.

---

trialDataToExcel

```
public final boolean trialDataToExcel(String parFileName,
                                      boolean parAllData)
```

Exports all the data of a trial run from the master database into an Excel file.
Parameters:
`parFileName` - The complete file name of the Excel file including the directory.
`parAllData` - Whether all data are required instead of the latest trial run only.
Returns:
`true` if the calibration data were exported without any error, and `false` otherwise.

Package edu.ou.weinmann.repsi.model.trial.metadata

Provides classes to manage the database metadata.

See:
  Description

Class Summary		*Page*
Column	Manages one column of a database table.	*24*
Columns	Manages the meta data a database table: primary key, imported foreign keys and database columns.	*26*
ForeignKey	Manages one foreign key of a database table.	*29*
ForeignKeys	Manages the foreign keys of a database table.	*29*
PrimaryKey	Manages the primary key of a database table.	*30*

Package edu.ou.weinmann.repsi.model.trial.metadata Description

Provides classes to manage the database metadata.

Class Column

edu.ou.weinmann.repsi.model.trial.metadata

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.trial.metadata.Column
```

final public class **Column**
extends Object

Manages one column of a database table.

Author:
  Walter Weinmann

Method Summary		*Page*
`String`	`getColumnName()` Returns the column name.	*25*
`int`	`getColumnSize()` Returns the column size.	*25*
`int`	`getDataType()` Returns the SQL type of `java.sql.Types`.	*25*
`int`	`getDecimalDigits()` Returns the number of fractional digits.	*25*
`BigDecimal`	`getDecimalFactor()` Returns the decimal factor.	*25*
`String`	`getIsNullable()` Returns whether the column can include `null`.	*26*

int	getKeySeq() Returns the sequence number within primary key.	26
boolean	isForeignKeyColumn() Returns whether the column is a foreign key column.	26

Method Detail

getColumnName

```
public String getColumnName()
```

Returns the column name.
Returns:
the column name.

getColumnSize

```
public int getColumnSize()
```

Returns the column size.
Returns:
the column size.

getDataType

```
public int getDataType()
```

Returns the SQL type of `java.sql.Types`.
Returns:
the SQL type of `java.sql.Types`.

getDecimalDigits

```
public int getDecimalDigits()
```

Returns the number of fractional digits.
Returns:
the number of fractional digits.

getDecimalFactor

```
public BigDecimal getDecimalFactor()
```

Returns the decimal factor.
Returns:
the decimal factor.

getIsNullable

```
public String getIsNullable()
```

Returns whether the column can include `null`.

Returns:

"YES", if the column can include `null`, "NO", if the column cannot include `null`, and an empty string, if the nullability for the column is unknown.

getKeySeq

```
public int getKeySeq()
```

Returns the sequence number within primary key.

Returns:

sequence number within primary key.

isForeignKeyColumn

```
public boolean isForeignKeyColumn()
```

Returns whether the column is a foreign key column.

Returns:

`true` if the column is a foreign key column.

Class Columns

edu.ou.weinmann.repsi.model.trial.metadata

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.trial.metadata.Columns
```

final public class **Columns**
extends Object

Manages the meta data a database table: primary key, imported foreign keys and database columns.

Author:

Walter Weinmann

Constructor Summary	Page
Columns(TrialRunProtocolMapper parTrialRunProtocol, DatabaseAccessor parDBAccess, String parCatalog, String parSchema, String parTableName) Constructs a `Columns` object.	27

Method Summary		Page
boolean	**determineRandomForeignKeys**() Determines a random set of foreign key values.	27
boolean	**determineRandomPrimaryKey**(DataGenerator parDataGenerator) Determines a random primary key.	27

Column	**getColumn**(String parColumnName) Returns the required `Column` object.	*28*
String	**getColumnName**(int parOrdinalPosition) Returns the database column name related to a given ordinal position.	*28*
ForeignKeys	**getForeignKeys**() Returns the `ForeignKeys` object of this database table.	*28*
PrimaryKey	**getPrimaryKey**() Returns the `PrimaryKey` object of this database table.	*28*
Object	**getRandomFkValue**(String parFkColumnName) Returns a random foreign key.	*28*
Object	**getRandomPkValue**(String parPkColumnName) Returns a random primary key.	*28*
void	**protocol**(TrialRunProtocolMapper parTrialRunProtocol) Creates a protocol entry.	*29*
int	**sizeColumns**() Returns the number of columns in this database table.	*29*

Constructor Detail

Columns

```
public Columns(TrialRunProtocolMapper parTrialRunProtocol,
               DatabaseAccessor parDBAccess,
               String parCatalog,
               String parSchema,
               String parTableName)
```

Constructs a `Columns` object.

Method Detail

determineRandomForeignKeys

```
public boolean determineRandomForeignKeys()
```

Determines a random set of foreign key values.
Returns:
`true` if the processing finished without errors.

---

determineRandomPrimaryKey

```
public boolean determineRandomPrimaryKey(DataGenerator parDataGenerator)
```

Determines a random primary key.
Parameters:
`parDataGenerator` - The `DataGenerator` object.
Returns:
`true` if the processing finished without errors.

---

getColumn

`public Column getColumn(String parColumnName)`

Returns the required `Column` object.
Parameters:
`parColumnName` - The name of the required dstabase column.
Returns:
the `Column` object or `null` if the required database column was not found.

---

getColumnName

`public String getColumnName(int parOrdinalPosition)`

Returns the database column name related to a given ordinal position.
Parameters:
`parOrdinalPosition` - The ordinal position of the column in the database table (starting with 1).
Returns:
the database column name or `null` if the required ordinal position is out of range.

---

getForeignKeys

`public ForeignKeys getForeignKeys()`

Returns the `ForeignKeys` object of this database table.
Returns:
a `ForeignKeys` object containing the foreign keys.

---

getPrimaryKey

`public PrimaryKey getPrimaryKey()`

Returns the `PrimaryKey` object of this database table.
Returns:
a `PrimaryKey` object containing the primary key.

---

getRandomFkValue

`public Object getRandomFkValue(String parFkColumnName)`

Returns a random foreign key.
Parameters:
`parFkColumnName` - The name of the foreign key column.
Returns:
a valid value for the given foreign key column.

---

getRandomPkValue

`public Object getRandomPkValue(String parPkColumnName)`

Returns a random primary key.

Parameters:
`parPkColumnName` - The name of the primary key column.
Returns:
a valid value for the given primary key column.

---

protocol

`public void **protocol**(TrialRunProtocolMapper parTrialRunProtocol)`

Creates a protocol entry.
Parameters:
`parTrialRunProtocol` - The `TrialRunProtocolMapper` object.

---

sizeColumns

`public int **sizeColumns**()`

Returns the number of columns in this database table.
Returns:
the number of columns in this database table

Class ForeignKey

edu.ou.weinmann.repsi.model.trial.metadata

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.trial.metadata.ForeignKey
```

---

final public class **ForeignKey**
extends Object

Manages one foreign key of a database table.

Author:
Walter Weinmann

Class ForeignKeys

edu.ou.weinmann.repsi.model.trial.metadata

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.trial.metadata.ForeignKeys
```

---

final public class **ForeignKeys**
extends Object

Manages the foreign keys of a database table.

Author:
Walter Weinmann

Class PrimaryKey

edu.ou.weinmann.repsi.model.trial.metadata

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.trial.metadata.PrimaryKey
```

final public class **PrimaryKey**
extends Object

Manages the primary key of a database table.

Author:
Walter Weinmann

Method Summary		Page
Object	`getRandomPkValue(String parPkColumnName)` Returns a random primary key column.	30

Method Detail

getRandomPkValue

```
public Object getRandomPkValue(String parPkColumnName)
```

Returns a random primary key column.
Parameters:
`parPkColumnName` - The name of the primary key column.
Returns:
a valid value for the given primary key column.

Package edu.ou.weinmann.repsi.model.trial.util

Provides the utility classes for the trial run component of the REPSI tool.

See:
Description

Class Summary		Page
DataGenerator	Generates the columns and rows of a given test database instance.	31
ResultSetComparator	Compares two `ResultSet` objects for equality.	33

Package edu.ou.weinmann.repsi.model.trial.util Description

Provides the utility classes for the trial run component of the REPSI tool.

Class DataGenerator

edu.ou.weinmann.repsi.model.trial.util

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.trial.util.DataGenerator
```

public class **DataGenerator**
extends Object

Generates the columns and rows of a given test database instance.

Author:
Walter Weinmann

Constructor Summary	Page
DataGenerator(TrialRunProtocolMapper parTrialRunProtocol, DatabaseAccessor parDBAccess, String parTableName, Columns parColumns) Constructs a `DataGenerator` object.	31

Method Summary		Page
final Date	**generateColumnValueDateKey**(Column parColumn) Generates a value for a given primary key column of type `DATE`.	32
static final Date	**generateColumnValueDateKeyLow**() Generates the lowest possible `DATE` value for a primary key.	32
final BigDecimal	**generateColumnValueNumericKey**(Column parColumn) Generates a value for a given primary key column of type `BigDecimal`.	32
static final BigDecimal	**generateColumnValueNumericKeyLow**() Generates the lowest possible `BigDecimal` value for a primary key.	32
final String	**generateColumnValueVarcharKey**(Column parColumn) Generates a value for a given primary key column of type `String`.	32
static final String	**generateColumnValueVarcharKeyLow**() Generates the lowest possible `String` value for a primary key.	33
static final String	**generateColumnValueVarcharKeyLow**(Column parColumn) Generates the lowest possible `String` value for a given primary key column.	33
final boolean	**generateRow**(long parExecutionFrequency) Generates set of rows in a database table.	33
final long	**getNumberRowsGenerated**() Returns the effective number of generated rows.	33

Constructor Detail

DataGenerator

```
public DataGenerator(TrialRunProtocolMapper parTrialRunProtocol,
                     DatabaseAccessor parDBAccess,
                     String parTableName,
                     Columns parColumns)
```

Constructs a `DataGenerator` object.

Method Detail

generateColumnValueDateKey

`public final Date generateColumnValueDateKey(Column parColumn)`

Generates a value for a given primary key column of type `DATE`.
Parameters:
`parColumn` - The name of the database column.
Returns:
a valid value of a `DATE` column, or `null` if any error occured.

---

generateColumnValueDateKeyLow

`public static final Date generateColumnValueDateKeyLow()`

Generates the lowest possible `DATE` value for a primary key.
Returns:
a `DATE` object.

---

generateColumnValueNumericKey

`public final BigDecimal generateColumnValueNumericKey(Column parColumn)`

Generates a value for a given primary key column of type `BigDecimal`.
Parameters:
`parColumn` - The name of the database column.
Returns:
a valid value of a `BigDecimal` column, or `null` if any error occured.

---

generateColumnValueNumericKeyLow

`public static final BigDecimal generateColumnValueNumericKeyLow()`

Generates the lowest possible `BigDecimal` value for a primary key.
Returns:
a `BigDecimal` object.

---

generateColumnValueVarcharKey

`public final String generateColumnValueVarcharKey(Column parColumn)`

Generates a value for a given primary key column of type `String`.
Parameters:
`parColumn` - The name of the database column.
Returns:
a valid value of a `String` column, or `null` if any error occured.

---

generateColumnValueVarcharKeyLow

`public static final String` **`generateColumnValueVarcharKeyLow`**`()`

Generates the lowest possible `String` value for a primary key.

Returns:

a `String` object.

---

generateColumnValueVarcharKeyLow

`public static final String` **`generateColumnValueVarcharKeyLow`**`(Column parColumn)`

Generates the lowest possible `String` value for a given primary key column.

Parameters:

`parColumn` - The name of the database column.

Returns:

a `String` object.

---

generateRow

`public final boolean` **`generateRow`**`(long parExecutionFrequency)`

Generates set of rows in a database table.

Parameters:

`parExecutionFrequency` - The number of rows to be generated.

Returns:

`true` if the required number of rows was generated successfully, and `false` otherwise.

---

getNumberRowsGenerated

`public final long` **`getNumberRowsGenerated`**`()`

Returns the effective number of generated rows.

Returns:

the effective number of generated rows.

Class ResultSetComparator

edu.ou.weinmann.repsi.model.trial.util

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.trial.util.ResultSetComparator
```

---

public class **ResultSetComparator**
extends Object

Compares two `ResultSet` objects for equality.

Author:

Walter Weinmann

Constructor Summary	Page
ResultSetComparator() Constructs a `ResultSetComparator` object.	34

Method Summary		Page
final boolean	**compare**(DatabaseAccessor[] parDBAccess) Compares two `ResultSet` objects.	34
final String	**getLastErrorMsg**() Returns the last error message.	34
final void	**setDescription**(String parDescription, int parPos) Sets the description at the required position.	34
final void	**setOrderBy**(String parOrderBy, int parPos) Sets the order by clause at the required position.	35
final void	**setSelectStmnt**(String parSelectStmnt, int parPos) Sets the `SELECT` statement at the required position.	35
final void	**setSqlSyntaxCode**(String parSqlSyntaxCode) Sets the `SQL` syntax code.	35

Constructor Detail

ResultSetComparator

```
public ResultSetComparator()
```

Constructs a `ResultSetComparator` object.

Method Detail

compare

```
public final boolean compare(DatabaseAccessor[] parDBAccess)
```

Compares two `ResultSet` objects.
Parameters:
`parDBAccess` - The `DatabaseAccessor` object.
Returns:
`true` if both `ResultSet`s are equal, or `false` otherwise.

---

getLastErrorMsg

```
public final String getLastErrorMsg()
```

Returns the last error message.
Returns:
the last error message.

---

setDescription

```
public final void setDescription(String parDescription,
                                 int parPos)
```

Sets the description at the required position.
Parameters:
`parDescription` - The new description.
`parPos` - The required position.

---

setOrderBy

```
public final void setOrderBy(String parOrderBy,
                             int parPos)
```

Sets the order by clause at the required position.
Parameters:
`parOrderBy` - The new order by clause.
`parPos` - The required position.

---

setSelectStmnt

```
public final void setSelectStmnt(String parSelectStmnt,
                                 int parPos)
```

Sets the `SELECT` statement at the required position.
Parameters:
`parSelectStmnt` - The new `SELECT` statement.
`parPos` - The required position.

---

setSqlSyntaxCode

```
public final void setSqlSyntaxCode(String parSqlSyntaxCode)
```

Sets the `SQL` syntax code.
Parameters:
`parSqlSyntaxCode` - The new `SQL` syntax code.

Package edu.ou.weinmann.repsi.model.util

Provides the utility classes for the model component of the REPSI tool.

See:
Description

Interface Summary		Page
Global	Provides global constants.	*48*

Class Summary		Page
Configurator	Manages the configuration parameters.	*36*
DatabaseAccessor	Accesses the database and manages the details related to `Connection`, `Statement` and `ResultSet`.	*37*
DatabaseToExcel	Create an Excel compatible file out of database tables.	*47*
SQLRewriter	Adapts the syntactical variations of different SQL versions by rewriting the SQL statements.	*65*

Package edu.ou.weinmann.repsi.model.util Description

Provides the utility classes for the model component of the REPSI tool.

Class Configurator

edu.ou.weinmann.repsi.model.util

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.util.Configurator
```

final public class **Configurator**
extends Object

Manages the configuration parameters.

Author:
Walter Weinmann

Method Summary		*Page*
static Configurator	**getInstance**() Returns the current instance of the `Configurator` class.	*36*
static Configurator	**getInstance**(String parFileName, boolean parXml) Returns the current instance of the `Configurator` class.	*36*
String	**getProperty**(String parKey) Returns the value of a given property key.	*37*
static boolean	**removeInstance**() Removes the current `Configurator` object.	*37*
boolean	**setProperty**(String parKey, String parValue) sets the value of a given property key.	*37*

Method Detail

getInstance

```
public static Configurator getInstance()
```

Returns the current instance of the `Configurator` class.
Returns:
the current instance of the `Configurator` class, if the instantiation was successful, or `null` otherwise.

getInstance

```
public static Configurator getInstance(String parFileName,
                                       boolean parXml)
```

Returns the current instance of the `Configurator` class.
Parameters:
`parFileName` - The complete file name of the proprties file including the directory.

`parXml` - Whether the properties file constitutes an XML document and `false` if the properties file consists of a flat file.

Returns:

an instance of the `Configurator` class, if the instantiation was successful, or `null` otherwise.

---

getProperty

```
public String getProperty(String parKey)
```

Returns the value of a given property key.

Parameters:

`parKey` - The name of the property key.

Returns:

the value of the property key.

---

removeInstance

```
public static boolean removeInstance()
```

Removes the current `Configurator` object.

Returns:

`true`, if an instance was existing, or `false` otherwise.

---

setProperty

```
public boolean setProperty(String parKey,
                           String parValue)
```

sets the value of a given property key.

Parameters:

`parKey` - The name of the property key.

`parValue` - The new value of this property key.

Returns:

`true` if the new value was successfully stored.

Class DatabaseAccessor

edu.ou.weinmann.repsi.model.util

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.util.DatabaseAccessor
```

All Implemented Interfaces:

Global

---

public class **DatabaseAccessor**
extends Object
implements Global

Accesses the database and manages the details related to `Connection`, `Statement` and `ResultSet`.

Author:

Walter Weinmann

Fields inherited from interface edu.ou.weinmann.repsi.model.util.Global
COLUMN_NAME_APPLIED_PATTERN_ORDER_BY, COLUMN_NAME_APPLIED_PATTERN_SELECT_STMNT, COLUMN_NAME_ARITHMETIC_MEAN, COLUMN_NAME_GEOMETRIC_MEAN, COLUMN_NAME_JDBC_DRIVER, COLUMN_NAME_JDBC_URL, COLUMN_NAME_KURTOSIS, COLUMN_NAME_MAXIMUM_VALUE, COLUMN_NAME_MINIMUM_VALUE, COLUMN_NAME_NAME, COLUMN_NAME_NUMBER_OF_VALUES, COLUMN_NAME_OBJECT, COLUMN_NAME_ORDER_BY, COLUMN_NAME_PASSWORD, COLUMN_NAME_PATTERN_SQL_IDIOM_NAME, COLUMN_NAME_PERCENTILE_25, COLUMN_NAME_PERCENTILE_50, COLUMN_NAME_PERCENTILE_75, COLUMN_NAME_READINGS, COLUMN_NAME_SKEWNESS, COLUMN_NAME_SQL_STATEMENT, COLUMN_NAME_SQL_SYNTAX_CODE, COLUMN_NAME_STANDARD_DEVIATION, COLUMN_NAME_TABLE_NAME, COLUMN_NAME_UNAPPLIED_PATTERN_ORDER_BY, COLUMN_NAME_UNAPPLIED_PATTERN_SELECT_STMNT, COLUMN_NAME_USER_NAME, COLUMN_NAME_VARIANCE, DATABASE_SCHEMA_IDENTIFIER_MASTER, DATABASE_SCHEMA_IDENTIFIER_TEST, DATE_FORMAT_DD_MM_YYYY_HH_MM_SS_SSS_JAVA, DATE_FORMAT_DD_MM_YYYY_HH_MM_SS_SSS_SQL, DATE_FORMAT_DD_MM_YYYY_JAVA, DATE_FORMAT_DD_MM_YYYY_SQL, DATE_FORMAT_YYYY_MM_DD_HH_MM_SS_SSS_SQL, DAY_FACTOR, ERROR_NOT_YET_IMPLEMENTED, FILE_TYPE_EXCEL, FILE_TYPE_XML, INITIAL_CAPACITY_FOREIGN_KEYS, INITIAL_CAPACITY_PRIMARY_KEYS, IS_NULLABLE_YES, MAX_STATISTICAL_OUTLINE, MAX_TRANSACTION_SIZE, META_DATA_COLUMN_NAME, META_DATA_COLUMN_SIZE, META_DATA_DATA_TYPE, META_DATA_DECIMAL_DIGITS, META_DATA_DECIMAL_FACTOR, META_DATA_FOREIGN_KEY_COLUMN_NAME, META_DATA_FOREIGN_KEY_NAME, META_DATA_FOREIGN_KEY_TABLE_NAME, META_DATA_IS_NULLABLE, META_DATA_KEY_SEQUENCE, META_DATA_ORDINAL_POSITION, META_DATA_PRIMARY_KEY_COLUMN_NAME, META_DATA_PRIMARY_KEY_NAME, META_DATA_PRIMARY_KEY_TABLE_NAME, MINIMAL_VALUE_VARCHAR, NULL, NUMBER_FORMAT_JAVA, NUMBER_FORMAT_LONG_JAVA, OPERATION_CODE_CREATE_TABLE, OPERATION_CODE_DROP_AND_CREATE_TABLE, OPERATION_CODE_DROP_TABLE, OPERATION_CODE_EXECUTE_QUERY, OPERATION_CODE_EXECUTE_QUERY_APPLIED, OPERATION_CODE_EXECUTE_QUERY_UNAPPLIED, OPERATION_CODE_INSERT_ROW, OPERATION_TYPE_INSTANCE, OPERATION_TYPE_QUERY, OPERATION_TYPE_SCHEMA, PROPERTIES_FILE_NAME, PROPERTY_PATH_1_DATABASE, PROPERTY_PATH_3_DRIVER, PROPERTY_PATH_3_PASSWORD, PROPERTY_PATH_3_SQL_SYNTAX_CODE, PROPERTY_PATH_3_URL, PROPERTY_PATH_3_USER_NAME, SEPARATOR_COMMA_SPACE_SINGLE_QUOTE, SEPARATOR_SINGLE_QUOTE_COMMA_SPACE_SINGLE_QUOTE, SQL_COLUMN_TYPE_CHAR, SQL_COLUMN_TYPE_VARCHAR2, SQL_SYNTAX_CODE_ORACLE_10G, SQL_SYNTAX_CODE_SQL_99, TRIAL_RUN_STATUS_END_ACTION, TRIAL_RUN_STATUS_END_FINALISATION, TRIAL_RUN_STATUS_END_INITIALISATION, TRIAL_RUN_STATUS_END_PROGRAM, TRIAL_RUN_STATUS_START_ACTION, TRIAL_RUN_STATUS_START_FINALISATION, TRIAL_RUN_STATUS_START_INITIALISATION, TRIAL_RUN_STATUS_START_PROGRAM

Constructor Summary	Page
DatabaseAccessor(String parDatabaseIdent, String parSQLSyntaxCodeTarget, boolean parIsMapper) Constructs a `DatabaseAccessor` object.	40
DatabaseAccessor(String parDatabaseIdent, String parSQLSyntaxCodeTarget, String parDriver, boolean parIsMapper) Constructs a `DatabaseAccessor` object.	40

Method Summary		Page
final ResultSet	**checkColumnName**(String parTableName, String parColumnName) Returns the metadata of a given database column.	40
final ResultSet	**checkTableName**(String parTableName) Returns the metadata of a given database table.	40
final boolean	**closeConnection**() Closes the currently open connection.	41
final boolean	**closeResultSet**(ResultSet parResultSet) Closes the given result set.	41
final boolean	**commit**() Makes the changes permanent which were created by all statements associated with this `Connection` object since the last `commit` or `rollback` was issued.	41
final boolean	**createStatement**() Creates a `Statement` object associtated with this `Connection` object.	41

final boolean	**executeQuery**(String parStmnt) Executes the `Statement` object by passing the specified SQL statement to the database.	41
final boolean	**executeQuery**(String parStmnt, String parSQLSyntaxCodeSource) Executes the `Statement` object by passing the specified SQL statement to the database.	42
final boolean	**executeQueryTrialRun**(String parStmnt, String parSQLSyntaxCodeSource) Executes the `Statement` object by passing the specified SQL statement to the database.	42
final boolean	**executeUpdate**(String parStmnt) Executes the `Statement` object by passing the specified SQL statement to the database.	42
final boolean	**executeUpdate**(String parStmnt, boolean parIgnoreDuplicate) Executes the `Statement` object by passing the specified SQL statement to the database.	42
final boolean	**executeUpdate**(String parStmnt, String parSQLSyntaxCodeSource) Executes the `Statement` object by passing the specified SQL statement to the database.	42
final boolean	**executeUpdate**(String parStmnt, String parSQLSyntaxCodeSource, boolean parIgnoreDuplicate) Executes the `Statement` object by passing the specified SQL statement to the database.	43
final boolean	**executeUpdateDirect**(String parStmnt, boolean parIgnoreDuplicate) Executes the `Statement` object by passing the specified SQL statement to the database.	43
final BigDecimal[]	**getArrayBigDecimal**(int parPos) Retrieves the value of the designated column in the current row of this `ResultSet` object as an `Array` object of `BigDecimal`.	43
static final BigDecimal[]	**getArrayBigDecimal**(ResultSet parResultSet, int parPos) Retrieves the value of the designated column in the current row of this `ResultSet` object as an `Array` object of `BigDecimal`.	43
final Object	**getColumn**(int parPos) Retrieves the value of the designated column in the current row of this `ResultSet` object as an `Object`.	44
final Object	**getColumn**(String parColumnName) Retrieves the value of the designated column in the current row of this `ResultSet` object as an `Object`.	44
final Map	**getColumns**() Retrieves a `Map` object containing the values of all columns of the database row.	44
final boolean	**getConnection**() Attempts to establish a connection to the database URL given in the properties.	44
final boolean	**getConnection**(String parUrl, String parUserName, String parPassword) Attempts to establish a connection to the given database URL.	44
final Connection	**getConnectionObject**() Returns the current `Connection` object.	45
final int	**getNumberRows**() Returns the number of rows processed by the SQL query.	45
final ResultSet	**getResultSetObject**() Returns the current `ResultSet` object.	45
final String	**getSqlSyntaxCodeTarget**() Returns the SQL syntax code of the target database.	45
final Date	**getTrialRunEndTime**() Returns the current date after the execution of the SQL query.	45
final String	**getTrialRunErrorMessage**() Returns the error message triggered by the execution of the SQL query.	46
final Date	**getTrialRunStartTime**() Returns the current date before the execution of the SQL query.	46

final long	**getTrialTimeQuantities**(long parPrecision) Returns the response time in a desired precision.	*46*
final String	**getUserName**() Returns the current user's name.	*46*
final boolean	**next**() Moves the cursor forward one row from its current position.	*46*
final boolean	**rollback**() Undoes all changes made in the current transaction and releases any database locks currently held by this `Connection` object.	*46*
final boolean	**setFetchSize**(int parFetchSize) Creates a `Statement` object associtated with this `Connnection` object.	*47*

Constructor Detail

DatabaseAccessor

```
public DatabaseAccessor(String parDatabaseIdent,
                        String parSQLSyntaxCodeTarget,
                        boolean parIsMapper)
```

Constructs a `DatabaseAccessor` object.

DatabaseAccessor

```
public DatabaseAccessor(String parDatabaseIdent,
                        String parSQLSyntaxCodeTarget,
                        String parDriver,
                        boolean parIsMapper)
```

Constructs a `DatabaseAccessor` object.

Method Detail

checkColumnName

```
public final ResultSet checkColumnName(String parTableName,
                                       String parColumnName)
```

Returns the metadata of a given database column.

Parameters:

`parTableName` - The name of the database table.

`parColumnName` - The name of the database column.

Returns:

the current `ResultSet` object if the given database column exist, `null` otherwise.

checkTableName

```
public final ResultSet checkTableName(String parTableName)
```

Returns the metadata of a given database table.

Parameters:

`parTableName` - The name of the database table.

Returns:
the current `ResultSet` object if the given database table exist, `null` otherwise.

---

closeConnection

`public final boolean closeConnection()`

Closes the currently open connection.
Returns:
`false` if the database `connectionObject` could not be closed, or `true` otherwise

---

closeResultSet

`public final boolean closeResultSet(ResultSet parResultSet)`

Closes the given result set.
Parameters:
`parResultSet` - The `ResultSet` object to be closed.
Returns:
`false` if the result set could not be closed, or `true` otherwise.

---

commit

`public final boolean commit()`

Makes the changes permanent which were created by all statements associated with this `Connection` object since the last `commit` or `rollback` was issued.
Returns:
`false` if the changes could not be committed, or `true` otherwise.

---

createStatement

`public final boolean createStatement()`

Creates a `Statement` object associtated with this `Connnection` object.
Returns:
`true` if the `Statement` could be created, or `false` otherwise.

---

executeQuery

`public final boolean executeQuery(String parStmnt)`

Executes the `Statement` object by passing the specified SQL statement to the database. It is used for executing queries formulated using SQL:1999.
Parameters:
`parStmnt` - The SQL statement.
Returns:
`true` if the `Statement` could be executed without any problems, or `false` otherwise.

---

executeQuery

```
public final boolean executeQuery(String parStmnt,
                                  String parSQLSyntaxCodeSource)
```

Executes the `Statement` object by passing the specified SQL statement to the database. It is used for executing queries formulated using a given SQL syntax version.

Parameters:

`parStmnt` - The SQL statement.

`parSQLSyntaxCodeSource` - Rhe SQL syntax version of the SQL statement.

Returns:

`true` if the `Statement` could be executed without any problems, or `false` otherwise.

---

executeQueryTrialRun

```
public final boolean executeQueryTrialRun(String parStmnt,
                                          String parSQLSyntaxCodeSource)
```

Executes the `Statement` object by passing the specified SQL statement to the database. It is used for executing queries formulated using a given SQL syntax version during a trial run.

Parameters:

`parStmnt` - The SQL statement.

`parSQLSyntaxCodeSource` - Rhe SQL syntax version of the SQL statement.

Returns:

`true` if the `Statement` could be executed without any problems, or `false` otherwise.

---

executeUpdate

```
public final boolean executeUpdate(String parStmnt)
```

Executes the `Statement` object by passing the specified SQL statement to the database. It is used for executing updates with SQL statements formulated using SQL:1999.

Parameters:

`parStmnt` - The SQL statement.

Returns:

`true` if the `Statement` could be executed without any problems, or `false` otherwise.

---

executeUpdate

```
public final boolean executeUpdate(String parStmnt,
                                   boolean parIgnoreDuplicate)
```

Executes the `Statement` object by passing the specified SQL statement to the database. It is used for executing updates with SQL statements formulated using SQL:1999. This method ignores duplicates.

Parameters:

`parStmnt` - The SQL statement.

`parIgnoreDuplicate` - Whether duplicates should be ignored.

Returns:

`true` if the `Statement` could be executed without any problems, or `false` otherwise.

---

executeUpdate

```
public final boolean executeUpdate(String parStmnt,
                                   String parSQLSyntaxCodeSource)
```

Executes the `Statement` object by passing the specified SQL statement to the database. It is used for executing updates with rewriting of the query.

Parameters:

`parStmnt` - The SQL statement.

`parSQLSyntaxCodeSource` - The SQL syntax version of the SQL statement.

Returns:

`true` if the `Statement` could be executed without any problems, or `false` otherwise.

---

executeUpdate

```
public final boolean executeUpdate(String parStmnt,
                                   String parSQLSyntaxCodeSource,
                                   boolean parIgnoreDuplicate)
```

Executes the `Statement` object by passing the specified SQL statement to the database. It is used for executing updates with rewriting of the query. This method ignores duplicates.

Parameters:

`parStmnt` - The SQL statement.

`parSQLSyntaxCodeSource` - The SQL syntax version of the SQL statement.

`parIgnoreDuplicate` - Whether duplicates should be ignored.

Returns:

`true` if the `Statement` could be executed without any problems, or `false` otherwise.

---

executeUpdateDirect

```
public final boolean executeUpdateDirect(String parStmnt,
                                         boolean parIgnoreDuplicate)
```

Executes the `Statement` object by passing the specified SQL statement to the database. It is used for executing updates with SQL statements without any rewriting of the SQL statement. This method ignores duplicates.

Parameters:

`parStmnt` - The SQL statement.

`parIgnoreDuplicate` - Whether duplicates should be ignored.

Returns:

`true` if the `Statement` could be executed without any problems, or `false` otherwise.

---

getArrayBigDecimal

```
public final BigDecimal[] getArrayBigDecimal(int parPos)
```

Retrieves the value of the designated column in the current row of this `ResultSet` object as an `Array` object of `BigDecimal`.

Parameters:

`parPos` - The index of the desired column (starting with 1).

Returns:

an `Array` object of `BigDecimal`.

---

getArrayBigDecimal

```
public static final BigDecimal[] getArrayBigDecimal(ResultSet parResultSet,
                                                    int parPos)
```

Retrieves the value of the designated column in the current row of this `ResultSet` object as an `Array` object of `BigDecimal`.

Parameters:

`parResultSet` - The `ResultSet` object.

`parPos` - The index of the desired column (starting with 1).

Returns:

an `Array` object of `BigDecimal`.

---

getColumn

```
public final Object getColumn(int parPos)
```

Retrieves the value of the designated column in the current row of this `ResultSet` object as an `Object`.

Parameters:

`parPos` - The index of the desired column (starting with 1).

Returns:

an `Object` if the access was successasful, and `null` otherwise.

---

getColumn

```
public final Object getColumn(String parColumnName)
```

Retrieves the value of the designated column in the current row of this `ResultSet` object as an `Object`.

Parameters:

`parColumnName` - The name of the required database column.

Returns:

an `Object` if the access was successasful, and `null` otherwise.

---

getColumns

```
public final Map getColumns()
```

Retrieves a `Map` object containing the values of all columns of the database row.

Returns:

a `Map` if the access was successasful, and `null` otherwise.

---

getConnection

```
public final boolean getConnection()
```

Attempts to establish a connection to the database URL given in the properties.

Returns:

`false` if the connection could not be established, or `true` otherwise.

---

getConnection

```
public final boolean getConnection(String parUrl,
                                   String parUserName,
                                   String parPassword)
```

Attempts to establish a connection to the given database URL.

Parameters:
`parUrl` - The database URL.
`parUserName` - The name of the database user.
`parPassword` - The password of the database user.
Returns:
`false` if the connection could not be established, or `true` otherwise.

---

getConnectionObject

```
public final Connection getConnectionObject()
```

Returns the current `Connection` object.
Returns:
the current `Connection` object.

---

getNumberRows

```
public final int getNumberRows()
```

Returns the number of rows processed by the SQL query.
Returns:
the number of rows processed by the SQL query.

---

getResultSetObject

```
public final ResultSet getResultSetObject()
```

Returns the current `ResultSet` object.
Returns:
the current `ResultSet` object.

---

getSqlSyntaxCodeTarget

```
public final String getSqlSyntaxCodeTarget()
```

Returns the SQL syntax code of the target database.
Returns:
the SQL syntax code of the target database.

---

getTrialRunEndTime

```
public final Date getTrialRunEndTime()
```

Returns the current date after the execution of the SQL query.
Returns:
the current date after the execution of the SQL query.

---

getTrialRunErrorMessage

`public final String **getTrialRunErrorMessage**()`

Returns the error message triggered by the execution of the SQL query.
Returns:
the error message triggered by the execution of the SQL query.

---

getTrialRunStartTime

`public final Date **getTrialRunStartTime**()`

Returns the current date before the execution of the SQL query.
Returns:
the current date before the execution of the SQL query.

---

getTrialTimeQuantities

`public final long **getTrialTimeQuantities**(long parPrecision)`

Returns the response time in a desired precision.
Parameters:
`parPrecision` - The exponent (base 10) of the desired presicion.
Returns:
the response time in the desired precision.

---

getUserName

`public final String **getUserName**()`

Returns the current user's name.
Returns:
the current user's name.

---

next

`public final boolean **next**()`

Moves the cursor forward one row from its current position.
Returns:
`false` if there are no further rows to process, or `true` otherwise.

---

rollback

`public final boolean **rollback**()`

Undoes all changes made in the current transaction and releases any database locks currently held by this `Connection` object.
Returns:
`false` if the changes could not be made undone, or `true` otherwise.

---

setFetchSize

```
public final boolean setFetchSize(int parFetchSize)
```

Creates a `Statement` object associtated with this `Connnection` object.

Parameters:

`parFetchSize` - The fetch size.

Returns:

`true` if the fetch size could be modified, or `false` otherwise.

Class DatabaseToExcel

edu.ou.weinmann.repsi.model.util

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.util.DatabaseToExcel
```

public class **DatabaseToExcel**
extends Object

Create an Excel compatible file out of database tables.

Author:

Walter Weinmann

Constructor Summary	*Page*
`DatabaseToExcel()` Constructs a `DatabaseToExcel` object.	*47*

Method Summary		*Page*
`final boolean`	`closeWorkbook()` Closes the current workbook.	*48*
`final boolean`	`createSheet(String parSheetName, int parSheetPosition, ResultSet parResultSet, boolean parHeader)` Creates a new sheet.	*48*
`final boolean`	`createWorkbook(String parFileName)` Creates a new workbook.	*48*
`final String`	`getLastErrorMsg()` Returns the last error message.	*48*

Constructor Detail

DatabaseToExcel

```
public DatabaseToExcel()
```

Constructs a `DatabaseToExcel` object.

Method Detail

closeWorkbook

```
public final boolean closeWorkbook()
```

Closes the current workbook.
Returns:
`true` if the creation of the file was completed without any error.

---

createSheet

```
public final boolean createSheet(String parSheetName,
                                 int parSheetPosition,
                                 ResultSet parResultSet,
                                 boolean parHeader)
```

Creates a new sheet.
Parameters:
`parSheetName` - The name of the worksheet.
`parSheetPosition` - The position of the worksheet in the Excel file.
`parResultSet` - The `ResultSet` to be converted into Excel format.
`parHeader` - Whether a header is required.
Returns:
`true` if the creation of the sheet was completed without any error.

---

createWorkbook

```
public final boolean createWorkbook(String parFileName)
```

Creates a new workbook.
Parameters:
`parFileName` - The complete file name including the directory.
Returns:
`true` if the creation of the file was completed without any error.

---

getLastErrorMsg

```
public final String getLastErrorMsg()
```

Returns the last error message.
Returns:
the last error message.

Interface Global

edu.ou.weinmann.repsi.model.util

All Known Implementing Classes:
DatabaseAccessor

---

public interface **Global**

Provides global constants.

Author:
Walter Weinmann

Field Summary		**Page**
`String`	**COLUMN_NAME_APPLIED_PATTERN_ORDER_BY** Database column name.	52
`String`	**COLUMN_NAME_APPLIED_PATTERN_SELECT_STMNT** Database column name.	53
`String`	**COLUMN_NAME_ARITHMETIC_MEAN** Database column name.	53
`String`	**COLUMN_NAME_GEOMETRIC_MEAN** Database column name.	53
`String`	**COLUMN_NAME_JDBC_DRIVER** Database column name.	53
`String`	**COLUMN_NAME_JDBC_URL** Database column name.	53
`String`	**COLUMN_NAME_KURTOSIS** Database column name.	53
`String`	**COLUMN_NAME_MAXIMUM_VALUE** Database column name.	53
`String`	**COLUMN_NAME_MINIMUM_VALUE** Database column name.	54
`String`	**COLUMN_NAME_NAME** Database column name.	54
`String`	**COLUMN_NAME_NUMBER_OF_VALUES** Database column name.	54
`String`	**COLUMN_NAME_OBJECT** Database column name.	54
`String`	**COLUMN_NAME_ORDER_BY** Database column name.	54
`String`	**COLUMN_NAME_PASSWORD** Database column name.	54
`String`	**COLUMN_NAME_PATTERN_SQL_IDIOM_NAME** Database column name.	54
`String`	**COLUMN_NAME_PERCENTILE_25** Database column name.	54
`String`	**COLUMN_NAME_PERCENTILE_50** Database column name.	55
`String`	**COLUMN_NAME_PERCENTILE_75** Database column name.	55
`String`	**COLUMN_NAME_READINGS** Database column name.	55
`String`	**COLUMN_NAME_SKEWNESS** Database column name.	55
`String`	**COLUMN_NAME_SQL_STATEMENT** Database column name.	55

String	COLUMN_NAME_SQL_SYNTAX_CODE Database column name.	55
String	COLUMN_NAME_STANDARD_DEVIATION Database column name.	55
String	COLUMN_NAME_TABLE_NAME Database column name.	55
String	COLUMN_NAME_UNAPPLIED_PATTERN_ORDER_BY Database column name.	56
String	COLUMN_NAME_UNAPPLIED_PATTERN_SELECT_STMNT Database column name.	56
String	COLUMN_NAME_USER_NAME Database column name.	56
String	COLUMN_NAME_VARIANCE Database column name.	56
String	DATABASE_SCHEMA_IDENTIFIER_MASTER Database schema identifier of the master database.	56
String	DATABASE_SCHEMA_IDENTIFIER_TEST Database schema identifier of the test database.	56
String	DATE_FORMAT_DD_MM_YYYY_HH_MM_SS_SSS_JAVA Date format - timestamp incl. milliseconds - Java format.	57
String	DATE_FORMAT_DD_MM_YYYY_HH_MM_SS_SSS_SQL Date format - timestamp incl. milliseconds - SQL format.	57
String	DATE_FORMAT_DD_MM_YYYY_JAVA Date format - date - Java format.	56
String	DATE_FORMAT_DD_MM_YYYY_SQL Date format - date - SQL format.	57
String	DATE_FORMAT_YYYY_MM_DD_HH_MM_SS_SSS_SQL Date format - timestamp incl. milliseconds - SQL format.	57
long	DAY_FACTOR Day factor in milliseconds.	57
String	ERROR_NOT_YET_IMPLEMENTED Error message - not yet implemented.	57
String	FILE_TYPE_EXCEL File type - Excel.	57
String	FILE_TYPE_XML File type - XML.	57
int	INITIAL_CAPACITY_FOREIGN_KEYS Initial capacity of foreign key columns.	58
int	INITIAL_CAPACITY_PRIMARY_KEYS Initial capacity of primary key columns.	58
String	IS_NULLABLE_YES Is nullable - yes.	58
long	MAX_STATISTICAL_OUTLINE Maximum number of statistical measurement values.	58
long	MAX_TRANSACTION_SIZE Maximum size of rows processed inside one transaction.	58
String	META_DATA_COLUMN_NAME Meta data - column name.	58
String	META_DATA_COLUMN_SIZE Meta data - column size.	58

String	META_DATA_DATA_TYPE Meta data - data type.	58
String	META_DATA_DECIMAL_DIGITS Meta data - decimal digits.	59
String	META_DATA_DECIMAL_FACTOR Meta data - decimal factor.	59
String	META_DATA_FOREIGN_KEY_COLUMN_NAME Meta data - foreign key column name.	59
String	META_DATA_FOREIGN_KEY_NAME Meta data - foreign key name.	59
String	META_DATA_FOREIGN_KEY_TABLE_NAME Meta data - foreign key table name.	59
String	META_DATA_IS_NULLABLE Meta data - is nullable.	59
String	META_DATA_KEY_SEQUENCE Meta data - key sequence number.	59
String	META_DATA_ORDINAL_POSITION Meta data - ordinal position.	60
String	META_DATA_PRIMARY_KEY_COLUMN_NAME Meta data - primary key column name.	60
String	META_DATA_PRIMARY_KEY_NAME Meta data - primary key name.	60
String	META_DATA_PRIMARY_KEY_TABLE_NAME Meta data - primary key table name.	60
String	MINIMAL_VALUE_VARCHAR Minimal value - VARCHAR.	60
String	NULL Null value.	60
String	NUMBER_FORMAT_JAVA Number format - Java format.	60
String	NUMBER_FORMAT_LONG_JAVA Number format - long - Java format.	60
String	OPERATION_CODE_CREATE_TABLE Operation code - create table.	61
String	OPERATION_CODE_DROP_AND_CREATE_TABLE Operation code - drop and create table.	61
String	OPERATION_CODE_DROP_TABLE Operation code - drop table.	61
String	OPERATION_CODE_EXECUTE_QUERY Operation code - execute query.	61
String	OPERATION_CODE_EXECUTE_QUERY_APPLIED Operation code - execute query applied.	61
String	OPERATION_CODE_EXECUTE_QUERY_UNAPPLIED Operation code - execute query unapplied.	61
String	OPERATION_CODE_INSERT_ROW Operation code - insert row.	61
String	OPERATION_TYPE_INSTANCE Operation type - instance.	61
String	OPERATION_TYPE_QUERY Operation type - query.	62

String	**OPERATION_TYPE_SCHEMA** Operation type - schema.	*62*
String	**PROPERTIES_FILE_NAME** Properties file name.	*62*
String	**PROPERTY_PATH_1_DATABASE** Property file path element - level 1 - database.	*62*
String	**PROPERTY_PATH_3_DRIVER** Property file path element - level 3 - driver.	*62*
String	**PROPERTY_PATH_3_PASSWORD** Property file path element - level 3 - password.	*62*
String	**PROPERTY_PATH_3_SQL_SYNTAX_CODE** Property file path element - level 3 - SQL syntax code.	*62*
String	**PROPERTY_PATH_3_URL** Property file path element - level 3 - database URL.	*63*
String	**PROPERTY_PATH_3_USER_NAME** Property file path element - level 3 - user name.	*63*
String	**SEPARATOR_COMMA_SPACE_SINGLE_QUOTE** Separator - comma, space & single quote.	*63*
String	**SEPARATOR_SINGLE_QUOTE_COMMA_SPACE_SINGLE_QUOTE** Separator - single quote, comma, space & single quote.	*63*
String	**SQL_COLUMN_TYPE_CHAR** SQL column type - CHAR.	*63*
String	**SQL_COLUMN_TYPE_VARCHAR2** SQL column type - VARCHAR2.	*63*
String	**SQL_SYNTAX_CODE_ORACLE_10G** SQL syntax code - Oracle 10g Release 2.	*63*
String	**SQL_SYNTAX_CODE_SQL_99** SQL syntax code - standard SQL:1999.	*63*
String	**TRIAL_RUN_STATUS_END_ACTION** Trial run status: end action.	*64*
String	**TRIAL_RUN_STATUS_END_FINALISATION** Trial run status: end finalisation.	*64*
String	**TRIAL_RUN_STATUS_END_INITIALISATION** Trial run status: end initialisation.	*64*
String	**TRIAL_RUN_STATUS_END_PROGRAM** Trial run status: end program.	*64*
String	**TRIAL_RUN_STATUS_START_ACTION** Trial run status: start action.	*64*
String	**TRIAL_RUN_STATUS_START_FINALISATION** Trial run status: start finalisation.	*64*
String	**TRIAL_RUN_STATUS_START_INITIALISATION** Trial run status: start initialisation.	*64*
String	**TRIAL_RUN_STATUS_START_PROGRAM** Trial run status: start program.	*64*

Field Detail

COLUMN_NAME_APPLIED_PATTERN_ORDER_BY

public static final String **COLUMN_NAME_APPLIED_PATTERN_ORDER_BY**

Database column name.

COLUMN_NAME_APPLIED_PATTERN_SELECT_STMNT

```
public static final String COLUMN_NAME_APPLIED_PATTERN_SELECT_STMNT
```

Database column name.

COLUMN_NAME_ARITHMETIC_MEAN

```
public static final String COLUMN_NAME_ARITHMETIC_MEAN
```

Database column name.

COLUMN_NAME_GEOMETRIC_MEAN

```
public static final String COLUMN_NAME_GEOMETRIC_MEAN
```

Database column name.

COLUMN_NAME_JDBC_DRIVER

```
public static final String COLUMN_NAME_JDBC_DRIVER
```

Database column name.

COLUMN_NAME_JDBC_URL

```
public static final String COLUMN_NAME_JDBC_URL
```

Database column name.

COLUMN_NAME_KURTOSIS

```
public static final String COLUMN_NAME_KURTOSIS
```

Database column name.

COLUMN_NAME_MAXIMUM_VALUE

```
public static final String COLUMN_NAME_MAXIMUM_VALUE
```

Database column name.

COLUMN_NAME_MINIMUM_VALUE

`public static final String` **`COLUMN_NAME_MINIMUM_VALUE`**

Database column name.

COLUMN_NAME_NAME

`public static final String` **`COLUMN_NAME_NAME`**

Database column name.

COLUMN_NAME_NUMBER_OF_VALUES

`public static final String` **`COLUMN_NAME_NUMBER_OF_VALUES`**

Database column name.

COLUMN_NAME_OBJECT

`public static final String` **`COLUMN_NAME_OBJECT`**

Database column name.

COLUMN_NAME_ORDER_BY

`public static final String` **`COLUMN_NAME_ORDER_BY`**

Database column name.

COLUMN_NAME_PASSWORD

`public static final String` **`COLUMN_NAME_PASSWORD`**

Database column name.

COLUMN_NAME_PATTERN_SQL_IDIOM_NAME

`public static final String` **`COLUMN_NAME_PATTERN_SQL_IDIOM_NAME`**

Database column name.

COLUMN_NAME_PERCENTILE_25

`public static final String` **`COLUMN_NAME_PERCENTILE_25`**

Database column name.

COLUMN_NAME_PERCENTILE_50

`public static final String` **`COLUMN_NAME_PERCENTILE_50`**

Database column name.

COLUMN_NAME_PERCENTILE_75

`public static final String` **`COLUMN_NAME_PERCENTILE_75`**

Database column name.

COLUMN_NAME_READINGS

`public static final String` **`COLUMN_NAME_READINGS`**

Database column name.

COLUMN_NAME_SKEWNESS

`public static final String` **`COLUMN_NAME_SKEWNESS`**

Database column name.

COLUMN_NAME_SQL_STATEMENT

`public static final String` **`COLUMN_NAME_SQL_STATEMENT`**

Database column name.

COLUMN_NAME_SQL_SYNTAX_CODE

`public static final String` **`COLUMN_NAME_SQL_SYNTAX_CODE`**

Database column name.

COLUMN_NAME_STANDARD_DEVIATION

`public static final String` **`COLUMN_NAME_STANDARD_DEVIATION`**

Database column name.

COLUMN_NAME_TABLE_NAME

`public static final String` **`COLUMN_NAME_TABLE_NAME`**

Database column name.

COLUMN_NAME_UNAPPLIED_PATTERN_ORDER_BY

`public static final String` **`COLUMN_NAME_UNAPPLIED_PATTERN_ORDER_BY`**

Database column name.

COLUMN_NAME_UNAPPLIED_PATTERN_SELECT_STMNT

`public static final String` **`COLUMN_NAME_UNAPPLIED_PATTERN_SELECT_STMNT`**

Database column name.

COLUMN_NAME_USER_NAME

`public static final String` **`COLUMN_NAME_USER_NAME`**

Database column name.

COLUMN_NAME_VARIANCE

`public static final String` **`COLUMN_NAME_VARIANCE`**

Database column name.

DATABASE_SCHEMA_IDENTIFIER_MASTER

`public static final String` **`DATABASE_SCHEMA_IDENTIFIER_MASTER`**

Database schema identifier of the master database.

DATABASE_SCHEMA_IDENTIFIER_TEST

`public static final String` **`DATABASE_SCHEMA_IDENTIFIER_TEST`**

Database schema identifier of the test database.

DATE_FORMAT_DD_MM_YYYY_JAVA

`public static final String` **`DATE_FORMAT_DD_MM_YYYY_JAVA`**

Date format - date - Java format.

DATE_FORMAT_DD_MM_YYYY_SQL

```
public static final String DATE_FORMAT_DD_MM_YYYY_SQL
```

Date format - date - SQL format.

DATE_FORMAT_DD_MM_YYYY_HH_MM_SS_SSS_JAVA

```
public static final String DATE_FORMAT_DD_MM_YYYY_HH_MM_SS_SSS_JAVA
```

Date format - timestamp incl. milliseconds - Java format.

DATE_FORMAT_DD_MM_YYYY_HH_MM_SS_SSS_SQL

```
public static final String DATE_FORMAT_DD_MM_YYYY_HH_MM_SS_SSS_SQL
```

Date format - timestamp incl. milliseconds - SQL format.

DATE_FORMAT_YYYY_MM_DD_HH_MM_SS_SSS_SQL

```
public static final String DATE_FORMAT_YYYY_MM_DD_HH_MM_SS_SSS_SQL
```

Date format - timestamp incl. milliseconds - SQL format.

DAY_FACTOR

```
public static final long DAY_FACTOR
```

Day factor in milliseconds.

ERROR_NOT_YET_IMPLEMENTED

```
public static final String ERROR_NOT_YET_IMPLEMENTED
```

Error message - not yet implemented.

FILE_TYPE_EXCEL

```
public static final String FILE_TYPE_EXCEL
```

File type - Excel.

FILE_TYPE_XML

```
public static final String FILE_TYPE_XML
```

File type - XML.

INITIAL_CAPACITY_FOREIGN_KEYS

```
public static final int INITIAL_CAPACITY_FOREIGN_KEYS
```

Initial capacity of foreign key columns.

INITIAL_CAPACITY_PRIMARY_KEYS

```
public static final int INITIAL_CAPACITY_PRIMARY_KEYS
```

Initial capacity of primary key columns.

IS_NULLABLE_YES

```
public static final String IS_NULLABLE_YES
```

Is nullable - yes.

MAX_STATISTICAL_OUTLINE

```
public static final long MAX_STATISTICAL_OUTLINE
```

Maximum number of statistical measurement values.

MAX_TRANSACTION_SIZE

```
public static final long MAX_TRANSACTION_SIZE
```

Maximum size of rows processed inside one transaction.

META_DATA_COLUMN_NAME

```
public static final String META_DATA_COLUMN_NAME
```

Meta data - column name.

META_DATA_COLUMN_SIZE

```
public static final String META_DATA_COLUMN_SIZE
```

Meta data - column size.

META_DATA_DATA_TYPE

```
public static final String META_DATA_DATA_TYPE
```

Meta data - data type.

META_DATA_DECIMAL_DIGITS

```
public static final String META_DATA_DECIMAL_DIGITS
```

Meta data - decimal digits.

META_DATA_DECIMAL_FACTOR

```
public static final String META_DATA_DECIMAL_FACTOR
```

Meta data - decimal factor.

META_DATA_FOREIGN_KEY_COLUMN_NAME

```
public static final String META_DATA_FOREIGN_KEY_COLUMN_NAME
```

Meta data - foreign key column name.

META_DATA_FOREIGN_KEY_NAME

```
public static final String META_DATA_FOREIGN_KEY_NAME
```

Meta data - foreign key name.

META_DATA_FOREIGN_KEY_TABLE_NAME

```
public static final String META_DATA_FOREIGN_KEY_TABLE_NAME
```

Meta data - foreign key table name.

META_DATA_IS_NULLABLE

```
public static final String META_DATA_IS_NULLABLE
```

Meta data - is nullable.

META_DATA_KEY_SEQUENCE

```
public static final String META_DATA_KEY_SEQUENCE
```

Meta data - key sequence number.

META_DATA_ORDINAL_POSITION

`public static final String` **`META_DATA_ORDINAL_POSITION`**

Meta data - ordinal position.

META_DATA_PRIMARY_KEY_COLUMN_NAME

`public static final String` **`META_DATA_PRIMARY_KEY_COLUMN_NAME`**

Meta data - primary key column name.

META_DATA_PRIMARY_KEY_NAME

`public static final String` **`META_DATA_PRIMARY_KEY_NAME`**

Meta data - primary key name.

META_DATA_PRIMARY_KEY_TABLE_NAME

`public static final String` **`META_DATA_PRIMARY_KEY_TABLE_NAME`**

Meta data - primary key table name.

MINIMAL_VALUE_VARCHAR

`public static final String` **`MINIMAL_VALUE_VARCHAR`**

Minimal value - VARCHAR.

NUMBER_FORMAT_JAVA

`public static final String` **`NUMBER_FORMAT_JAVA`**

Number format - Java format.

NUMBER_FORMAT_LONG_JAVA

`public static final String` **`NUMBER_FORMAT_LONG_JAVA`**

Number format - long - Java format.

NULL

`public static final String` **`NULL`**

Null value.

OPERATION_CODE_CREATE_TABLE

`public static final String` **`OPERATION_CODE_CREATE_TABLE`**

Operation code - create table.

OPERATION_CODE_DROP_AND_CREATE_TABLE

`public static final String` **`OPERATION_CODE_DROP_AND_CREATE_TABLE`**

Operation code - drop and create table.

OPERATION_CODE_DROP_TABLE

`public static final String` **`OPERATION_CODE_DROP_TABLE`**

Operation code - drop table.

OPERATION_CODE_EXECUTE_QUERY

`public static final String` **`OPERATION_CODE_EXECUTE_QUERY`**

Operation code - execute query.

OPERATION_CODE_EXECUTE_QUERY_APPLIED

`public static final String` **`OPERATION_CODE_EXECUTE_QUERY_APPLIED`**

Operation code - execute query applied.

OPERATION_CODE_EXECUTE_QUERY_UNAPPLIED

`public static final String` **`OPERATION_CODE_EXECUTE_QUERY_UNAPPLIED`**

Operation code - execute query unapplied.

OPERATION_CODE_INSERT_ROW

`public static final String` **`OPERATION_CODE_INSERT_ROW`**

Operation code - insert row.

OPERATION_TYPE_INSTANCE

`public static final String` **`OPERATION_TYPE_INSTANCE`**

Operation type - instance.

OPERATION_TYPE_QUERY

```
public static final String OPERATION_TYPE_QUERY
```

Operation type - query.

OPERATION_TYPE_SCHEMA

```
public static final String OPERATION_TYPE_SCHEMA
```

Operation type - schema.

PROPERTIES_FILE_NAME

```
public static final String PROPERTIES_FILE_NAME
```

Properties file name.

PROPERTY_PATH_1_DATABASE

```
public static final String PROPERTY_PATH_1_DATABASE
```

Property file path element - level 1 - database.

PROPERTY_PATH_3_DRIVER

```
public static final String PROPERTY_PATH_3_DRIVER
```

Property file path element - level 3 - driver.

PROPERTY_PATH_3_PASSWORD

```
public static final String PROPERTY_PATH_3_PASSWORD
```

Property file path element - level 3 - password.

PROPERTY_PATH_3_SQL_SYNTAX_CODE

```
public static final String PROPERTY_PATH_3_SQL_SYNTAX_CODE
```

Property file path element - level 3 - SQL syntax code.

PROPERTY_PATH_3_URL

```
public static final String PROPERTY_PATH_3_URL
```

Property file path element - level 3 - database URL.

PROPERTY_PATH_3_USER_NAME

```
public static final String PROPERTY_PATH_3_USER_NAME
```

Property file path element - level 3 - user name.

SEPARATOR_COMMA_SPACE_SINGLE_QUOTE

```
public static final String SEPARATOR_COMMA_SPACE_SINGLE_QUOTE
```

Separator - comma, space & single quote.

SEPARATOR_SINGLE_QUOTE_COMMA_SPACE_SINGLE_QUOTE

```
public static final String SEPARATOR_SINGLE_QUOTE_COMMA_SPACE_SINGLE_QUOTE
```

Separator - single quote, comma, space & single quote.

SQL_COLUMN_TYPE_CHAR

```
public static final String SQL_COLUMN_TYPE_CHAR
```

SQL column type - CHAR.

SQL_COLUMN_TYPE_VARCHAR2

```
public static final String SQL_COLUMN_TYPE_VARCHAR2
```

SQL column type - VARCHAR2.

SQL_SYNTAX_CODE_ORACLE_10G

```
public static final String SQL_SYNTAX_CODE_ORACLE_10G
```

SQL syntax code - Oracle 10g Release 2.

SQL_SYNTAX_CODE_SQL_99

```
public static final String SQL_SYNTAX_CODE_SQL_99
```

SQL syntax code - standard SQL:1999.

TRIAL_RUN_STATUS_END_ACTION

`public static final String` **`TRIAL_RUN_STATUS_END_ACTION`**

Trial run status: end action.

TRIAL_RUN_STATUS_END_FINALISATION

`public static final String` **`TRIAL_RUN_STATUS_END_FINALISATION`**

Trial run status: end finalisation.

TRIAL_RUN_STATUS_END_INITIALISATION

`public static final String` **`TRIAL_RUN_STATUS_END_INITIALISATION`**

Trial run status: end initialisation.

TRIAL_RUN_STATUS_END_PROGRAM

`public static final String` **`TRIAL_RUN_STATUS_END_PROGRAM`**

Trial run status: end program.

TRIAL_RUN_STATUS_START_ACTION

`public static final String` **`TRIAL_RUN_STATUS_START_ACTION`**

Trial run status: start action.

TRIAL_RUN_STATUS_START_FINALISATION

`public static final String` **`TRIAL_RUN_STATUS_START_FINALISATION`**

Trial run status: start finalisation.

TRIAL_RUN_STATUS_START_INITIALISATION

`public static final String` **`TRIAL_RUN_STATUS_START_INITIALISATION`**

Trial run status: start initialisation.

TRIAL_RUN_STATUS_START_PROGRAM

`public static final String` **`TRIAL_RUN_STATUS_START_PROGRAM`**

Trial run status: start program.

Class SQLRewriter

edu.ou.weinmann.repsi.model.util

```
java.lang.Object
  └edu.ou.weinmann.repsi.model.util.SQLRewriter
```

public class **SQLRewriter**
extends Object

Adapts the syntactical variations of different SQL versions by rewriting the SQL statements.

Author:
Walter Weinmann

Constructor Summary	Page
`SQLRewriter()` Constructs a `SQLRewriter` object.	65

Method Summary		Page
`final void`	`deleteStoredDomainDefinitions()` Deletes the stored domain definitions.	65
`final String`	`getLastErrorMsg()` Returns the latest error message.	65
`final String`	`rewrite(String parSQLSyntaxCodeSource, String parSQLSyntaxCodeTarget, String parStmnt)` Rewrites a SQL statement from one syntactical version to another syntactical version.	66

Constructor Detail

SQLRewriter

```
public SQLRewriter()
```

Constructs a `SQLRewriter` object.

Method Detail

deleteStoredDomainDefinitions

```
public final void deleteStoredDomainDefinitions()
```

Deletes the stored domain definitions.

getLastErrorMsg

```
public final String getLastErrorMsg()
```

Returns the latest error message.

Returns:

the latest error message.

rewrite

```
public final String rewrite(String parSQLSyntaxCodeSource,
                            String parSQLSyntaxCodeTarget,
                            String parStmnt)
```

Rewrites a SQL statement from one syntactical version to another syntactical version.

Parameters:

`parSQLSyntaxCodeSource` - The SQL syntax version of input statement.
`parSQLSyntaxCodeTarget` - The SQL syntax version of output statement.
`parStmnt` - The SQL statement.

Returns:

the SQL statement in the new syntax version.