| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Statement.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/sql/SQLXML.html)   [**NEXT CLASS**](http://docs.google.com/java/sql/Struct.html) | [**FRAMES**](http://docs.google.com/index.html?java/sql/Statement.html)    [**NO FRAMES**](http://docs.google.com/Statement.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#3dy6vkm) | CONSTR | [METHOD](#35nkun2) |

## **java.sql**

Interface Statement

**All Superinterfaces:** [Wrapper](http://docs.google.com/java/sql/Wrapper.html) **All Known Subinterfaces:** [CallableStatement](http://docs.google.com/java/sql/CallableStatement.html), [PreparedStatement](http://docs.google.com/java/sql/PreparedStatement.html)

public interface **Statement**extends [Wrapper](http://docs.google.com/java/sql/Wrapper.html)

The object used for executing a static SQL statement and returning the results it produces.

By default, only one ResultSet object per Statement object can be open at the same time. Therefore, if the reading of one ResultSet object is interleaved with the reading of another, each must have been generated by different Statement objects. All execution methods in the Statement interface implicitly close a statment's current ResultSet object if an open one exists.

**See Also:**[Connection.createStatement()](http://docs.google.com/java/sql/Connection.html#createStatement()), [ResultSet](http://docs.google.com/java/sql/ResultSet.html)

| **Field Summary** | |
| --- | --- |
| static int | [**CLOSE\_ALL\_RESULTS**](http://docs.google.com/java/sql/Statement.html#CLOSE_ALL_RESULTS)            The constant indicating that all ResultSet objects that have previously been kept open should be closed when calling getMoreResults. |
| static int | [**CLOSE\_CURRENT\_RESULT**](http://docs.google.com/java/sql/Statement.html#CLOSE_CURRENT_RESULT)            The constant indicating that the current ResultSet object should be closed when calling getMoreResults. |
| static int | [**EXECUTE\_FAILED**](http://docs.google.com/java/sql/Statement.html#EXECUTE_FAILED)            The constant indicating that an error occured while executing a batch statement. |
| static int | [**KEEP\_CURRENT\_RESULT**](http://docs.google.com/java/sql/Statement.html#KEEP_CURRENT_RESULT)            The constant indicating that the current ResultSet object should not be closed when calling getMoreResults. |
| static int | [**NO\_GENERATED\_KEYS**](http://docs.google.com/java/sql/Statement.html#NO_GENERATED_KEYS)            The constant indicating that generated keys should not be made available for retrieval. |
| static int | [**RETURN\_GENERATED\_KEYS**](http://docs.google.com/java/sql/Statement.html#RETURN_GENERATED_KEYS)            The constant indicating that generated keys should be made available for retrieval. |
| static int | [**SUCCESS\_NO\_INFO**](http://docs.google.com/java/sql/Statement.html#SUCCESS_NO_INFO)            The constant indicating that a batch statement executed successfully but that no count of the number of rows it affected is available. |

| **Method Summary** | |
| --- | --- |
| void | [**addBatch**](http://docs.google.com/java/sql/Statement.html#addBatch(java.lang.String))([String](http://docs.google.com/java/lang/String.html) sql)            Adds the given SQL command to the current list of commmands for this Statement object. |
| void | [**cancel**](http://docs.google.com/java/sql/Statement.html#cancel())()            Cancels this Statement object if both the DBMS and driver support aborting an SQL statement. |
| void | [**clearBatch**](http://docs.google.com/java/sql/Statement.html#clearBatch())()            Empties this Statement object's current list of SQL commands. |
| void | [**clearWarnings**](http://docs.google.com/java/sql/Statement.html#clearWarnings())()            Clears all the warnings reported on this Statement object. |
| void | [**close**](http://docs.google.com/java/sql/Statement.html#close())()            Releases this Statement object's database and JDBC resources immediately instead of waiting for this to happen when it is automatically closed. |
| boolean | [**execute**](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String))([String](http://docs.google.com/java/lang/String.html) sql)            Executes the given SQL statement, which may return multiple results. |
| boolean | [**execute**](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) sql, int autoGeneratedKeys)            Executes the given SQL statement, which may return multiple results, and signals the driver that any auto-generated keys should be made available for retrieval. |
| boolean | [**execute**](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String,%20int%5B%5D))([String](http://docs.google.com/java/lang/String.html) sql, int[] columnIndexes)            Executes the given SQL statement, which may return multiple results, and signals the driver that the auto-generated keys indicated in the given array should be made available for retrieval. |
| boolean | [**execute**](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String,%20java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html) sql, [String](http://docs.google.com/java/lang/String.html)[] columnNames)            Executes the given SQL statement, which may return multiple results, and signals the driver that the auto-generated keys indicated in the given array should be made available for retrieval. |
| int[] | [**executeBatch**](http://docs.google.com/java/sql/Statement.html#executeBatch())()            Submits a batch of commands to the database for execution and if all commands execute successfully, returns an array of update counts. |
| [ResultSet](http://docs.google.com/java/sql/ResultSet.html) | [**executeQuery**](http://docs.google.com/java/sql/Statement.html#executeQuery(java.lang.String))([String](http://docs.google.com/java/lang/String.html) sql)            Executes the given SQL statement, which returns a single ResultSet object. |
| int | [**executeUpdate**](http://docs.google.com/java/sql/Statement.html#executeUpdate(java.lang.String))([String](http://docs.google.com/java/lang/String.html) sql)            Executes the given SQL statement, which may be an INSERT, UPDATE, or DELETE statement or an SQL statement that returns nothing, such as an SQL DDL statement. |
| int | [**executeUpdate**](http://docs.google.com/java/sql/Statement.html#executeUpdate(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) sql, int autoGeneratedKeys)            Executes the given SQL statement and signals the driver with the given flag about whether the auto-generated keys produced by this Statement object should be made available for retrieval. |
| int | [**executeUpdate**](http://docs.google.com/java/sql/Statement.html#executeUpdate(java.lang.String,%20int%5B%5D))([String](http://docs.google.com/java/lang/String.html) sql, int[] columnIndexes)            Executes the given SQL statement and signals the driver that the auto-generated keys indicated in the given array should be made available for retrieval. |
| int | [**executeUpdate**](http://docs.google.com/java/sql/Statement.html#executeUpdate(java.lang.String,%20java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html) sql, [String](http://docs.google.com/java/lang/String.html)[] columnNames)            Executes the given SQL statement and signals the driver that the auto-generated keys indicated in the given array should be made available for retrieval. |
| [Connection](http://docs.google.com/java/sql/Connection.html) | [**getConnection**](http://docs.google.com/java/sql/Statement.html#getConnection())()            Retrieves the Connection object that produced this Statement object. |
| int | [**getFetchDirection**](http://docs.google.com/java/sql/Statement.html#getFetchDirection())()            Retrieves the direction for fetching rows from database tables that is the default for result sets generated from this Statement object. |
| int | [**getFetchSize**](http://docs.google.com/java/sql/Statement.html#getFetchSize())()            Retrieves the number of result set rows that is the default fetch size for ResultSet objects generated from this Statement object. |
| [ResultSet](http://docs.google.com/java/sql/ResultSet.html) | [**getGeneratedKeys**](http://docs.google.com/java/sql/Statement.html#getGeneratedKeys())()            Retrieves any auto-generated keys created as a result of executing this Statement object. |
| int | [**getMaxFieldSize**](http://docs.google.com/java/sql/Statement.html#getMaxFieldSize())()            Retrieves the maximum number of bytes that can be returned for character and binary column values in a ResultSet object produced by this Statement object. |
| int | [**getMaxRows**](http://docs.google.com/java/sql/Statement.html#getMaxRows())()            Retrieves the maximum number of rows that a ResultSet object produced by this Statement object can contain. |
| boolean | [**getMoreResults**](http://docs.google.com/java/sql/Statement.html#getMoreResults())()            Moves to this Statement object's next result, returns true if it is a ResultSet object, and implicitly closes any current ResultSet object(s) obtained with the method getResultSet. |
| boolean | [**getMoreResults**](http://docs.google.com/java/sql/Statement.html#getMoreResults(int))(int current)            Moves to this Statement object's next result, deals with any current ResultSet object(s) according to the instructions specified by the given flag, and returns true if the next result is a ResultSet object. |
| int | [**getQueryTimeout**](http://docs.google.com/java/sql/Statement.html#getQueryTimeout())()            Retrieves the number of seconds the driver will wait for a Statement object to execute. |
| [ResultSet](http://docs.google.com/java/sql/ResultSet.html) | [**getResultSet**](http://docs.google.com/java/sql/Statement.html#getResultSet())()            Retrieves the current result as a ResultSet object. |
| int | [**getResultSetConcurrency**](http://docs.google.com/java/sql/Statement.html#getResultSetConcurrency())()            Retrieves the result set concurrency for ResultSet objects generated by this Statement object. |
| int | [**getResultSetHoldability**](http://docs.google.com/java/sql/Statement.html#getResultSetHoldability())()            Retrieves the result set holdability for ResultSet objects generated by this Statement object. |
| int | [**getResultSetType**](http://docs.google.com/java/sql/Statement.html#getResultSetType())()            Retrieves the result set type for ResultSet objects generated by this Statement object. |
| int | [**getUpdateCount**](http://docs.google.com/java/sql/Statement.html#getUpdateCount())()            Retrieves the current result as an update count; if the result is a ResultSet object or there are no more results, -1 is returned. |
| [SQLWarning](http://docs.google.com/java/sql/SQLWarning.html) | [**getWarnings**](http://docs.google.com/java/sql/Statement.html#getWarnings())()            Retrieves the first warning reported by calls on this Statement object. |
| boolean | [**isClosed**](http://docs.google.com/java/sql/Statement.html#isClosed())()            Retrieves whether this Statement object has been closed. |
| boolean | [**isPoolable**](http://docs.google.com/java/sql/Statement.html#isPoolable())()            Returns a value indicating whether the Statement is poolable or not. |
| void | [**setCursorName**](http://docs.google.com/java/sql/Statement.html#setCursorName(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Sets the SQL cursor name to the given String, which will be used by subsequent Statement object execute methods. |
| void | [**setEscapeProcessing**](http://docs.google.com/java/sql/Statement.html#setEscapeProcessing(boolean))(boolean enable)            Sets escape processing on or off. |
| void | [**setFetchDirection**](http://docs.google.com/java/sql/Statement.html#setFetchDirection(int))(int direction)            Gives the driver a hint as to the direction in which rows will be processed in ResultSet objects created using this Statement object. |
| void | [**setFetchSize**](http://docs.google.com/java/sql/Statement.html#setFetchSize(int))(int rows)            Gives the JDBC driver a hint as to the number of rows that should be fetched from the database when more rows are needed for ResultSet objects genrated by this Statement. |
| void | [**setMaxFieldSize**](http://docs.google.com/java/sql/Statement.html#setMaxFieldSize(int))(int max)            Sets the limit for the maximum number of bytes that can be returned for character and binary column values in a ResultSet object produced by this Statement object. |
| void | [**setMaxRows**](http://docs.google.com/java/sql/Statement.html#setMaxRows(int))(int max)            Sets the limit for the maximum number of rows that any ResultSet object generated by this Statement object can contain to the given number. |
| void | [**setPoolable**](http://docs.google.com/java/sql/Statement.html#setPoolable(boolean))(boolean poolable)            Requests that a Statement be pooled or not pooled. |
| void | [**setQueryTimeout**](http://docs.google.com/java/sql/Statement.html#setQueryTimeout(int))(int seconds)            Sets the number of seconds the driver will wait for a Statement object to execute to the given number of seconds. |

| **Methods inherited from interface java.sql.**[**Wrapper**](http://docs.google.com/java/sql/Wrapper.html) |
| --- |
| [isWrapperFor](http://docs.google.com/java/sql/Wrapper.html#isWrapperFor(java.lang.Class)), [unwrap](http://docs.google.com/java/sql/Wrapper.html#unwrap(java.lang.Class)) |

| **Field Detail** |
| --- |

### CLOSE\_CURRENT\_RESULT

static final int **CLOSE\_CURRENT\_RESULT**

The constant indicating that the current ResultSet object should be closed when calling getMoreResults.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Statement.CLOSE_CURRENT_RESULT)

### KEEP\_CURRENT\_RESULT

static final int **KEEP\_CURRENT\_RESULT**

The constant indicating that the current ResultSet object should not be closed when calling getMoreResults.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Statement.KEEP_CURRENT_RESULT)

### CLOSE\_ALL\_RESULTS

static final int **CLOSE\_ALL\_RESULTS**

The constant indicating that all ResultSet objects that have previously been kept open should be closed when calling getMoreResults.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Statement.CLOSE_ALL_RESULTS)

### SUCCESS\_NO\_INFO

static final int **SUCCESS\_NO\_INFO**

The constant indicating that a batch statement executed successfully but that no count of the number of rows it affected is available.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Statement.SUCCESS_NO_INFO)

### EXECUTE\_FAILED

static final int **EXECUTE\_FAILED**

The constant indicating that an error occured while executing a batch statement.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Statement.EXECUTE_FAILED)

### RETURN\_GENERATED\_KEYS

static final int **RETURN\_GENERATED\_KEYS**

The constant indicating that generated keys should be made available for retrieval.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Statement.RETURN_GENERATED_KEYS)

### NO\_GENERATED\_KEYS

static final int **NO\_GENERATED\_KEYS**

The constant indicating that generated keys should not be made available for retrieval.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.sql.Statement.NO_GENERATED_KEYS)

| **Method Detail** |
| --- |

### executeQuery

[ResultSet](http://docs.google.com/java/sql/ResultSet.html) **executeQuery**([String](http://docs.google.com/java/lang/String.html) sql)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Executes the given SQL statement, which returns a single ResultSet object.

**Parameters:**sql - an SQL statement to be sent to the database, typically a static SQL SELECT statement **Returns:**a ResultSet object that contains the data produced by the given query; never null **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the given SQL statement produces anything other than a single ResultSet object

### executeUpdate

int **executeUpdate**([String](http://docs.google.com/java/lang/String.html) sql)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Executes the given SQL statement, which may be an INSERT, UPDATE, or DELETE statement or an SQL statement that returns nothing, such as an SQL DDL statement.

**Parameters:**sql - an SQL Data Manipulation Language (DML) statement, such as INSERT, UPDATE or DELETE; or an SQL statement that returns nothing, such as a DDL statement. **Returns:**either (1) the row count for SQL Data Manipulation Language (DML) statements or (2) 0 for SQL statements that return nothing **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the given SQL statement produces a ResultSet object

### close

void **close**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Releases this Statement object's database and JDBC resources immediately instead of waiting for this to happen when it is automatically closed. It is generally good practice to release resources as soon as you are finished with them to avoid tying up database resources.

Calling the method close on a Statement object that is already closed has no effect.

**Note:**When a Statement object is closed, its current ResultSet object, if one exists, is also closed.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs

### getMaxFieldSize

int **getMaxFieldSize**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the maximum number of bytes that can be returned for character and binary column values in a ResultSet object produced by this Statement object. This limit applies only to BINARY, VARBINARY, LONGVARBINARY, CHAR, VARCHAR, NCHAR, NVARCHAR, LONGNVARCHAR and LONGVARCHAR columns. If the limit is exceeded, the excess data is silently discarded.

**Returns:**the current column size limit for columns storing character and binary values; zero means there is no limit **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**See Also:**[setMaxFieldSize(int)](http://docs.google.com/java/sql/Statement.html#setMaxFieldSize(int))

### setMaxFieldSize

void **setMaxFieldSize**(int max)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the limit for the maximum number of bytes that can be returned for character and binary column values in a ResultSet object produced by this Statement object. This limit applies only to BINARY, VARBINARY, LONGVARBINARY, CHAR, VARCHAR, NCHAR, NVARCHAR, LONGNVARCHAR and LONGVARCHAR fields. If the limit is exceeded, the excess data is silently discarded. For maximum portability, use values greater than 256.

**Parameters:**max - the new column size limit in bytes; zero means there is no limit **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the condition max >= 0 is not satisfied**See Also:**[getMaxFieldSize()](http://docs.google.com/java/sql/Statement.html#getMaxFieldSize())

### getMaxRows

int **getMaxRows**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the maximum number of rows that a ResultSet object produced by this Statement object can contain. If this limit is exceeded, the excess rows are silently dropped.

**Returns:**the current maximum number of rows for a ResultSet object produced by this Statement object; zero means there is no limit **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**See Also:**[setMaxRows(int)](http://docs.google.com/java/sql/Statement.html#setMaxRows(int))

### setMaxRows

void **setMaxRows**(int max)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the limit for the maximum number of rows that any ResultSet object generated by this Statement object can contain to the given number. If the limit is exceeded, the excess rows are silently dropped.

**Parameters:**max - the new max rows limit; zero means there is no limit **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the condition max >= 0 is not satisfied**See Also:**[getMaxRows()](http://docs.google.com/java/sql/Statement.html#getMaxRows())

### setEscapeProcessing

void **setEscapeProcessing**(boolean enable)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets escape processing on or off. If escape scanning is on (the default), the driver will do escape substitution before sending the SQL statement to the database. Note: Since prepared statements have usually been parsed prior to making this call, disabling escape processing for PreparedStatements objects will have no effect.

**Parameters:**enable - true to enable escape processing; false to disable it **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement

### getQueryTimeout

int **getQueryTimeout**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the number of seconds the driver will wait for a Statement object to execute. If the limit is exceeded, a SQLException is thrown.

**Returns:**the current query timeout limit in seconds; zero means there is no limit **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**See Also:**[setQueryTimeout(int)](http://docs.google.com/java/sql/Statement.html#setQueryTimeout(int))

### setQueryTimeout

void **setQueryTimeout**(int seconds)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the number of seconds the driver will wait for a Statement object to execute to the given number of seconds. If the limit is exceeded, an SQLException is thrown. A JDBC driver must apply this limit to the execute, executeQuery and executeUpdate methods. JDBC driver implementations may also apply this limit to ResultSet methods (consult your driver vendor documentation for details).

**Parameters:**seconds - the new query timeout limit in seconds; zero means there is no limit **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the condition seconds >= 0 is not satisfied**See Also:**[getQueryTimeout()](http://docs.google.com/java/sql/Statement.html#getQueryTimeout())

### cancel

void **cancel**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Cancels this Statement object if both the DBMS and driver support aborting an SQL statement. This method can be used by one thread to cancel a statement that is being executed by another thread.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method

### getWarnings

[SQLWarning](http://docs.google.com/java/sql/SQLWarning.html) **getWarnings**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the first warning reported by calls on this Statement object. Subsequent Statement object warnings will be chained to this SQLWarning object.

The warning chain is automatically cleared each time a statement is (re)executed. This method may not be called on a closed Statement object; doing so will cause an SQLException to be thrown.

**Note:** If you are processing a ResultSet object, any warnings associated with reads on that ResultSet object will be chained on it rather than on the Statement object that produced it.

**Returns:**the first SQLWarning object or null if there are no warnings **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement

### clearWarnings

void **clearWarnings**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Clears all the warnings reported on this Statement object. After a call to this method, the method getWarnings will return null until a new warning is reported for this Statement object.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement

### setCursorName

void **setCursorName**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Sets the SQL cursor name to the given String, which will be used by subsequent Statement object execute methods. This name can then be used in SQL positioned update or delete statements to identify the current row in the ResultSet object generated by this statement. If the database does not support positioned update/delete, this method is a noop. To insure that a cursor has the proper isolation level to support updates, the cursor's SELECT statement should have the form SELECT FOR UPDATE. If FOR UPDATE is not present, positioned updates may fail.

**Note:** By definition, the execution of positioned updates and deletes must be done by a different Statement object than the one that generated the ResultSet object being used for positioning. Also, cursor names must be unique within a connection.

**Parameters:**name - the new cursor name, which must be unique within a connection **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method

### execute

boolean **execute**([String](http://docs.google.com/java/lang/String.html) sql)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Executes the given SQL statement, which may return multiple results. In some (uncommon) situations, a single SQL statement may return multiple result sets and/or update counts. Normally you can ignore this unless you are (1) executing a stored procedure that you know may return multiple results or (2) you are dynamically executing an unknown SQL string.

The execute method executes an SQL statement and indicates the form of the first result. You must then use the methods getResultSet or getUpdateCount to retrieve the result, and getMoreResults to move to any subsequent result(s).

**Parameters:**sql - any SQL statement **Returns:**true if the first result is a ResultSet object; false if it is an update count or there are no results **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**See Also:**[getResultSet()](http://docs.google.com/java/sql/Statement.html#getResultSet()), [getUpdateCount()](http://docs.google.com/java/sql/Statement.html#getUpdateCount()), [getMoreResults()](http://docs.google.com/java/sql/Statement.html#getMoreResults())

### getResultSet

[ResultSet](http://docs.google.com/java/sql/ResultSet.html) **getResultSet**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the current result as a ResultSet object. This method should be called only once per result.

**Returns:**the current result as a ResultSet object or null if the result is an update count or there are no more results **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**See Also:**[execute(java.lang.String)](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String))

### getUpdateCount

int **getUpdateCount**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the current result as an update count; if the result is a ResultSet object or there are no more results, -1 is returned. This method should be called only once per result.

**Returns:**the current result as an update count; -1 if the current result is a ResultSet object or there are no more results **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**See Also:**[execute(java.lang.String)](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String))

### getMoreResults

boolean **getMoreResults**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Moves to this Statement object's next result, returns true if it is a ResultSet object, and implicitly closes any current ResultSet object(s) obtained with the method getResultSet.

There are no more results when the following is true:

// stmt is a Statement object  
 ((stmt.getMoreResults() == false) && (stmt.getUpdateCount() == -1))

**Returns:**true if the next result is a ResultSet object; false if it is an update count or there are no more results **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**See Also:**[execute(java.lang.String)](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String))

### setFetchDirection

void **setFetchDirection**(int direction)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Gives the driver a hint as to the direction in which rows will be processed in ResultSet objects created using this Statement object. The default value is ResultSet.FETCH\_FORWARD.

Note that this method sets the default fetch direction for result sets generated by this Statement object. Each result set has its own methods for getting and setting its own fetch direction.

**Parameters:**direction - the initial direction for processing rows **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the given direction is not one of ResultSet.FETCH\_FORWARD, ResultSet.FETCH\_REVERSE, or ResultSet.FETCH\_UNKNOWN**Since:** 1.2 **See Also:**[getFetchDirection()](http://docs.google.com/java/sql/Statement.html#getFetchDirection())

### getFetchDirection

int **getFetchDirection**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the direction for fetching rows from database tables that is the default for result sets generated from this Statement object. If this Statement object has not set a fetch direction by calling the method setFetchDirection, the return value is implementation-specific.

**Returns:**the default fetch direction for result sets generated from this Statement object **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**Since:** 1.2 **See Also:**[setFetchDirection(int)](http://docs.google.com/java/sql/Statement.html#setFetchDirection(int))

### setFetchSize

void **setFetchSize**(int rows)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Gives the JDBC driver a hint as to the number of rows that should be fetched from the database when more rows are needed for ResultSet objects genrated by this Statement. If the value specified is zero, then the hint is ignored. The default value is zero.

**Parameters:**rows - the number of rows to fetch **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the condition rows >= 0 is not satisfied.**Since:** 1.2 **See Also:**[getFetchSize()](http://docs.google.com/java/sql/Statement.html#getFetchSize())

### getFetchSize

int **getFetchSize**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the number of result set rows that is the default fetch size for ResultSet objects generated from this Statement object. If this Statement object has not set a fetch size by calling the method setFetchSize, the return value is implementation-specific.

**Returns:**the default fetch size for result sets generated from this Statement object **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**Since:** 1.2 **See Also:**[setFetchSize(int)](http://docs.google.com/java/sql/Statement.html#setFetchSize(int))

### getResultSetConcurrency

int **getResultSetConcurrency**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the result set concurrency for ResultSet objects generated by this Statement object.

**Returns:**either ResultSet.CONCUR\_READ\_ONLY or ResultSet.CONCUR\_UPDATABLE **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**Since:** 1.2

### getResultSetType

int **getResultSetType**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the result set type for ResultSet objects generated by this Statement object.

**Returns:**one of ResultSet.TYPE\_FORWARD\_ONLY, ResultSet.TYPE\_SCROLL\_INSENSITIVE, or ResultSet.TYPE\_SCROLL\_SENSITIVE **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**Since:** 1.2

### addBatch

void **addBatch**([String](http://docs.google.com/java/lang/String.html) sql)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Adds the given SQL command to the current list of commmands for this Statement object. The commands in this list can be executed as a batch by calling the method executeBatch.

**Parameters:**sql - typically this is a SQL INSERT or UPDATE statement **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the driver does not support batch updates**Since:** 1.2 **See Also:**[executeBatch()](http://docs.google.com/java/sql/Statement.html#executeBatch()), [DatabaseMetaData.supportsBatchUpdates()](http://docs.google.com/java/sql/DatabaseMetaData.html#supportsBatchUpdates())

### clearBatch

void **clearBatch**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Empties this Statement object's current list of SQL commands.

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the driver does not support batch updates**Since:** 1.2 **See Also:**[addBatch(java.lang.String)](http://docs.google.com/java/sql/Statement.html#addBatch(java.lang.String)), [DatabaseMetaData.supportsBatchUpdates()](http://docs.google.com/java/sql/DatabaseMetaData.html#supportsBatchUpdates())

### executeBatch

int[] **executeBatch**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Submits a batch of commands to the database for execution and if all commands execute successfully, returns an array of update counts. The int elements of the array that is returned are ordered to correspond to the commands in the batch, which are ordered according to the order in which they were added to the batch. The elements in the array returned by the method executeBatch may be one of the following:

1. A number greater than or equal to zero -- indicates that the command was processed successfully and is an update count giving the number of rows in the database that were affected by the command's execution
2. A value of SUCCESS\_NO\_INFO -- indicates that the command was processed successfully but that the number of rows affected is unknown  
   If one of the commands in a batch update fails to execute properly, this method throws a BatchUpdateException, and a JDBC driver may or may not continue to process the remaining commands in the batch. However, the driver's behavior must be consistent with a particular DBMS, either always continuing to process commands or never continuing to process commands. If the driver continues processing after a failure, the array returned by the method BatchUpdateException.getUpdateCounts will contain as many elements as there are commands in the batch, and at least one of the elements will be the following:
3. A value of EXECUTE\_FAILED -- indicates that the command failed to execute successfully and occurs only if a driver continues to process commands after a command fails

The possible implementations and return values have been modified in the Java 2 SDK, Standard Edition, version 1.3 to accommodate the option of continuing to proccess commands in a batch update after a BatchUpdateException obejct has been thrown.

**Returns:**an array of update counts containing one element for each command in the batch. The elements of the array are ordered according to the order in which commands were added to the batch. **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the driver does not support batch statements. Throws [BatchUpdateException](http://docs.google.com/java/sql/BatchUpdateException.html) (a subclass of SQLException) if one of the commands sent to the database fails to execute properly or attempts to return a result set.**Since:** 1.3 **See Also:**[addBatch(java.lang.String)](http://docs.google.com/java/sql/Statement.html#addBatch(java.lang.String)), [DatabaseMetaData.supportsBatchUpdates()](http://docs.google.com/java/sql/DatabaseMetaData.html#supportsBatchUpdates())

### getConnection

[Connection](http://docs.google.com/java/sql/Connection.html) **getConnection**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the Connection object that produced this Statement object.

**Returns:**the connection that produced this statement **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**Since:** 1.2

### getMoreResults

boolean **getMoreResults**(int current)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Moves to this Statement object's next result, deals with any current ResultSet object(s) according to the instructions specified by the given flag, and returns true if the next result is a ResultSet object.

There are no more results when the following is true:

// stmt is a Statement object  
 ((stmt.getMoreResults(current) == false) && (stmt.getUpdateCount() == -1))

**Parameters:**current - one of the following Statement constants indicating what should happen to current ResultSet objects obtained using the method getResultSet: Statement.CLOSE\_CURRENT\_RESULT, Statement.KEEP\_CURRENT\_RESULT, or Statement.CLOSE\_ALL\_RESULTS **Returns:**true if the next result is a ResultSet object; false if it is an update count or there are no more results **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the argument supplied is not one of the following: Statement.CLOSE\_CURRENT\_RESULT, Statement.KEEP\_CURRENT\_RESULT or Statement.CLOSE\_ALL\_RESULTS [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if DatabaseMetaData.supportsMultipleOpenResults returns false and either Statement.KEEP\_CURRENT\_RESULT or Statement.CLOSE\_ALL\_RESULTS are supplied as the argument.**Since:** 1.4 **See Also:**[execute(java.lang.String)](http://docs.google.com/java/sql/Statement.html#execute(java.lang.String))

### getGeneratedKeys

[ResultSet](http://docs.google.com/java/sql/ResultSet.html) **getGeneratedKeys**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves any auto-generated keys created as a result of executing this Statement object. If this Statement object did not generate any keys, an empty ResultSet object is returned.

**Note:**If the columns which represent the auto-generated keys were not specified, the JDBC driver implementation will determine the columns which best represent the auto-generated keys.

**Returns:**a ResultSet object containing the auto-generated key(s) generated by the execution of this Statement object **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

### executeUpdate

int **executeUpdate**([String](http://docs.google.com/java/lang/String.html) sql,  
 int autoGeneratedKeys)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Executes the given SQL statement and signals the driver with the given flag about whether the auto-generated keys produced by this Statement object should be made available for retrieval. The driver will ignore the flag if the SQL statement is not an INSERT statement, or an SQL statement able to return auto-generated keys (the list of such statements is vendor-specific).

**Parameters:**sql - an SQL Data Manipulation Language (DML) statement, such as INSERT, UPDATE or DELETE; or an SQL statement that returns nothing, such as a DDL statement.autoGeneratedKeys - a flag indicating whether auto-generated keys should be made available for retrieval; one of the following constants: Statement.RETURN\_GENERATED\_KEYS Statement.NO\_GENERATED\_KEYS **Returns:**either (1) the row count for SQL Data Manipulation Language (DML) statements or (2) 0 for SQL statements that return nothing **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement, the given SQL statement returns a ResultSet object, or the given constant is not one of those allowed [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method with a constant of Statement.RETURN\_GENERATED\_KEYS**Since:** 1.4

### executeUpdate

int **executeUpdate**([String](http://docs.google.com/java/lang/String.html) sql,  
 int[] columnIndexes)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Executes the given SQL statement and signals the driver that the auto-generated keys indicated in the given array should be made available for retrieval. This array contains the indexes of the columns in the target table that contain the auto-generated keys that should be made available. The driver will ignore the array if the SQL statement is not an INSERT statement, or an SQL statement able to return auto-generated keys (the list of such statements is vendor-specific).

**Parameters:**sql - an SQL Data Manipulation Language (DML) statement, such as INSERT, UPDATE or DELETE; or an SQL statement that returns nothing, such as a DDL statement.columnIndexes - an array of column indexes indicating the columns that should be returned from the inserted row **Returns:**either (1) the row count for SQL Data Manipulation Language (DML) statements or (2) 0 for SQL statements that return nothing **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement, the SQL statement returns a ResultSet object, or the second argument supplied to this method is not an int array whose elements are valid column indexes [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

### executeUpdate

int **executeUpdate**([String](http://docs.google.com/java/lang/String.html) sql,  
 [String](http://docs.google.com/java/lang/String.html)[] columnNames)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Executes the given SQL statement and signals the driver that the auto-generated keys indicated in the given array should be made available for retrieval. This array contains the names of the columns in the target table that contain the auto-generated keys that should be made available. The driver will ignore the array if the SQL statement is not an INSERT statement, or an SQL statement able to return auto-generated keys (the list of such statements is vendor-specific).

**Parameters:**sql - an SQL Data Manipulation Language (DML) statement, such as INSERT, UPDATE or DELETE; or an SQL statement that returns nothing, such as a DDL statement.columnNames - an array of the names of the columns that should be returned from the inserted row **Returns:**either the row count for INSERT, UPDATE, or DELETE statements, or 0 for SQL statements that return nothing **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement, the SQL statement returns a ResultSet object, or the second argument supplied to this method is not a String array whose elements are valid column names [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4

### execute

boolean **execute**([String](http://docs.google.com/java/lang/String.html) sql,  
 int autoGeneratedKeys)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Executes the given SQL statement, which may return multiple results, and signals the driver that any auto-generated keys should be made available for retrieval. The driver will ignore this signal if the SQL statement is not an INSERT statement, or an SQL statement able to return auto-generated keys (the list of such statements is vendor-specific).

In some (uncommon) situations, a single SQL statement may return multiple result sets and/or update counts. Normally you can ignore this unless you are (1) executing a stored procedure that you know may return multiple results or (2) you are dynamically executing an unknown SQL string.

The execute method executes an SQL statement and indicates the form of the first result. You must then use the methods getResultSet or getUpdateCount to retrieve the result, and getMoreResults to move to any subsequent result(s).

**Parameters:**sql - any SQL statementautoGeneratedKeys - a constant indicating whether auto-generated keys should be made available for retrieval using the method getGeneratedKeys; one of the following constants: Statement.RETURN\_GENERATED\_KEYS or Statement.NO\_GENERATED\_KEYS **Returns:**true if the first result is a ResultSet object; false if it is an update count or there are no results **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the second parameter supplied to this method is not Statement.RETURN\_GENERATED\_KEYS or Statement.NO\_GENERATED\_KEYS. [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method with a constant of Statement.RETURN\_GENERATED\_KEYS**Since:** 1.4 **See Also:**[getResultSet()](http://docs.google.com/java/sql/Statement.html#getResultSet()), [getUpdateCount()](http://docs.google.com/java/sql/Statement.html#getUpdateCount()), [getMoreResults()](http://docs.google.com/java/sql/Statement.html#getMoreResults()), [getGeneratedKeys()](http://docs.google.com/java/sql/Statement.html#getGeneratedKeys())

### execute

boolean **execute**([String](http://docs.google.com/java/lang/String.html) sql,  
 int[] columnIndexes)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Executes the given SQL statement, which may return multiple results, and signals the driver that the auto-generated keys indicated in the given array should be made available for retrieval. This array contains the indexes of the columns in the target table that contain the auto-generated keys that should be made available. The driver will ignore the array if the SQL statement is not an INSERT statement, or an SQL statement able to return auto-generated keys (the list of such statements is vendor-specific).

Under some (uncommon) situations, a single SQL statement may return multiple result sets and/or update counts. Normally you can ignore this unless you are (1) executing a stored procedure that you know may return multiple results or (2) you are dynamically executing an unknown SQL string.

The execute method executes an SQL statement and indicates the form of the first result. You must then use the methods getResultSet or getUpdateCount to retrieve the result, and getMoreResults to move to any subsequent result(s).

**Parameters:**sql - any SQL statementcolumnIndexes - an array of the indexes of the columns in the inserted row that should be made available for retrieval by a call to the method getGeneratedKeys **Returns:**true if the first result is a ResultSet object; false if it is an update count or there are no results **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the elements in the int array passed to this method are not valid column indexes [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**[getResultSet()](http://docs.google.com/java/sql/Statement.html#getResultSet()), [getUpdateCount()](http://docs.google.com/java/sql/Statement.html#getUpdateCount()), [getMoreResults()](http://docs.google.com/java/sql/Statement.html#getMoreResults())

### execute

boolean **execute**([String](http://docs.google.com/java/lang/String.html) sql,  
 [String](http://docs.google.com/java/lang/String.html)[] columnNames)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Executes the given SQL statement, which may return multiple results, and signals the driver that the auto-generated keys indicated in the given array should be made available for retrieval. This array contains the names of the columns in the target table that contain the auto-generated keys that should be made available. The driver will ignore the array if the SQL statement is not an INSERT statement, or an SQL statement able to return auto-generated keys (the list of such statements is vendor-specific).

In some (uncommon) situations, a single SQL statement may return multiple result sets and/or update counts. Normally you can ignore this unless you are (1) executing a stored procedure that you know may return multiple results or (2) you are dynamically executing an unknown SQL string.

The execute method executes an SQL statement and indicates the form of the first result. You must then use the methods getResultSet or getUpdateCount to retrieve the result, and getMoreResults to move to any subsequent result(s).

**Parameters:**sql - any SQL statementcolumnNames - an array of the names of the columns in the inserted row that should be made available for retrieval by a call to the method getGeneratedKeys **Returns:**true if the next result is a ResultSet object; false if it is an update count or there are no more results **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs, this method is called on a closed Statement or the elements of the String array passed to this method are not valid column names [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.4 **See Also:**[getResultSet()](http://docs.google.com/java/sql/Statement.html#getResultSet()), [getUpdateCount()](http://docs.google.com/java/sql/Statement.html#getUpdateCount()), [getMoreResults()](http://docs.google.com/java/sql/Statement.html#getMoreResults()), [getGeneratedKeys()](http://docs.google.com/java/sql/Statement.html#getGeneratedKeys())

### getResultSetHoldability

int **getResultSetHoldability**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves the result set holdability for ResultSet objects generated by this Statement object.

**Returns:**either ResultSet.HOLD\_CURSORS\_OVER\_COMMIT or ResultSet.CLOSE\_CURSORS\_AT\_COMMIT **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs or this method is called on a closed Statement**Since:** 1.4

### isClosed

boolean **isClosed**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Retrieves whether this Statement object has been closed. A Statement is closed if the method close has been called on it, or if it is automatically closed.

**Returns:**true if this Statement object is closed; false if it is still open **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if a database access error occurs**Since:** 1.6

### setPoolable

void **setPoolable**(boolean poolable)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Requests that a Statement be pooled or not pooled. The value specified is a hint to the statement pool implementation indicating whether the applicaiton wants the statement to be pooled. It is up to the statement pool manager as to whether the hint is used.

The poolable value of a statement is applicable to both internal statement caches implemented by the driver and external statement caches implemented by application servers and other applications.

By default, a Statement is not poolable when created, and a PreparedStatement and CallableStatement are poolable when created.

**Parameters:**poolable - requests that the statement be pooled if true and that the statement not be pooled if false

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if this method is called on a closed Statement

**Since:** 1.6

### isPoolable

boolean **isPoolable**()  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Returns a value indicating whether the Statement is poolable or not.

**Returns:**true if the Statement is poolable; false otherwise

**Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if this method is called on a closed Statement

**Since:** 1.6

**See Also:**[setPoolable(boolean)](http://docs.google.com/java/sql/Statement.html#setPoolable(boolean))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Statement.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/sql/SQLXML.html)   [**NEXT CLASS**](http://docs.google.com/java/sql/Struct.html) | [**FRAMES**](http://docs.google.com/index.html?java/sql/Statement.html)    [**NO FRAMES**](http://docs.google.com/Statement.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#3dy6vkm) | CONSTR | [METHOD](#35nkun2) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).