| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.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 PACKAGE**](http://docs.google.com/java/security/spec/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/java/text/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?java/sql/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

## Package java.sql

Provides the API for accessing and processing data stored in a data source (usually a relational database) using the JavaTM programming language.

**See:**

[**Description**](#3znysh7)

| **Interface Summary** | |
| --- | --- |
| [**Array**](http://docs.google.com/java/sql/Array.html) | The mapping in the Java programming language for the SQL type ARRAY. |
| [**Blob**](http://docs.google.com/java/sql/Blob.html) | The representation (mapping) in the JavaTM programming language of an SQL BLOB value. |
| [**CallableStatement**](http://docs.google.com/java/sql/CallableStatement.html) | The interface used to execute SQL stored procedures. |
| [**Clob**](http://docs.google.com/java/sql/Clob.html) | The mapping in the JavaTM programming language for the SQL CLOB type. |
| [**Connection**](http://docs.google.com/java/sql/Connection.html) | A connection (session) with a specific database. |
| [**DatabaseMetaData**](http://docs.google.com/java/sql/DatabaseMetaData.html) | Comprehensive information about the database as a whole. |
| [**Driver**](http://docs.google.com/java/sql/Driver.html) | The interface that every driver class must implement. |
| [**NClob**](http://docs.google.com/java/sql/NClob.html) | The mapping in the JavaTM programming language for the SQL NCLOB type. |
| [**ParameterMetaData**](http://docs.google.com/java/sql/ParameterMetaData.html) | An object that can be used to get information about the types and properties for each parameter marker in a PreparedStatement object. |
| [**PreparedStatement**](http://docs.google.com/java/sql/PreparedStatement.html) | An object that represents a precompiled SQL statement. |
| [**Ref**](http://docs.google.com/java/sql/Ref.html) | The mapping in the Java programming language of an SQL REF value, which is a reference to an SQL structured type value in the database. |
| [**ResultSet**](http://docs.google.com/java/sql/ResultSet.html) | A table of data representing a database result set, which is usually generated by executing a statement that queries the database. |
| [**ResultSetMetaData**](http://docs.google.com/java/sql/ResultSetMetaData.html) | An object that can be used to get information about the types and properties of the columns in a ResultSet object. |
| [**RowId**](http://docs.google.com/java/sql/RowId.html) | The representation (mapping) in the Java programming language of an SQL ROWID value. |
| [**Savepoint**](http://docs.google.com/java/sql/Savepoint.html) | The representation of a savepoint, which is a point within the current transaction that can be referenced from the Connection.rollback method. |
| [**SQLData**](http://docs.google.com/java/sql/SQLData.html) | The interface used for the custom mapping of an SQL user-defined type (UDT) to a class in the Java programming language. |
| [**SQLInput**](http://docs.google.com/java/sql/SQLInput.html) | An input stream that contains a stream of values representing an instance of an SQL structured type or an SQL distinct type. |
| [**SQLOutput**](http://docs.google.com/java/sql/SQLOutput.html) | The output stream for writing the attributes of a user-defined type back to the database. |
| [**SQLXML**](http://docs.google.com/java/sql/SQLXML.html) | The mapping in the JavaTM programming language for the SQL XML type. |
| [**Statement**](http://docs.google.com/java/sql/Statement.html) | The object used for executing a static SQL statement and returning the results it produces. |
| [**Struct**](http://docs.google.com/java/sql/Struct.html) | The standard mapping in the Java programming language for an SQL structured type. |
| [**Wrapper**](http://docs.google.com/java/sql/Wrapper.html) | Interface for JDBC classes which provide the ability to retrieve the delegate instance when the instance in question is in fact a proxy class. |

| **Class Summary** | |
| --- | --- |
| [**Date**](http://docs.google.com/java/sql/Date.html) | A thin wrapper around a millisecond value that allows JDBC to identify this as an SQL DATE value. |
| [**DriverManager**](http://docs.google.com/java/sql/DriverManager.html) | The basic service for managing a set of JDBC drivers.  **NOTE:** The DataSource interface, new in the JDBC 2.0 API, provides another way to connect to a data source. |
| [**DriverPropertyInfo**](http://docs.google.com/java/sql/DriverPropertyInfo.html) | Driver properties for making a connection. |
| [**SQLPermission**](http://docs.google.com/java/sql/SQLPermission.html) | The permission for which the SecurityManager will check when code that is running in an applet calls the DriverManager.setLogWriter method or the DriverManager.setLogStream (deprecated) method. |
| [**Time**](http://docs.google.com/java/sql/Time.html) | A thin wrapper around the java.util.Date class that allows the JDBC API to identify this as an SQL TIME value. |
| [**Timestamp**](http://docs.google.com/java/sql/Timestamp.html) | A thin wrapper around java.util.Date that allows the JDBC API to identify this as an SQL TIMESTAMP value. |
| [**Types**](http://docs.google.com/java/sql/Types.html) | The class that defines the constants that are used to identify generic SQL types, called JDBC types. |

| **Enum Summary** | |
| --- | --- |
| [**ClientInfoStatus**](http://docs.google.com/java/sql/ClientInfoStatus.html) | Enumeration for status of the reason that a property could not be set via a call to Connection.setClientInfo |
| [**RowIdLifetime**](http://docs.google.com/java/sql/RowIdLifetime.html) | Enumeration for RowId life-time values. |

| **Exception Summary** | |
| --- | --- |
| [**BatchUpdateException**](http://docs.google.com/java/sql/BatchUpdateException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) thrown when an error occurs during a batch update operation. |
| [**DataTruncation**](http://docs.google.com/java/sql/DataTruncation.html) | An exception thrown as a DataTruncation exception (on writes) or reported as a DataTruncation warning (on reads) when a data values is unexpectedly truncated for reasons other than its having execeeded MaxFieldSize. |
| [**SQLClientInfoException**](http://docs.google.com/java/sql/SQLClientInfoException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) is thrown when one or more client info properties could not be set on a Connection. |
| [**SQLDataException**](http://docs.google.com/java/sql/SQLDataException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) thrown when the SQLState class value is '*22*'. |
| [**SQLException**](http://docs.google.com/java/sql/SQLException.html) | An exception that provides information on a database access error or other errors. |
| [**SQLFeatureNotSupportedException**](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) thrown when the SQLState class value is '*0A*' ( the value is 'zero' A). |
| [**SQLIntegrityConstraintViolationException**](http://docs.google.com/java/sql/SQLIntegrityConstraintViolationException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) thrown when the SQLState class value is '*23*'. |
| [**SQLInvalidAuthorizationSpecException**](http://docs.google.com/java/sql/SQLInvalidAuthorizationSpecException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) thrown when the SQLState class value is '*28*'. |
| [**SQLNonTransientConnectionException**](http://docs.google.com/java/sql/SQLNonTransientConnectionException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) thrown for the SQLState class value '*08*', representing that the connection operation that failed will not succeed when the operation is retried without the cause of the failure being corrected. |
| [**SQLNonTransientException**](http://docs.google.com/java/sql/SQLNonTransientException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) thrown when an instance where a retry of the same operation would fail unless the cause of the SQLException is corrected. |
| [**SQLRecoverableException**](http://docs.google.com/java/sql/SQLRecoverableException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) thrown in situations where a previously failed operation might be able to succeed if the application performs some recovery steps and retries the entire transaction or in the case of a distributed transaction, the transaction branch. |
| [**SQLSyntaxErrorException**](http://docs.google.com/java/sql/SQLSyntaxErrorException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) thrown when the SQLState class value is '*42*'. |
| [**SQLTimeoutException**](http://docs.google.com/java/sql/SQLTimeoutException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) thrown when the timeout specified by Statement has expired. |
| [**SQLTransactionRollbackException**](http://docs.google.com/java/sql/SQLTransactionRollbackException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) thrown when the SQLState class value is '*40*'. |
| [**SQLTransientConnectionException**](http://docs.google.com/java/sql/SQLTransientConnectionException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) for the SQLState class value '*08*', representing that the connection operation that failed might be able to succeed when the operation is retried without any application-level changes. |
| [**SQLTransientException**](http://docs.google.com/java/sql/SQLTransientException.html) | The subclass of [SQLException](http://docs.google.com/java/sql/SQLException.html) is thrown in situations where a previoulsy failed operation might be able to succeed when the operation is retried without any intervention by application-level functionality. |
| [**SQLWarning**](http://docs.google.com/java/sql/SQLWarning.html) | An exception that provides information on database access warnings. |

## Package java.sql Description

Provides the API for accessing and processing data stored in a data source (usually a relational database) using the JavaTM programming language. This API includes a framework whereby different drivers can be installed dynamically to access different data sources. Although the JDBCTM API is mainly geared to passing SQL statements to a database, it provides for reading and writing data from any data source with a tabular format. The reader/writer facility, available through the javax.sql.RowSet group of interfaces, can be customized to use and update data from a spread sheet, flat file, or any other tabular data source.

## What the JDBC**TM** 4.0 API Includes

The JDBCTM 4.0 API includes both the java.sql package, referred to as the JDBC core API, and the javax.sql package, referred to as the JDBC Optional Package API. This complete JDBC API is included in the JavaTM Standard Edition (Java SETM), version 6. The javax.sql package extends the functionality of the JDBC API from a client-side API to a server-side API, and it is an essential part of the JavaTM Enterprise Edition (Java EETM) technology.

## Versions

The JDBC 4.0 API incorporates all of the previous JDBC API versions:

* The JDBC 3.0 API
* The JDBC 2.1 core API
* The JDBC 2.0 Optional Package API  
  (Note that the JDBC 2.1 core API and the JDBC 2.0 Optional Package API together are referred to as the JDBC 2.0 API.)
* The JDBC 1.2 API
* The JDBC 1.0 API

Classes, interfaces, methods, fields, constructors, and exceptions have the following "since" tags that indicate when they were introduced into the Java platform. When these "since" tags are used in JavadocTM comments for the JDBC API, they indicate the following:

* Since 1.6 -- new in the JDBC 4.0 API and part of the Java SE platform, version 6
* Since 1.4 -- new in the JDBC 3.0 API and part of the J2SE platform, version 1.4
* Since 1.2 -- new in the JDBC 2.0 API and part of the J2SE platform, version 1.2
* Since 1.1 or no "since" tag -- in the original JDBC 1.0 API and part of the JDKTM, version 1.1

**NOTE:** Many of the new features are optional; consequently, there is some variation in drivers and the features they support. Always check your driver's documentation to see whether it supports a feature before you try to use it.

**NOTE:** The class SQLPermission was added in the JavaTM 2 SDK, Standard Edition, version 1.3 release. This class is used to prevent unauthorized access to the logging stream associated with the DriverManager, which may contain information such as table names, column data, and so on.

## What the java.sql Package Contains

The java.sql package contains API for the following:

* Making a connection with a database via the DriverManager facility
  + DriverManager class -- makes a connection with a driver
  + SQLPermission class -- provides permission when code running within a Security Manager, such as an applet, attempts to set up a logging stream through the DriverManager
  + Driver interface -- provides the API for registering and connecting drivers based on JDBC technology ("JDBC drivers"); generally used only by the DriverManager class
  + DriverPropertyInfo class -- provides properties for a JDBC driver; not used by the general user
* Sending SQL statements to a database
  + Statement -- used to send basic SQL statements
  + PreparedStatement -- used to send prepared statements or basic SQL statements (derived from Statement)
  + CallableStatement -- used to call database stored procedures (derived from PreparedStatement)
  + Connection interface -- provides methods for creating statements and managing connections and their properties
  + Savepoint -- provides savepoints in a transaction
* Retrieving and updating the results of a query
  + ResultSet interface
* Standard mappings for SQL types to classes and interfaces in the Java programming language
  + Array interface -- mapping for SQL ARRAY
  + Blob interface -- mapping for SQL BLOB
  + Clob interface -- mapping for SQL CLOB
  + Date class -- mapping for SQL DATE
  + NClob interface -- mapping for SQL NCLOB
  + Ref interface -- mapping for SQL REF
  + RowId interface -- mapping for SQL ROWID
  + Struct interface -- mapping for SQL STRUCT
  + SQLXML interface -- mapping for SQL XML
  + Time class -- mapping for SQL TIME
  + Timestamp class -- mapping for SQL TIMESTAMP
  + Types class -- provides constants for SQL types
* Custom mapping an SQL user-defined type (UDT) to a class in the Java programming language
  + SQLData interface -- specifies the mapping of a UDT to an instance of this class
  + SQLInput interface -- provides methods for reading UDT attributes from a stream
  + SQLOutput interface -- provides methods for writing UDT attributes back to a stream
* Metadata
  + DatabaseMetaData interface -- provides information about the database
  + ResultSetMetaData interface -- provides information about the columns of a ResultSet object
  + ParameterMetaData interface -- provides information about the parameters to PreparedStatement commands
* Exceptions
  + SQLException -- thrown by most methods when there is a problem accessing data and by some methods for other reasons
  + SQLWarning -- thrown to indicate a warning
  + DataTruncation -- thrown to indicate that data may have been truncated
  + BatchUpdateException -- thrown to indicate that not all commands in a batch update executed successfully

### java.sql and javax.sql Features Introduced in the JDBC 4.0 API

* auto java.sql.Driver discovery -- no longer need to load a java.sql.Driver class via Class.forName
* National Character Set support added
* Support added for the SQL:2003 XML data type
* SQLException enhancements -- Added support for cause chaining; New SQLExceptions added for common SQLState class value codes
* Enhanced Blob/Clob functionality -- Support provided to create and free a Blob/Clob instance as well as additional methods added to improve accessiblity
* Support added for accessing a SQL ROWID
* Support added to allow a JDBC application to access an instance of a JDBC resource that has been wrapped by a vendor, usually in an application server or connection pooling environment.
* Availability to be notfied when a PreparedStatement that is associated with a PooledConnection has been closed or the driver determines is invalid

### java.sql and javax.sql Features Introduced in the JDBC 3.0 API

* Pooled statements -- reuse of statements associated with a pooled connection
* Savepoints -- allow a transaction to be rolled back to a designated savepoint
* Properties defined for ConnectionPoolDataSource -- specify how connections are to be pooled
* Metadata for parameters of a PreparedStatement object
* Ability to retrieve values from automatically generated columns
* Ability to have multiple ResultSet objects returned from CallableStatement objects open at the same time
* Ability to identify parameters to CallableStatement objects by name as well as by index
* ResultSet holdability -- ability to specify whether cursors should be held open or closed at the end of a transaction
* Ability to retrieve and update the SQL structured type instance that a Ref object references
* Ability to programmatically update BLOB, CLOB, ARRAY, and REF values.
* Addition of the java.sql.Types.DATALINK data type -- allows JDBC drivers access to objects stored outside a data source
* Addition of metadata for retrieving SQL type hierarchies

### java.sql Features Introduced in the JDBC 2.1 Core API

* Scrollable result sets--using new methods in the ResultSet interface that allow the cursor to be moved to a particular row or to a position relative to its current position
* Batch updates
* Programmatic updates--using ResultSet updater methods
* New data types--interfaces mapping the SQL3 data types
* Custom mapping of user-defined types (UDTs)
* Miscellaneous features, including performance hints, the use of character streams, full precision for java.math.BigDecimal values, additional security, and support for time zones in date, time, and timestamp values.

### javax.sql Features Introduced in the JDBC 2.0 Optional Package API

* The DataSource interface as a means of making a connection. The Java Naming and Directory InterfaceTM (JNDI) is used for registering a DataSource object with a naming service and also for retrieving it.
* Pooled connections -- allowing connections to be used and reused
* Distributed transactions -- allowing a transaction to span diverse DBMS servers
* RowSet technology -- providing a convenient means of handling and passing data

### Custom Mapping of UDTs

A user-defined type (UDT) defined in SQL can be mapped to a class in the Java programming language. An SQL structured type or an SQL DISTINCT type are the UDTs that may be custom mapped. The following three steps set up a custom mapping:

1. Defining the SQL structured type or DISTINCT type in SQL
2. Defining the class in the Java programming language to which the SQL UDT will be mapped. This class must implement the SQLData interface.
3. Making an entry in a Connection object's type map that contains two things:
   * the fully-qualified SQL name of the UDT
   * the Class object for the class that implements the SQLData interface

When these are in place for a UDT, calling the methods ResultSet.getObject or CallableStatement.getObject on that UDT will automatically retrieve the custom mapping for it. Also, the PreparedStatement.setObject method will automatically map the object back to its SQL type to store it in the data source.

## Package Specification

* [Specification of the JDBC 4.0 API](http://java.sun.com/products/jdbc/download.html)

## Related Documentation

* [Getting Started](http://docs.google.com/guide/jdbc/getstart/GettingStartedTOC.fm.html)--overviews of the major interfaces
* [Chapters on the JDBC API](http://java.sun.com/docs/books/tutorial/jdbc)--from the online version of *The Java Tutorial Continued*
* [*JDBCTMAPI Tutorial and Reference, Third Edition*](http://java.sun.com/docs/books/jdbc)-- a complete reference and tutorial for the JDBC 3.0 API

**Since:** 1.1

| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.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 PACKAGE**](http://docs.google.com/java/security/spec/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/java/text/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?java/sql/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

[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).