| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SQLData.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/SQLClientInfoException.html)   [**NEXT CLASS**](http://docs.google.com/java/sql/SQLDataException.html) | [**FRAMES**](http://docs.google.com/index.html?java/sql/SQLData.html)    [**NO FRAMES**](http://docs.google.com/SQLData.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

## **java.sql**

Interface SQLData

public interface **SQLData**

The interface used for the custom mapping of an SQL user-defined type (UDT) to a class in the Java programming language. The class object for a class implementing the SQLData interface will be entered in the appropriate Connection object's type map along with the SQL name of the UDT for which it is a custom mapping.

Typically, a SQLData implementation will define a field for each attribute of an SQL structured type or a single field for an SQL DISTINCT type. When the UDT is retrieved from a data source with the ResultSet.getObject method, it will be mapped as an instance of this class. A programmer can operate on this class instance just as on any other object in the Java programming language and then store any changes made to it by calling the PreparedStatement.setObject method, which will map it back to the SQL type.

It is expected that the implementation of the class for a custom mapping will be done by a tool. In a typical implementation, the programmer would simply supply the name of the SQL UDT, the name of the class to which it is being mapped, and the names of the fields to which each of the attributes of the UDT is to be mapped. The tool will use this information to implement the SQLData.readSQL and SQLData.writeSQL methods. The readSQL method calls the appropriate SQLInput methods to read each attribute from an SQLInput object, and the writeSQL method calls SQLOutput methods to write each attribute back to the data source via an SQLOutput object.

An application programmer will not normally call SQLData methods directly, and the SQLInput and SQLOutput methods are called internally by SQLData methods, not by application code.

**Since:** 1.2

| **Method Summary** | |
| --- | --- |
| [String](http://docs.google.com/java/lang/String.html) | [**getSQLTypeName**](http://docs.google.com/java/sql/SQLData.html#getSQLTypeName())()            Returns the fully-qualified name of the SQL user-defined type that this object represents. |
| void | [**readSQL**](http://docs.google.com/java/sql/SQLData.html#readSQL(java.sql.SQLInput,%20java.lang.String))([SQLInput](http://docs.google.com/java/sql/SQLInput.html) stream, [String](http://docs.google.com/java/lang/String.html) typeName)            Populates this object with data read from the database. |
| void | [**writeSQL**](http://docs.google.com/java/sql/SQLData.html#writeSQL(java.sql.SQLOutput))([SQLOutput](http://docs.google.com/java/sql/SQLOutput.html) stream)            Writes this object to the given SQL data stream, converting it back to its SQL value in the data source. |

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

### getSQLTypeName

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

Returns the fully-qualified name of the SQL user-defined type that this object represents. This method is called by the JDBC driver to get the name of the UDT instance that is being mapped to this instance of SQLData.

**Returns:**the type name that was passed to the method readSQL when this object was constructed and populated **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if there is a database access error [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.2

### readSQL

void **readSQL**([SQLInput](http://docs.google.com/java/sql/SQLInput.html) stream,  
 [String](http://docs.google.com/java/lang/String.html) typeName)  
 throws [SQLException](http://docs.google.com/java/sql/SQLException.html)

Populates this object with data read from the database. The implementation of the method must follow this protocol:

* It must read each of the attributes or elements of the SQL type from the given input stream. This is done by calling a method of the input stream to read each item, in the order that they appear in the SQL definition of the type.
* The method readSQL then assigns the data to appropriate fields or elements (of this or other objects). Specifically, it must call the appropriate *reader* method (SQLInput.readString, SQLInput.readBigDecimal, and so on) method(s) to do the following: for a distinct type, read its single data element; for a structured type, read a value for each attribute of the SQL type.

The JDBC driver initializes the input stream with a type map before calling this method, which is used by the appropriate SQLInput reader method on the stream.

**Parameters:**stream - the SQLInput object from which to read the data for the value that is being custom mappedtypeName - the SQL type name of the value on the data stream **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if there is a database access error [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.2 **See Also:**[SQLInput](http://docs.google.com/java/sql/SQLInput.html)

### writeSQL

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

Writes this object to the given SQL data stream, converting it back to its SQL value in the data source. The implementation of the method must follow this protocol:

It must write each of the attributes of the SQL type to the given output stream. This is done by calling a method of the output stream to write each item, in the order that they appear in the SQL definition of the type. Specifically, it must call the appropriate SQLOutput writer method(s) (writeInt, writeString, and so on) to do the following: for a Distinct Type, write its single data element; for a Structured Type, write a value for each attribute of the SQL type.

**Parameters:**stream - the SQLOutput object to which to write the data for the value that was custom mapped **Throws:** [SQLException](http://docs.google.com/java/sql/SQLException.html) - if there is a database access error [SQLFeatureNotSupportedException](http://docs.google.com/java/sql/SQLFeatureNotSupportedException.html) - if the JDBC driver does not support this method**Since:** 1.2 **See Also:**[SQLOutput](http://docs.google.com/java/sql/SQLOutput.html)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SQLData.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/SQLClientInfoException.html)   [**NEXT CLASS**](http://docs.google.com/java/sql/SQLDataException.html) | [**FRAMES**](http://docs.google.com/index.html?java/sql/SQLData.html)    [**NO FRAMES**](http://docs.google.com/SQLData.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

[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).