| | [**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/javax/sql/rowset/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/javax/sql/rowset/spi/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sql/rowset/serial/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

## Package javax.sql.rowset.serial

Provides utility classes to allow serializable mappings between SQL types and data types in the Java programming language.

**See:**

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

| **Class Summary** | |
| --- | --- |
| [**SerialArray**](http://docs.google.com/javax/sql/rowset/serial/SerialArray.html) | A serialized version of an Array object, which is the mapping in the Java programming language of an SQL ARRAY value. |
| [**SerialBlob**](http://docs.google.com/javax/sql/rowset/serial/SerialBlob.html) | A serialized mapping in the Java programming language of an SQL BLOB value. |
| [**SerialClob**](http://docs.google.com/javax/sql/rowset/serial/SerialClob.html) | A serialized mapping in the Java programming language of an SQL CLOB value. |
| [**SerialDatalink**](http://docs.google.com/javax/sql/rowset/serial/SerialDatalink.html) | A serialized mapping in the Java programming language of an SQL DATALINK value. |
| [**SerialJavaObject**](http://docs.google.com/javax/sql/rowset/serial/SerialJavaObject.html) | A serializable mapping in the Java programming language of an SQL JAVA\_OBJECT value. |
| [**SerialRef**](http://docs.google.com/javax/sql/rowset/serial/SerialRef.html) | A serialized mapping of a Ref object, which is the mapping in the Java programming language of an SQL REF value. |
| [**SerialStruct**](http://docs.google.com/javax/sql/rowset/serial/SerialStruct.html) | A serialized mapping in the Java programming language of an SQL structured type. |
| [**SQLInputImpl**](http://docs.google.com/javax/sql/rowset/serial/SQLInputImpl.html) | An input stream used for custom mapping user-defined types (UDTs). |
| [**SQLOutputImpl**](http://docs.google.com/javax/sql/rowset/serial/SQLOutputImpl.html) | The output stream for writing the attributes of a custom-mapped user-defined type (UDT) back to the database. |

| **Exception Summary** | |
| --- | --- |
| [**SerialException**](http://docs.google.com/javax/sql/rowset/serial/SerialException.html) | Indicates and an error with the serialization or de-serialization of SQL types such as BLOB, CLOB, STRUCT or ARRAY in addition to SQL types such as DATALINK and JAVAOBJECT |

## Package javax.sql.rowset.serial Description

Provides utility classes to allow serializable mappings between SQL types and data types in the Java programming language.

Standard JDBC RowSet implementations may use these utility classes to assist in the serialization of disconnected RowSet objects. This is useful when transmitting a disconnected RowSet object over the wire to a different VM or across layers within an application.

### 1.0 SerialArray

A serializable mapping in the Java programming language of an SQL ARRAY value.

The SerialArray class provides a constructor for creating a SerialArray instance from an Array object, methods for getting the base type and the SQL name for the base type, and methods for copying all or part of a SerialArray object.

### 2.0 SerialBlob

A serializable mapping in the Java programming language of an SQL BLOB value.

The SerialBlob class provides a constructor for creating an instance from a Blob object. Note that the Blob object should have brought the SQL BLOB value's data over to the client before a SerialBlob object is constructed from it. The data of an SQL BLOB value can be materialized on the client as an array of bytes (using the method Blob.getBytes) or as a stream of uninterpreted bytes (using the method Blob.getBinaryStream).

SerialBlob methods make it possible to make a copy of a SerialBlob object as an array of bytes or as a stream. They also make it possible to locate a given pattern of bytes or a Blob object within a SerialBlob object.

### 3.0 SerialClob

A serializable mapping in the Java programming language of an SQL CLOB value.

The SerialClob class provides a constructor for creating an instance from a Clob object. Note that the Clob object should have brought the SQL CLOB value's data over to the client before a SerialClob object is constructed from it. The data of an SQL CLOB value can be materialized on the client as a stream of Unicode characters.

SerialClob methods make it possible to get a substring from a SerialClob object or to locate the start of a pattern of characters.

### 5.0 SerialDatalink

A serializable mapping in the Java programming language of an SQL DATALINK value. A DATALINK value references a file outside of the underlying data source that the the originating data source manages.

RowSet implementations can use the method RowSet.getURL() to retrieve a java.net.URL object, which can be used to manipulate the external data.

      java.net.URL url = rowset.getURL(1);

### 6.0 SerialJavaObject

A serializable mapping in the Java programming language of an SQL JAVA\_OBJECT value. Assuming the Java object instance implements the Serializable interface, this simply wraps the serialization process.

If however, the serialization is not possible in the case where the Java object is not immediately serializable, this class will attempt to serialize all non static members to permit the object instance state to be serialized. Static or transient fields cannot be serialized and attempting to do so will result in a SerialException being thrown.

### 7.0 SerialRef

A serializable mapping between the SQL REF type and the Java programming language.

The SerialRef class provides a constructor for creating a SerialRef instance from a Ref type and provides methods for getting and setting the Ref object type.

### 8.0 SerialStruct

A serializable mapping in the Java programming language of an SQL structured type. Each attribute that is not already serializable is mapped to a serializable form, and if an attribute is itself a structured type, each of its attributes that is not already serializable is mapped to a serializable form.

In addition, if a Map object is passed to one of the constructors or to the method getAttributes, the structured type is custom mapped according to the mapping specified in the Map object.

The SerialStruct class provides a constructor for creating an instance from a Struct object, a method for retrieving the SQL type name of the SQL structured type in the database, and methods for retrieving its attribute values.

### 9.0 SQLInputImpl

An input stream used for custom mapping user-defined types (UDTs). An SQLInputImpl object is an input stream that contains a stream of values that are the attributes of a UDT. This class is used by the driver behind the scenes when the method getObject is called on an SQL structured or distinct type that has a custom mapping; a programmer never invokes SQLInputImpl methods directly.

The SQLInputImpl class provides a set of reader methods analogous to the ResultSet getter methods. These methods make it possible to read the values in an SQLInputImpl object. The method wasNull is used to determine whether the the last value read was SQL NULL.

When a constructor or getter method that takes a Map object is called, the JDBC driver calls the method SQLData.getSQLType to determine the SQL type of the UDT being custom mapped. The driver creates an instance of SQLInputImpl, populating it with the attributes of the UDT. The driver then passes the input stream to the method SQLData.readSQL, which in turn calls the SQLInputImpl methods to read the attributes from the input stream.

### 10.0 SQLOutputImpl

The output stream for writing the attributes of a custom mapped user-defined type (UDT) back to the database. The driver uses this interface internally, and its methods are never directly invoked by an application programmer.

When an application calls the method PreparedStatement.setObject, the driver checks to see whether the value to be written is a UDT with a custom mapping. If it is, there will be an entry in a type map containing the Class object for the class that implements SQLData for this UDT. If the value to be written is an instance of SQLData, the driver will create an instance of SQLOutputImpl and pass it to the method SQLData.writeSQL. The method writeSQL in turn calls the appropriate SQLOutputImpl writer methods to write data from the SQLData object to the SQLOutputImpl output stream as the representation of an SQL user-defined type.

### Custom Mapping

The JDBC API provides mechanisms for mapping an SQL structured type or DISTINCT type to the Java programming language. Typically, a structured type is mapped to a class, and its attributes are mapped to fields in the class. (A DISTINCT type can thought of as having one attribute.) However, there are many other possibilities, and there may be any number of different mappings.

A programmer defines the mapping by implementing the interface SQLData. For example, if an SQL structured type named AUTHORS has the attributes NAME, TITLE, and PUBLISHER, it could be mapped to a Java class named Authors. The Authors class could have the fields name, title, and publisher, to which the attributes of AUTHORS are mapped. In such a case, the implementation of SQLData could look like the following:

public class Authors implements SQLData {  
 public String name;  
 public String title;  
 public String publisher;  
  
 private String sql\_type;  
 public String getSQLTypeName() {  
 return sql\_type;  
 }  
  
 public void readSQL(SQLInput stream, String type)  
 throws SQLException {  
 sql\_type = type;  
 name = stream.readString();  
 title = stream.readString();  
 publisher = stream.readString();  
 }  
  
 public void writeSQL(SQLOutput stream) throws SQLException {  
 stream.writeString(name);  
 stream.writeString(title);  
 stream.writeString(publisher);  
 }  
 }

A java.util.Map object is used to associate the SQL structured type with its mapping to the class Authors. The following code fragment shows how a Map object might be created and given an entry associating AUTHORS and Authors.

java.util.Map map = new java.util.HashMap();  
 map.put("SCHEMA\_NAME.AUTHORS", Class.forName("Authors");

The Map object *map* now contains an entry with the fully qualified name of the SQL structured type and the Class object for the class Authors. It can be passed to a method to tell the driver how to map AUTHORS to Authors.

For a disconnected RowSet object, custom mapping can be done only when a Map object is passed to the method or constructor that will be doing the custom mapping. The situation is different for connected RowSet objects because they maintain a connection with the data source. A method that does custom mapping and is called by a disconnected RowSet object may use the Map object that is associated with the Connection object being used. So, in other words, if no map is specified, the connection's type map can be used by default.

| | [**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/javax/sql/rowset/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/javax/sql/rowset/spi/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sql/rowset/serial/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).