9 Oracle Web RowSet

In the previous two chapters we mapped an XML document to a relational database using XSU and a relational database table to an XML document using XSU and XSQL. In this chapter we will use the XML document representation of a result set generated with an SQL query to modify a relational database table.

The `ResultSet` interface requires a persistent connection with a database to invoke the insert, update, and delete row operations on the database table data. The `RowSet` interface extends the `ResultSet` interface and is a container for tabular data that may operate without being connected to the data source. Thus, the `RowSet` interface reduces the overhead of a persistent connection with the database.

In J2SE 5.0, five new implementations of RowSet—JdbcRowSet, CachedRowSet, WebRowSet, FilteredRowSet, and JoinRowSet—were introduced. The `WebRowSet` interface extends the `RowSet` interface and is the XML document representation of a `RowSet` object. A `WebRowSet` object represents a set of fetched database table rows, which may be modified without being connected to the database.

Support for Oracle Web RowSet is a new feature in Oracle Database 10g driver. Oracle Web RowSet precludes the requirement for a persistent connection with the database. A connection is required only for retrieving data from the database with a `SELECT` query and for updating data in the database after all the required row operations on the retrieved data have been performed. Oracle Web RowSet is used for queries and modifications on the data retrieved from the database. Oracle Web RowSet, as an XML document representation of a `RowSet` facilitates the transfer of data.

In Oracle Database 10g and 11g JDBC drivers, Oracle Web RowSet is implemented in the `oracle.jdbc.rowset` package. The `OracleWebRowSet` class represents a Oracle Web RowSet. The data in the Web RowSet may be modified without connecting to the database. The database table may be updated with the `OracleWebRowSet` class after the modifications to the Web RowSet have been made. A database JDBC connection is required only for retrieving data from the database and for updating the database. An XML document representation of the data in a Web RowSet may be obtained for data exchange. In this chapter the Web RowSet feature in Oracle 10g database JDBC driver is implemented in JDeveloper 10g. An example Web RowSet will be created from a database. The Web RowSet will be modified and stored in the database table.

In this chapter we will learn the following:

- Creating a Oracle Web RowSet object
- Adding a row to Oracle Web RowSet
- Reading a row from Oracle Web RowSet
- Updating a row in Oracle Web RowSet
- Deleting a row from Oracle Web RowSet
- Updating Database Table with modified Oracle Web RowSet

Setting the Environment

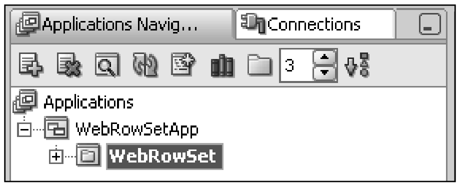
We will use Oracle database to generate an updatable `OracleWebRowSet` object. Therefore, install Oracle database 10g including the sample schemas. Connect to the database with the OE schema:

```
SQL> CONNECT OE/<password>
```

Create an example database table, `Catalog`, with the following SQL script:

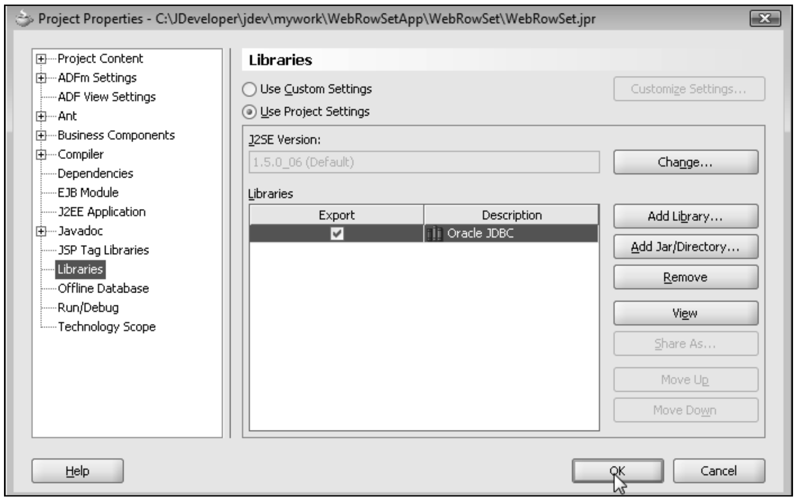
```
CREATE TABLE OE.Catalog(Journal VARCHAR(25), Publisher Varchar(25),
  Edition VARCHAR(25), Title Varchar(45), Author Varchar(25));
INSERT INTO OE.Catalog VALUES('Oracle Magazine',  'Oracle
  Publishing', 'July-August 2005', 'Tuning Undo Tablespace',
  'Kimberly Floss');
INSERT INTO OE.Catalog VALUES('Oracle Magazine',    'Oracle
  Publishing', 'March-April 2005', 'Starting with Oracle ADF', 'Steve
  Muench');
```

Configure JDeveloper 10g for Web RowSet implementation. Create a project in JDeveloper. Select **File | New | General | Application**. In the **Create Application** window specify an **Application Name** and click on **Next**. In the **Create Project** window specify a **Project Name** and click on **Next**. A project is added in the **Applications Navigator**.



Next, we will set the project libraries. Select **Tools | Project Properties** and in the **Project Properties** window select **Libraries | Add Library** to add a library. Add the **Oracle JDBC** library to project libraries. If the Oracle JDBC drivers version prior to the Oracle database 10g (R2) JDBC drivers version is used, create a library from the Oracle Web RowSet implementation classes JAR file, `C:\JDeveloper10.1.3\jdbc\lib\ocrs12.jar`. The `ocrs12.jar` is required only for JDBC drivers prior to Oracle database 10g (R2) JDBC drivers. In Oracle database 10g (R2) JDBC drivers Oracle RowSet implementation classes are packaged in the `ojdbc14.jar`. In Oracle database 11g JDBC drivers Oracle RowSet implementation classes are packaged in `ojdbc5.jar` and `ojdbc6.jar`.

In the **Add Library** window select the **User** node and click on **New**. In the **Create Library** window specify a **Library Name**, select the **Class Path** node and click on **Add Entry**. Add an entry for `ocrs12.jar`. As Web RowSet was introduced in J2SE 5.0, if J2SE 1.4 is being used we also need to add an entry for the RowSet implementations JAR file, `rowset.jar`. Download the JDBC RowSet Implementations 1.0.1 zip file, `jdbc_rowset_tiger-1_0_1-mrel-ri.zip`, from `http://java.sun.com/products/jdbc/download.html#rowset1_0_1` and extract the JDBC RowSet zip file to a directory. Click on **OK** in the **Create Library** window. Click on **OK** in the **Add Library** window. A library for the Web RowSet application is added.


Project Properties - C:\JDeveloper\jdev\mywork\WebRowSetApp\WebRowSet\WebRowSet.jpr

Libraries

Use Custom Settings
Use Project Settings (selected)

J2SE Version: 1.5.0_06 (Default)

Libraries

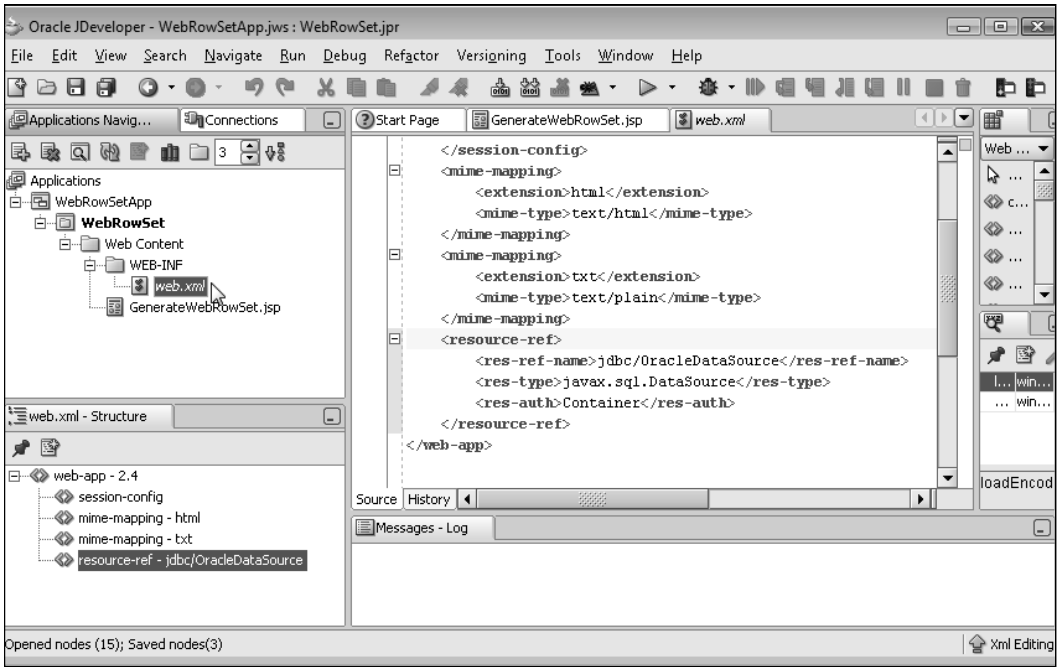
Export	Description
✓	Oracle JDBC

Now configure an OC4J data source. Select **Tools | Embedded OC4J Server Preferences**. A data source may be configured globally or for the current workspace. If a global data source is created using **Global | Data Sources,** the data source is configured in the `C:\JDeveloper10.1.3\jdev\system\oracle.j2ee.10.1.3.36.73\embedded-oc4j\config \data-sources.xml` file. If a data source is configured for the current workspace using **Current Workspace | Data Sources**, the data source is configured in the `data-sources.xml` file. For example, the data source file for the `WebRowSetApp` application is `WebRowSetApp-data-sources.xml`. In the **Embedded OC4J Server Preferences** window configure either a global data source or a data source in the current workspace with the procedure discussed in Chapter 2. A global data source definition is available to all applications deployed in the OC4J server instance. A `managed-data-source` element is added to the `data-sources.xml` file.

```
<managed-data-source name='OracleDataSource' connection-pool-
  name='Oracle Connection Pool' jndi-name='jdbc/OracleDataSource'/>
<connection-pool name='Oracle Connection Pool'>
 <connection-factory factory-
    class='oracle.jdbc.pool.OracleDataSource' user='OE' password='pw'
    url="jdbc:oracle:thin:@localhost:1521:ORCL">
 </connection-factory>
</connection-pool>
```

Add a JSP, `GenerateWebRowSet.jsp`, to the `WebRowSet` project. Select **File | New | Web Tier | JSP | JSP**. Click on **OK**. Select `J2EE 1.3 or J2EE 1.4` in the Web Application window and click on **Next**. In the **JSP File** window specify a **File Name** and click on **Next**. Select the default settings in the **Error Page Options** page and click on **Next**. Select the default settings in the **Tag Libraries** window and click on **Next**. Select the default options in the **HTML Options** window and click on **Next**. Click on **Finish** in the **Finish** window. Next, configure the `web.xml` deployment descriptor to include a reference to the data source resource configured in the `data-sources.xml` file as shown in following listing:

```
<resource-ref>
 <res-ref-name>jdbc/OracleDataSource</res-ref-name>
 <res-type>javax.sql.DataSource</res-type>
 <res-auth>Container</res-auth>
</resource-ref>
```


Oracle JDeveloper - WebRowSetApp.jws : WebRowSet.jpr
File Edit View Search Navigate Run Debug Refactor Versioning Tools Window Help
Applications Navig... Connections
Start Page GenerateWebRowSet.jsp web.xml
Applications
WebRowSetApp
WebRowSet
Web Content
WEB-INF
web.xml
GenerateWebRowSet.jsp
web.xml - Structure
web-app - 2.4
session-config
mime-mapping - html
mime-mapping - txt
resource-ref - jdbc/OracleDataSource
Source History
Messages - Log
Xml Editing

```
    </session-config>
    <mime-mapping>
        <extension>html</extension>
        <mime-type>text/html</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>txt</extension>
        <mime-type>text/plain</mime-type>
    </mime-mapping>
    <resource-ref>
        <res-ref-name>jdbc/OracleDataSource</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
    </resource-ref>
</web-app>
```

Opened nodes (15); Saved nodes(3)

Creating a Web RowSet

In this section we will create a Web RowSet from a database table and an XML document representation of the Web RowSet is generated. Create a Java class in JDeveloper with **File | New | General | Java Class**. In the **Create Java Class** window specify the class name, `WebRowSetQuery`, and package name and click on **OK**. A Java class, `WebRowSetQuery.java` gets added to the `WebRowSet` project. In the Java application first import the `oracle.jdbc.rowset` package classes. Create an `OracleWebRowSet` class object:

```
OracleWebRowSet webRowSet=new OracleWebRowSet();
```

Set the data source name to obtain a JDBC connection with the database. The data source name is configured in the `data-sources.xml` file:

```
webRowSet.setDataSourceName("jdbc/OracleDataSource");
```

Set the SQL query command for the `OracleWebRowSet` class object:

```
webRowSet.setCommand(selectQuery);
```

Variable `selectQuery` is the `String` value for the SQL statement that is to be used to query the database. SQL statement value is obtained from an input field in a JSP. Set the username and password to obtain a JDBC connection:

```
webRowSet.setUsername("OE");
webRowSet.setPassword("<password>");
```

Set the read only, fetch size, and max rows attributes of the `OracleWebRowSet` object:

```
webRowSet.setReadOnly(false);
webRowSet.setFetchSize(5);
webRowSet.setMaxRows(3);
```

Run the SQL command specified in the `setCommand()` method with the `execute()` method:

```
webRowSet.execute();
```

A Web RowSet is created consisting of the data retrieved from the database table with the SQL query. Generate an XML document from the Web RowSet using the `writeXml()` method;

```
OutputStreamWriter output=new OutputStreamWriter( new
FileOutputStream(new File("c:/output/output.xml")));
webRowSet.writeXml(output);
```

Oracle Web RowSet also provides `readXml()` methods to read an Oracle Web RowSet object in XML format using a Reader object or an InputStream object. If the `readXml()` methods are to be used set one of the following JAXP system properties:

- `javax.xml.parsers.SAXParserFactory`
- `javax.xml.parsers.DocumentBuilderFactory`

For example, set the `SAXParserFactory` property as follows:

```
System.setProperty("javax.xml.parsers.SAXParserFactory",
       "oracle.xml.jaxp.JXSAXParserFactory");
```

`WebRowSetQuery.java` also has methods to read, update, delete, and insert a row in the database table, which will be discussed in the subsequent sections. `WebRowSetQuery.java` application is listed below:

```
package webrowset;
import oracle.jdbc.rowset.*;
import java.io.*;
import java.sql.SQLException;
public class WebRowSetQuery
{
```

```
public OracleWebRowSet webRowSet;
public String selectQuery;
public WebRowSetQuery()
{
}
public WebRowSetQuery(OracleWebRowSet webRowSet)
{
 this.webRowSet = webRowSet;
}
public void generateWebRowSet(String selectQuery)
{
 try
 {
  webRowSet = new OracleWebRowSet();
  webRowSet.setDataSourceName("jdbc/OracleDataSource");
  webRowSet.setCommand(selectQuery);
  webRowSet.setUsername("oe");
  webRowSet.setPassword("pw");
  webRowSet.setReadOnly(false);
  webRowSet.setFetchSize(5);
  webRowSet.setMaxRows(3);
  webRowSet.execute();
 }
 catch (SQLException e)
 {
  System.out.println(e.getMessage());
 }
}
public void generateXMLDocument()
{
 try
 {
  OutputStreamWriter output = new OutputStreamWriter(
            new FileOutputStream(new File("c:/output/output.xml")));
  webRowSet.writeXml(output);
 }
 catch (SQLException e)
 {
  System.out.println(e.getMessage());
 }
 catch (IOException e)
 {
 }
}
public void deleteRow(int row)
{
 try
 {
  webRowSet.absolute(row);
  webRowSet.deleteRow();
```

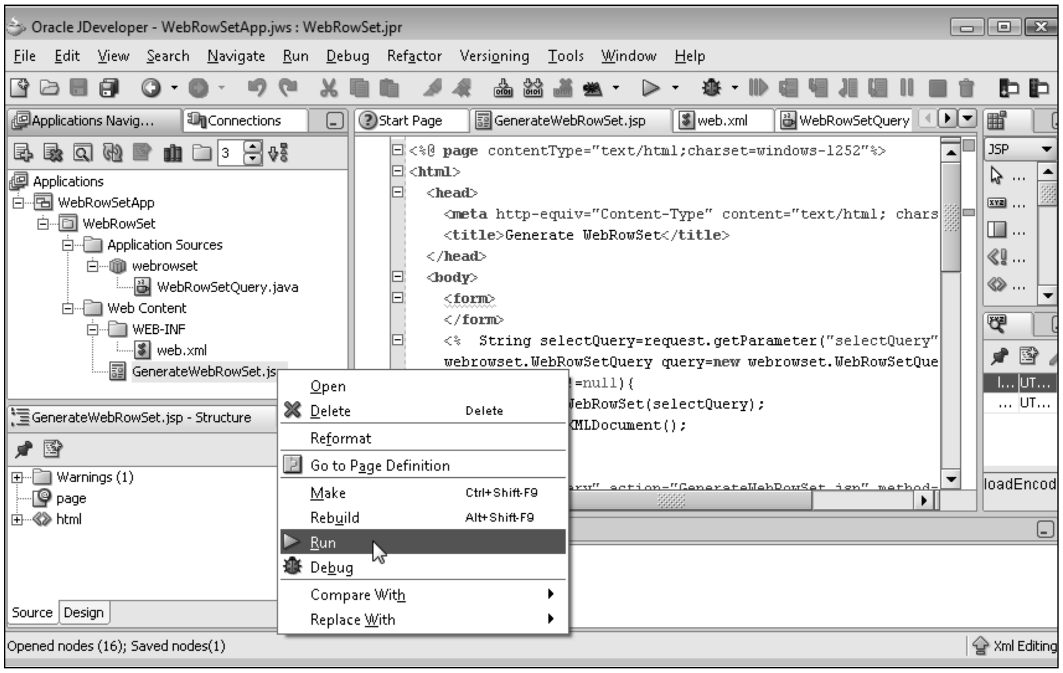
```
     }
     catch (SQLException e)
     {
      System.out.println(e.getMessage());
     }
    }
    public void insertRow(String journal, String publisher,
              String edition, String title, String author)
    {
     try
     {
      webRowSet.moveToInsertRow();
      webRowSet.updateString(1, journal);
      webRowSet.updateString(2, publisher);
      webRowSet.updateString(3, edition);
      webRowSet.updateString(4, title);
      webRowSet.updateString(5, author);
      webRowSet.insertRow();
     }
     catch (SQLException e)
     {
      System.out.println(e.getMessage());
     }
    }
    public void updateRow(int rowUpdate, String journal,
          String publisher, String edition, String title, String author)
    {
     try
     {
      webRowSet.absolute(rowUpdate);
      webRowSet.updateString(1, journal);
      webRowSet.updateString(2, publisher);
      webRowSet.updateString(3, edition);
      webRowSet.updateString(4, title);
      webRowSet.updateString(5, author);
      webRowSet.updateRow();
     }
     catch (SQLException e)
     {
      System.out.println(e.getMessage());
     }
    }
    public String[] readRow(int rowRead)
    {
     String[] resultSet = null;
     try
     {
      resultSet = new String[5];
      webRowSet.absolute(rowRead);
      resultSet[0] = webRowSet.getString(1);
      resultSet[1] = webRowSet.getString(2);
      resultSet[2] = webRowSet.getString(3);
      resultSet[3] = webRowSet.getString(4);
      resultSet[4] = webRowSet.getString(5);
     }
     catch (SQLException e)
     {
      System.out.println(e.getMessage());
     }
     return resultSet;
    }
   public void updateDatabase()
    {
     try
     {
      webRowSet.acceptChanges();
     }
     catch (java.sql.SQLException e)
     {
      System.out.println(e.getMessage());
     }
    }
   }
```

The SELECT query with which the Web RowSet is created is input from the GenerateWebRowSet.jsp JSP, which was added in the *Setting the Environment* section, and is listed below:

```
<%@ page contentType="text/html;charset=windows-1252"%>
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html;
charset=windows-1252">
  <title>Generate WebRowSet</title>
 </head>
 <body>
  <form>
  </form>
  <%
  String selectQuery=request.getParameter("selectQuery");
  webrowset.WebRowSetQuery query=new webrowset.WebRowSetQuery();
  if(selectQuery!=null)
  {
   query.generateWebRowSet(selectQuery);
   query.generateXMLDocument();
  }
  %>
    <form name="query" action="GenerateWebRowSet.jsp" method="post">
      <table>
        <tr>
```

```
            <td>Select Query:</td>
          </tr><tr><td>
              <textarea name="selectQuery" rows="5"
                 cols="50"></textarea>
            </td>
          </tr><tr><td>
              <input class="Submit" type="submit" value="Apply"/>
            </td>
          </tr>
        </table>
      </form>
    </body>
  </html>
```

Right-click on the `GenerateWebRowSet.jsp` and select **Run** to run the JSP.

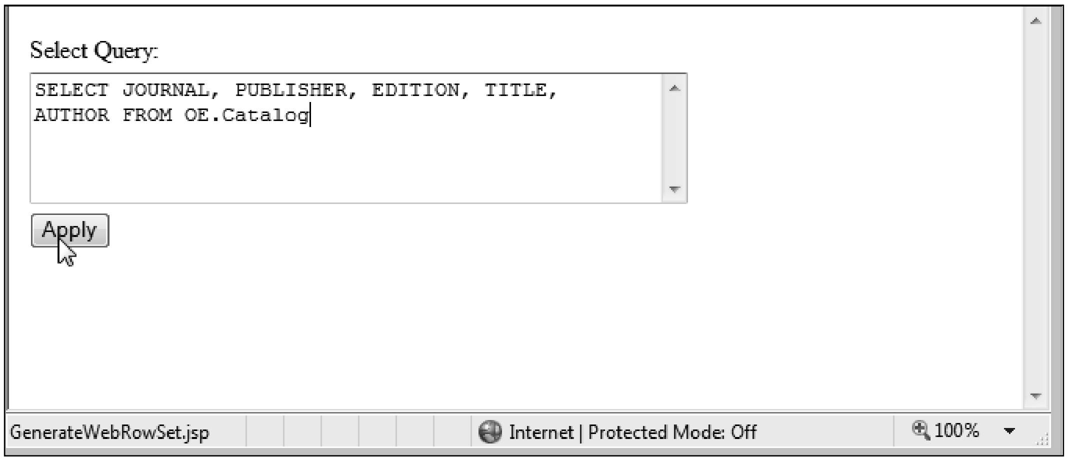


```
<%@ page contentType="text/html;charset=windows-1252"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; chars
    <title>Generate WebRowSet</title>
  </head>
  <body>
    <form>
    </form>
    <%  String selectQuery=request.getParameter("selectQuery"
    webrowset.WebRowSetQuery query=new webrowset.WebRowSetQue
    !=null){
    WebRowSet(selectQuery);
    XMLDocument();
```

In the JSP page displayed, specify the SQL query from which a Web RowSet is to be generated. For example, specify SQL Query:

`SELECT JOURNAL, PUBLISHER, EDITION, TITLE, AUTHOR FROM OE.Catalog`

Click on **Apply**.



```
SELECT JOURNAL, PUBLISHER, EDITION, TITLE,
AUTHOR FROM OE.Catalog
```

A Web RowSet is generated and an XML document is generated from the Web RowSet. The XML document output from the Web RowSet includes the metadata information for the JDBC data source, the database table, and the data in the table; the `data` element tag represents the data in the database table. An XML document generated from a Web RowSet is based on the DTD (`http://java.sun.com/j2ee/dtds/RowSet.dtd`). The XML document generated from the example database table `Catalog` as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RowSet PUBLIC '-//Sun Microsystems, Inc.//DTD RowSet//EN'
    'http://java.sun.com/j2ee/dtds/RowSet.dtd'>
<RowSet>
  <properties>
    <command>SELECT JOURNAL, PUBLISHER, EDITION, TITLE, AUTHOR FROM
         OE.Catalog</command>
    <concurrency>1007</concurrency>
    <datasource>jdbc/OracleDataSource</datasource>
    <escape-processing>true</escape-processing>
    <fetch-direction>1002</fetch-direction>
    <fetch-size>10</fetch-size>
    <isolation-level>2</isolation-level>
    <key-columns>
    </key-columns>
    <map></map>
    <max-field-size>0</max-field-size>
    <max-rows>3</max-rows>
    <query-timeout>0</query-timeout>
    <read-only>false</read-only>
    <rowset-type>1005</rowset-type>
    <show-deleted>false</show-deleted>
    <url>jdbc:oracle:thin:@localhost:1521:ORCL</url>
  </properties>
  <metadata>
    <column-count>5</column-count>
    <column-definition>
      <column-index>1</column-index>
      <auto-increment>false</auto-increment>
      <case-sensitive>true</case-sensitive>
      <currency>false</currency>
      <nullable>1</nullable>
      <signed>true</signed>
      <searchable>true</searchable>
      <column-display-size>25</column-display-size>
      <column-label>JOURNAL</column-label>
      <column-name>JOURNAL</column-name>
      <schema-name></schema-name>
      <column-precision>0</column-precision>
      <column-scale>0</column-scale>
      <table-name></table-name>
      <catalog-name></catalog-name>
      <column-type>12</column-type>
      <column-type-name>VARCHAR2</column-type-name>
    </column-definition>
    <column-definition>
      <column-index>2</column-index>
      <auto-increment>false</auto-increment>
      <case-sensitive>true</case-sensitive>
      <currency>false</currency>
      <nullable>1</nullable>
      <signed>true</signed>
      <searchable>true</searchable>
      <column-display-size>25</column-display-size>
      <column-label>PUBLISHER</column-label>
      <column-name>PUBLISHER</column-name>
      <schema-name></schema-name>
      <column-precision>0</column-precision>
      <column-scale>0</column-scale>
      <table-name></table-name>
      <catalog-name></catalog-name>
      <column-type>12</column-type>
      <column-type-name>VARCHAR2</column-type-name>
```

```
</column-definition>
<column-definition>
  <column-index>3</column-index>
  <auto-increment>false</auto-increment>
  <case-sensitive>true</case-sensitive>
  <currency>false</currency>
  <nullable>1</nullable>
  <signed>true</signed>
  <searchable>true</searchable>
  <column-display-size>25</column-display-size>
  <column-label>EDITION</column-label>
  <column-name>EDITION</column-name>
  <schema-name></schema-name>
  <column-precision>0</column-precision>
  <column-scale>0</column-scale>
  <table-name></table-name>
  <catalog-name></catalog-name>
  <column-type>12</column-type>
  <column-type-name>VARCHAR2</column-type-name>
</column-definition>
<column-definition>
  <column-index>4</column-index>
  <auto-increment>false</auto-increment>
  <case-sensitive>true</case-sensitive>
  <currency>false</currency>
  <nullable>1</nullable>
  <signed>true</signed>
  <searchable>true</searchable>
  <column-display-size>45</column-display-size>
  <column-label>TITLE</column-label>
  <column-name>TITLE</column-name>
  <schema-name></schema-name>
  <column-precision>0</column-precision>
  <column-scale>0</column-scale>
  <table-name></table-name>
  <catalog-name></catalog-name>
  <column-type>12</column-type>
  <column-type-name>VARCHAR2</column-type-name>
</column-definition>
<column-definition>
  <column-index>5</column-index>
  <auto-increment>false</auto-increment>
  <case-sensitive>true</case-sensitive>
  <currency>false</currency>
```

```
        <nullable>1</nullable>
        <signed>true</signed>
        <searchable>true</searchable>
        <column-display-size>25</column-display-size>
        <column-label>AUTHOR</column-label>
        <column-name>AUTHOR</column-name>
        <schema-name></schema-name>
        <column-precision>0</column-precision>
        <column-scale>0</column-scale>
        <table-name></table-name>
        <catalog-name></catalog-name>
        <column-type>12</column-type>
        <column-type-name>VARCHAR2</column-type-name>
      </column-definition>
    </metadata>
    <data>
      <row>
        <col>Oracle Magazine</col>
        <col>Oracle Publishing</col>
        <col>July-August 2005</col>
        <col>Tuning Undo Tablespace</col>
        <col>Kimberly Floss</col>
      </row>
      <row>
        <col>Oracle Magazine</col>
        <col>Oracle Publishing</col>
        <col>March-April 2005</col>
        <col>Starting with Oracle ADF</col>
        <col>Steve Muench</col>
      </row>
    </data>
  </RowSet>
```

In this section, the procedure to generate a Web RowSet from a database table was explained. In the following section the Web RowSet is modified and the modified data stored in the database table.

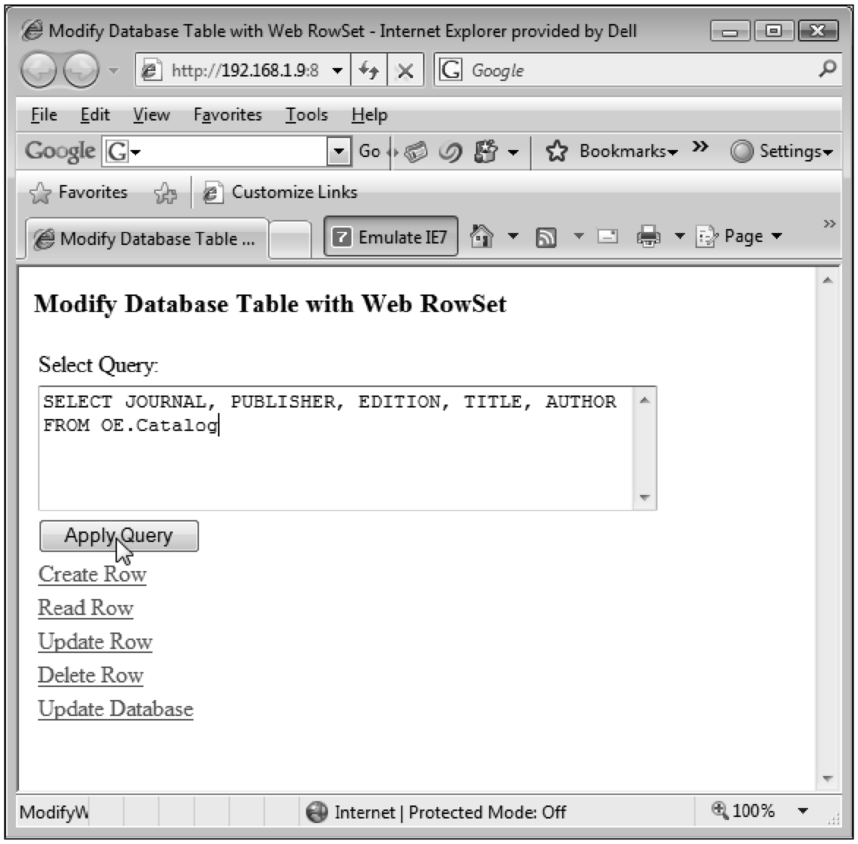
Modifying a Database Table with Web RowSet

With `ResultSet` interface, to modify the data in the database, a JDBC connection with the database is required to insert, delete, or update a database table row. With a Web RowSet, the data may be modified in the `OracleWebRowSet` object, and a connection is required only to update the database table with the data in the Web RowSet after all the modifications have been made to the Web RowSet. In this section, the data in the Web RowSet is modified and the database table is updated with the modified Web RowSet. A JDBC connection is not required to modify the data in the example Web RowSet. An `OracleWebRowSet` object is generated as in the previous section.

Create a JSP, ModifyWebRowSet.jsp, to create and modify a Web RowSet from an SQL query. Also add JSPs `CreateRow.jsp`, `ReadRow.jsp`, `UpdateRow.jsp`, `DeleteRow.jsp`, and `UpdateDatabase.jsp`, which are listed later in this chapter. `ModifyWebRowSet.jsp`, the JSP used to create and modify a Web RowSet is listed as follows:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page contentType="text/html;charset=windows-1252"%>
<%@ page session="true"%>
<html>
  <head>
    <title>Modify Database Table with Web RowSet</title>
  </head>
  <body>
    <h3>Modify Database Table with Web RowSet</h3>
      <%webrowset.WebRowSetQuery query=null;%>
    <%String selectQuery=request.getParameter("selectQuery");
    if(selectQuery!=null){
    query=new webrowset.WebRowSetQuery();
    query.generateWebRowSet(selectQuery);
    session.setAttribute("query", query);
    }
%>
    <form name="query" action="ModifyWebRowSet.jsp" method="post">
       <table>
        <tr>
          <td>Select Query:</td>
        </tr><tr><td>
            <textarea name="selectQuery" rows="5"
                 cols="50"></textarea>
          </td></tr><tr><td>
            <input class="Submit" type="submit" value="Apply Query"/>
          </td></tr>
      <tr><td><a href="CreateRow.jsp">Create Row</a></td></tr>
      <tr><td><a href="ReadRow.jsp">Read Row</a></td></tr>
      <tr><td><a href="UpdateRow.jsp">Update Row</a></td></tr>
       <tr><td><a href="DeleteRow.jsp">Delete Row</a></td></tr>
       <tr><td><a href="UpdateDatabase.jsp">Update
            Database</a></td></tr>
      </table>
    </form>
  </body>
</html>
```

The directory structure of the Web RowSet application is shown in the **Applications Navigator**. Run the `ModifyWebRowSet.jsp` JSP in JDeveloper. The JSP is displayed in a browser. Specify a SQL query to generate a Web RowSet. Click on **Apply Query** Subsequently we will modify the Web RowSet and update the database.


Modify Database Table with Web RowSet - Internet Explorer provided by Dell
http://192.168.1.9:8
Google
File Edit View Favorites Tools Help
Google Go Bookmarks Settings
Favorites Customize Links
Modify Database Table ...
Emulate IE7
Page
ModifyW
Internet | Protected Mode: Off
100%

Modify Database Table with Web RowSet

Select Query:

```
SELECT JOURNAL, PUBLISHER, EDITION, TITLE, AUTHOR
FROM OE.Catalog
```

Apply Query

Create Row
Read Row
Update Row
Delete Row
Update Database

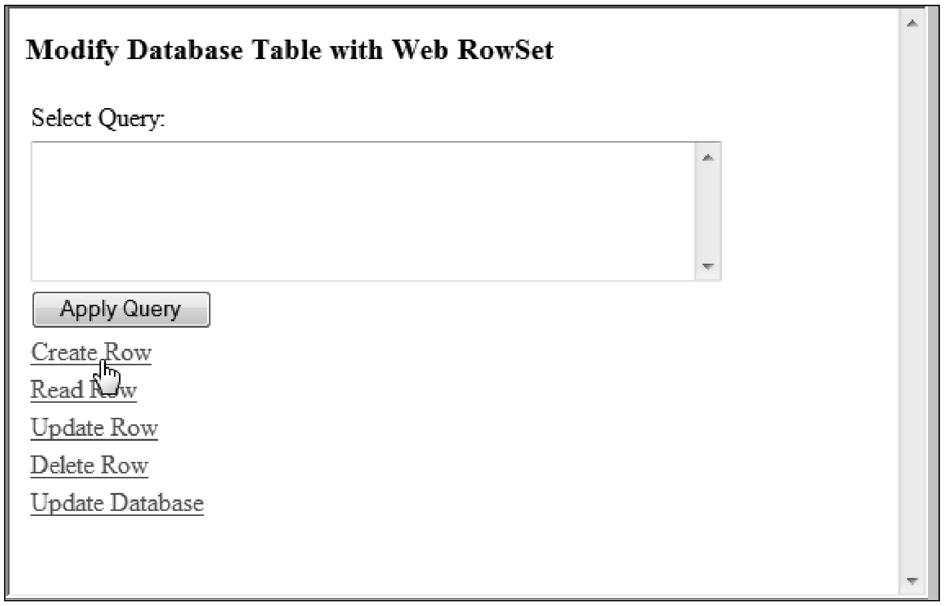
A Web RowSet is generated. We will use the Web RowSet object to create, read, update, and delete the result set obtained with the SQL query. In the `ModifyWebRowSet.jsp`, set the `WebRowSetQuery` object as a `session` object attribute:

```
session.setAttribute("query", query);
```

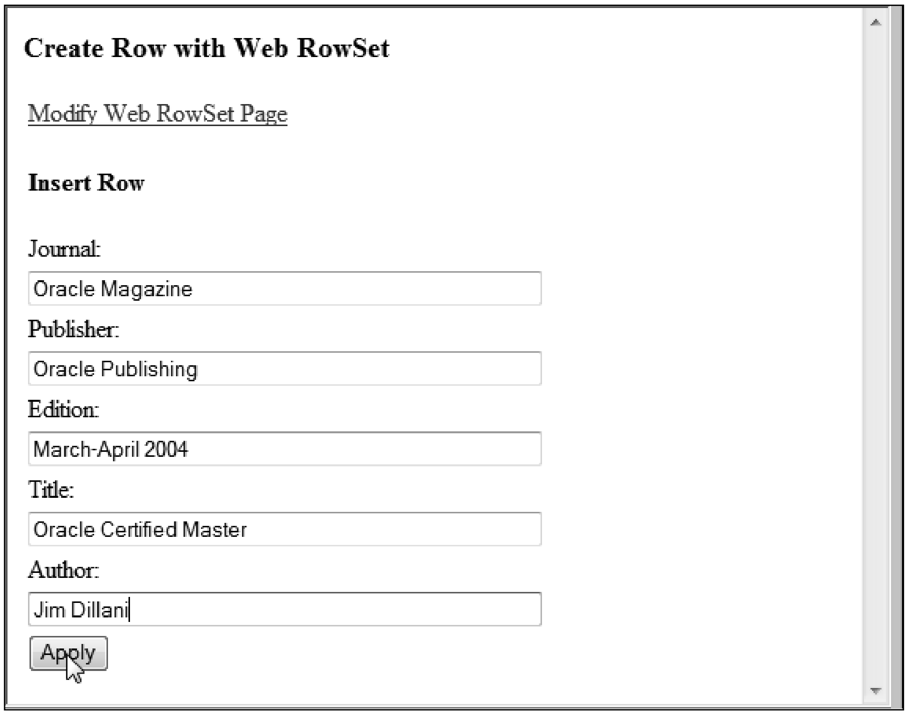
The `OracleWebRowSet` object of the `ModifyWebRowSet` object will be used in the Create, Read, Update, and Delete JSPs.

Creating a New Row

Next, create a new row in the Web RowSet. Click on the **Create Row** link in the `ModifyWebRowSet.jsp` JSP.



The `CreateRow.jsp` is displayed. Specify the row values to add and click on **Apply**.



Create Row with Web RowSet

Modify Web RowSet Page

Insert Row

Journal:
Oracle Magazine
Publisher:
Oracle Publishing
Edition:
March-April 2004
Title:
Oracle Certified Master
Author:
Jim Dillani

In the `CreateRow.jsp`, the input values are retrieved from the JSP and the `insertRow()` method of the `WebRowSetQuery` class is invoked. The `WebRowSetQuery` object is retrieved from the `session` object:

```
WebRowSetQuery query=( webrowset.WebRowSetQuery)
               session.getAttribute("query");
```

In the `insertRow()` method `OracleWebRowSet` object cursor is moved to the insert row:

```
webRowSet.moveToInsertRow();
```

Set the row values with the `updateString()` method:

```
webRowSet.updateString(1, journal);
webRowSet.updateString(2, publisher);
webRowSet.updateString(3, edition);
webRowSet.updateString(4, title);
webRowSet.updateString(5, author);
```

Add the row to the `OracleWebRowSet`:

```
webRowSet.insertRow();
```

A new row is added in the `OracleWebRowSet` object. A new row is not yet added to the database. `CreateRow.jsp` is listed as follows:

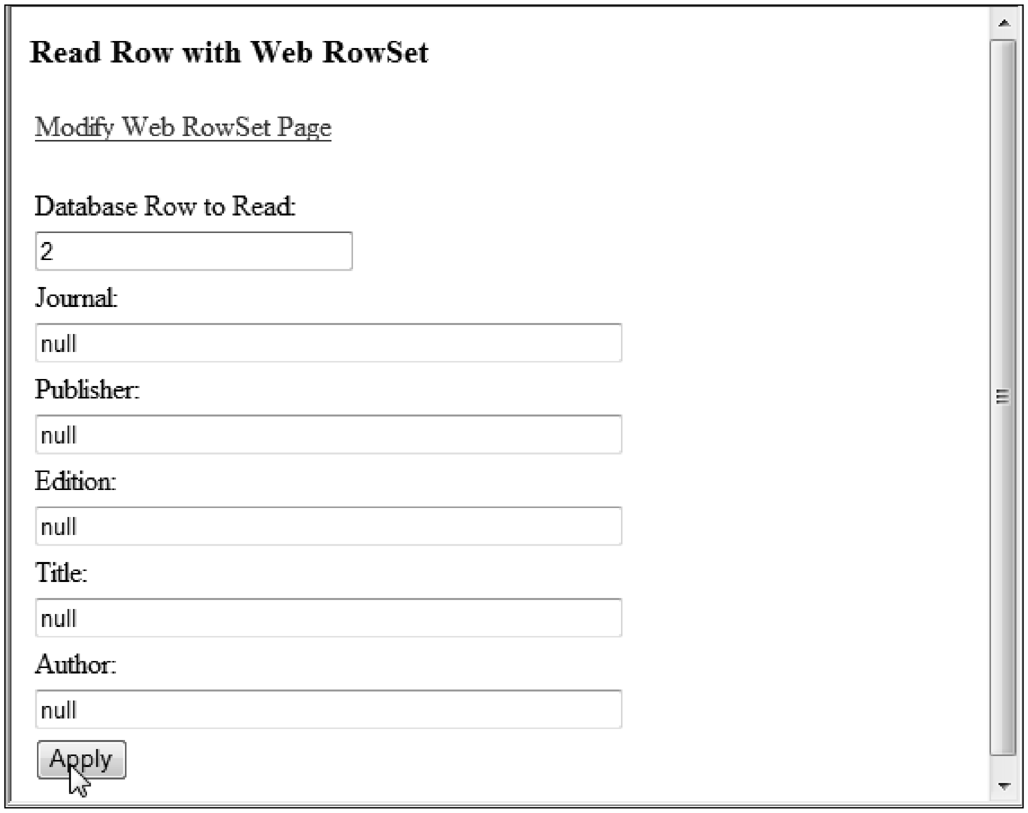
```
<%@ page contentType="text/html;charset=windows-1252"%>
<%@ page session="true"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
           charset=windows-1252">
    <title>Create Row  with Web RowSet</title>
  </head>
  <body>
    <form><h3>Create Row with Web RowSet</h3>
      <table>
       <tr><td><a href="ModifyWebRowSet.jsp">Modify Web RowSet
            Page</a></td></tr>
      </table>
    </form>
    <%
        webrowset.WebRowSetQuery query=null;
        query=(webrowset.WebRowSetQuery)
            session.getAttribute("query");
    String journal=request.getParameter("journal");
    String publisher=request.getParameter("publisher");
    String edition=request.getParameter("edition");
    String title=request.getParameter("title");
    String author=request.getParameter("author");
    if(journal!=null||publisher!=null||edition!=null||title!=null
                                             ||author!=null){
    query.insertRow(journal, publisher, edition, title, author);

    }
   %>
    <form name="query" action="CreateRow.jsp" method="post">
      <table>
         <tr>
          <td>
            <h4>Insert Row</h4>
          </td>
        </tr>
```

```
<tr>
  <td>Journal:</td>
</tr>
<tr>
  <td>
    <input name="journal" type="text" size="50"
        maxlength="250"/>
  </td>
</tr>
<tr>
  <td>Publisher:</td>
</tr>
<tr>
  <td>
    <input name="publisher" type="text" size="50"
       maxlength="250"/>
  </td>
</tr>
<tr>
  <td>Edition:</td>
</tr>
<tr>
  <td>
    <input name="edition" type="text" size="50"
      maxlength="250"/>
  </td>
</tr>
<tr>
  <td>Title:</td>
</tr>
<tr>
  <td>
    <input name="title" type="text" size="50"
     maxlength="250"/>
  </td>
</tr>
<tr>
  <td>Author:</td>
</tr>
<tr>
  <td>
    <input name="author" type="text" size="50"
     maxlength="250"/>
  </td>
```

```
            </tr>
            <tr>
              <td>
                <input class="Submit" type="submit" value="Apply"/>
              </td>
            </tr>
          </table></form></body></html>
```

Reading a Row

Next, we will read a row from the `OracleWebRowSet` object. Click on **Modify Web RowSet** link in the `CreateRow.jsp`. In the `ModifyWebRowSet` JSP click on the **Read Row** link. The `ReadRow.jsp` JSP is displayed. In the `ReadRow` JSP specify the **Database Row to Read** and click on **Apply**.



Read Row with Web RowSet

Modify Web RowSet Page

Database Row to Read: 2

Journal: null

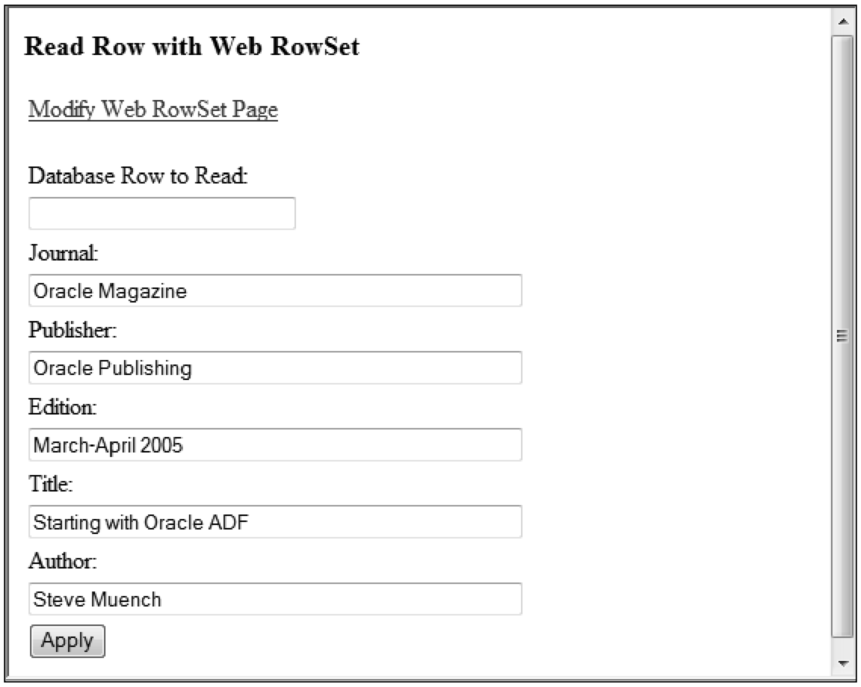
Publisher: null

Edition: null

Title: null

Author: null

The second row values are retrieved from the Web RowSet:


Read Row with Web RowSet

Modify Web RowSet Page

Database Row to Read:

Journal: Oracle Magazine

Publisher: Oracle Publishing

Edition: March-April 2005

Title: Starting with Oracle ADF

Author: Steve Muench

In the `ReadRow` JSP the `readRow()` method of the `WebRowSetQuery.java` application is invoked. The `WebRowSetQuery` object is retrieved from the `session` object.

```
WebRowSetQuery query=( webrowset.WebRowSetQuery)
               session.getAttribute("query");
```

The `String[]` values returned by the `readRow()` method are added to the `ReadRow` JSP fields. In the `readRow()` method the `OracleWebRowSet` object cursor is moved to the row to be read.

```
webRowSet.absolute(rowRead);
```

Retrieve the row values with the `getString()` method and add to `String[]`. Return the `String[]` object.

```
String[] resultSet=new String[5];
resultSet[0]=webRowSet.getString(1);
resultSet[1]=webRowSet.getString(2);
resultSet[2]=webRowSet.getString(3);
resultSet[3]=webRowSet.getString(4);
resultSet[4]=webRowSet.getString(5);
return resultSet;
```

`ReadRow.jsp` JSP is listed as follows:

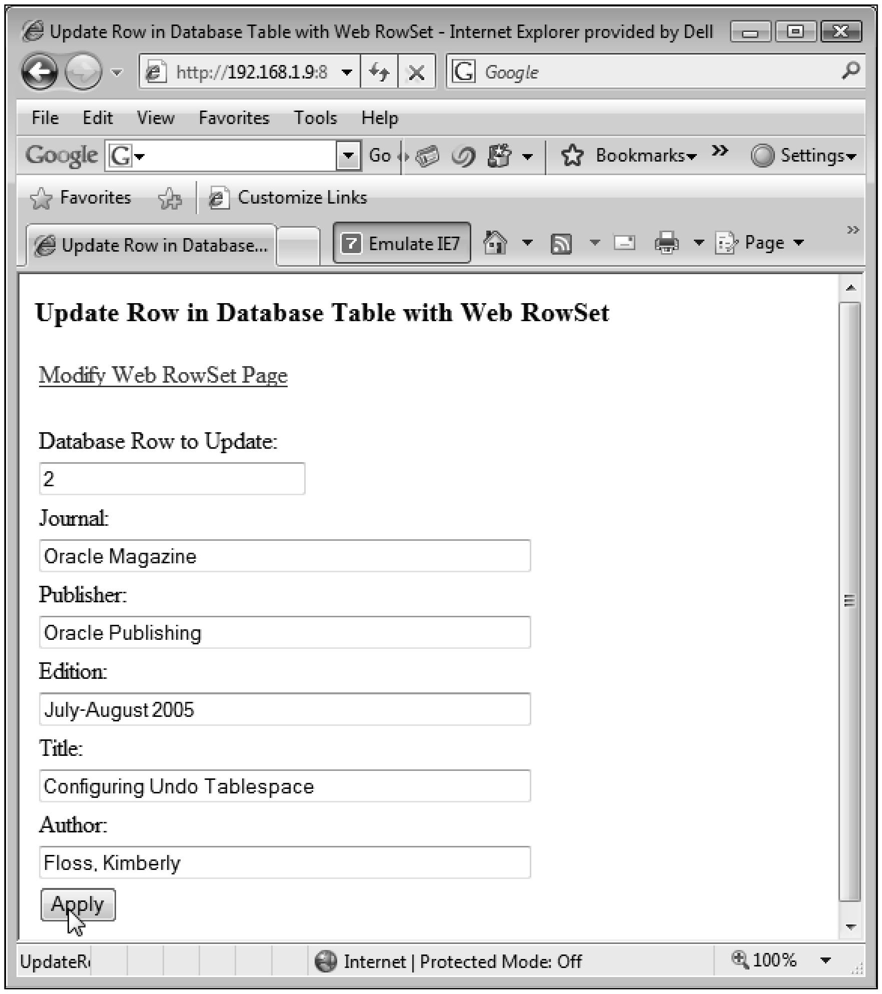
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page contentType="text/html;charset=windows-1252"%>
<%@ page session="true"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
          charset=windows-1252">
    <title>Read Row with Web RowSet</title>
  </head>
  <body>
    <form><h3>Read Row  with Web RowSet</h3>
<table>
      <tr>
          <td><a href="ModifyWebRowSet.jsp">Modify Web RowSet
          Page</a></td>
      </tr>
</table>
    </form>
    <%
        webrowset.WebRowSetQuery query=null;
        query=( webrowset.WebRowSetQuery)
          session.getAttribute("query");
    String rowRead=request.getParameter("rowRead");
    String journalUpdate=request.getParameter("journalUpdate");
    String publisherUpdate=request.getParameter("publisherUpdate");
    String editionUpdate=request.getParameter("editionUpdate");
    String titleUpdate=request.getParameter("titleUpdate");
    String authorUpdate=request.getParameter("authorUpdate");
        if((rowRead!=null))
{
    int row_Read=Integer.parseInt(rowRead);
    String[] resultSet=query.readRow(row_Read);
journalUpdate=resultSet[0];
publisherUpdate=resultSet[1];
  editionUpdate=resultSet[2];
titleUpdate=resultSet[3];
authorUpdate=resultSet[4];
   }
   %>
    <form name="query" action="ReadRow.jsp" method="post">
      <table>
        <tr>
          <td>Database Row to Read:</td>
        </tr>
        <tr>
          <td>
            <input name="rowRead"  type="text" size="25"
```

```
           maxlength="50"/>
      </td>
    </tr>
    <tr>
      <td>Journal:</td>
    </tr>
    <tr>
      <td>
        <input name="journalUpdate" value='<%=journalUpdate%>'
           type="text" size="50" maxlength="250"/>
      </td>
    </tr>
    <tr>
      <td>Publisher:</td>
    </tr>
    <tr>
      <td>
        <input name="publisherUpdate"
           value='<%=publisherUpdate%>' type="text" size="50"
           maxlength="250"/>
      </td>
    </tr>
    <tr>
      <td>Edition:</td>
    </tr>
    <tr>
      <td>
        <input name="editionUpdate" value='<%=editionUpdate%>'
           type="text" size="50" maxlength="250"/>
      </td>
    </tr>
    <tr>
      <td>Title:</td>
    </tr>
    <tr>
      <td>
        <input name="titleUpdate" value='<%=titleUpdate%>'
           type="text" size="50" maxlength="250"/>
      </td>
    </tr>
    <tr>
      <td>Author:</td>
    </tr>
    <tr>
      <td>
        <input name="authorUpdate" value='<%=authorUpdate%>'
           type="text" size="50" maxlength="250"/>
      </td>
    </tr><tr>
      <td>
```

```
          <input class="Submit" type="submit" value="Apply"/>
        </td>
      </tr>
    </table>
  </form>
</body></html>
```

Updating a Row

Next, we will update a row in the `OracleWebRowSet` object. Click on the **Modify Web RowSet Page** link in the `ReadRow` JSP. In the `ModifyWebRowSet` JSP click on the **Update Row** link. In the `UpdateRow` JSP specify the row to be updated and specify the modified values. For example, update the second row. Click on **Apply**.


Update Row in Database Table with Web RowSet - Internet Explorer provided by Dell
http://192.168.1.9:8
Google
File Edit View Favorites Tools Help
Bookmarks Settings
Favorites Customize Links
Update Row in Database...
Emulate IE7
Page
UpdateR
Internet | Protected Mode: Off
100%

Update Row in Database Table with Web RowSet

Modify Web RowSet Page

Database Row to Update:
2
Journal:
Oracle Magazine
Publisher:
Oracle Publishing
Edition:
July-August 2005
Title:
Configuring Undo Tablespace
Author:
Floss, Kimberly
Apply

The `UpdateRow` JSP invokes the `updateRow()` method of the `WebRowSetQuery` Java class. The `WebRowSetQuery` object is retrieved from the `session` object:

```
WebRowSetQuery query=( webrowset.WebRowSetQuery)
               session.getAttribute("query");
```

In the `updateRow()` method the `OracleWebRowSet` object cursor is moved to the row to be updated:

```
webRowSet.absolute(rowUpdate);
```

The row values are updated with the `updateString()` method of the `OracleWebRowSet` object:

```
webRowSet.updateString(1, journal);
webRowSet.updateString(2, publisher);
webRowSet.updateString(3, edition);
webRowSet.updateString(4, title);
webRowSet.updateString(5, author);
```

Update the `OracleWebRowSet` object with the `updateRow()` method:

```
webRowSet.updateRow();
```

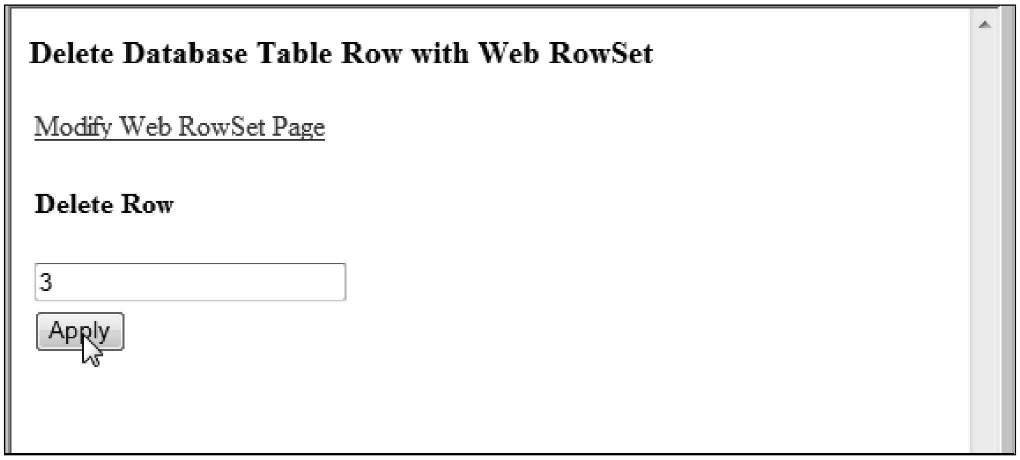
The row in the `OracleWebRowSet` object is updated. The row in the database table is not updated yet. `UpdateRow.jsp` is listed as follows:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page contentType="text/html;charset=windows-1252"%>
<%@ page session="true"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=windows-1252">
    <title>Update Row in Database Table with Web RowSet</title>
  </head>
  <body>
    <form><h3>Update Row in Database Table with Web RowSet</h3>
      <table>
      <tr>
          <td><a href="ModifyWebRowSet.jsp">Modify Web RowSet
        Page</a></td>
      </tr>
</table>
    </form>
    <%
webrowset.WebRowSetQuery query=null;
        query=( webrowset.WebRowSetQuery)
          session.getAttribute("query");
    String rowUpdate=request.getParameter("rowUpdate");
    String journalUpdate=request.getParameter("journalUpdate");
    String publisherUpdate=request.getParameter("publisherUpdate");
    String editionUpdate=request.getParameter("editionUpdate");
    String titleUpdate=request.getParameter("titleUpdate");
    String authorUpdate=request.getParameter("authorUpdate");
        if((rowUpdate!=null))
{
        System.out.println(rowUpdate +"Row to Update");
    int row_Update=Integer.parseInt(rowUpdate);
    query.updateRow(row_Update, journalUpdate, publisherUpdate,
       editionUpdate, titleUpdate, authorUpdate);
   }
   %>
    <form name="query" action="UpdateRow.jsp" method="post">
      <table>
        <tr>
          <td>Database Row to Update:</td>
        </tr>
        <tr>
          <td>
            <input name="rowUpdate" type="text" size="25"
                 maxlength="50"/>
          </td>
        </tr>
        <tr>
          <td>Journal:</td>
        </tr>
        <tr>
          <td>
            <input name="journalUpdate" type="text" size="50"
                  maxlength="250"/>
          </td>
        </tr>
        <tr>
          <td>Publisher:</td>
        </tr>
        <tr>
          <td>
            <input name="publisherUpdate" type="text" size="50"
                  maxlength="250"/>
          </td>
        </tr>
        <tr>
          <td>Edition:</td>
        </tr>
        <tr>
          <td>
            <input name="editionUpdate" type="text" size="50"
```

```
                     maxlength="250"/>
             </td>
           </tr>
           <tr>
             <td>Title:</td>
           </tr>
           <tr>
             <td>
               <input name="titleUpdate" type="text" size="50"
                    maxlength="250"/>
             </td>
           </tr>
           <tr>
             <td>Author:</td>
           </tr>
           <tr>
             <td>
               <input name="authorUpdate" type="text" size="50"
                   maxlength="250"/>
             </td>
           </tr>
           <tr>
             <td>
               <input class="Submit" type="submit" value="Apply"/>
             </td>
           </tr>
         </table>
       </form>
     </body>
   </html>
```

Deleting a Row

Next, we will delete a row from the `OracleWebRowSet` object. Click on the **Modify Web RowSet** link in the `UpdateRow` JSP. In the `ModifyWebRowSet` JSP click on the **Delete Row** link. In the **Delete Row** JSP specify the row to delete and click on **Apply**. For example, delete the third row.


Delete Database Table Row with Web RowSet

Modify Web RowSet Page

Delete Row

3

In the `DeleteRow` JSP the `deleteRow()` method of the `WebRowSetQuery` Java class is invoked. The `WebRowSetQuery` object is retrieved from the `session` object:

```
WebRowSetQuery query=( webrowset.WebRowSetQuery)
               session.getAttribute("query");
```

In the `deleteRow()` method the `OracleWebRowSet` object cursor is moved to the row to be deleted:

```
webRowSet.absolute(row);
```

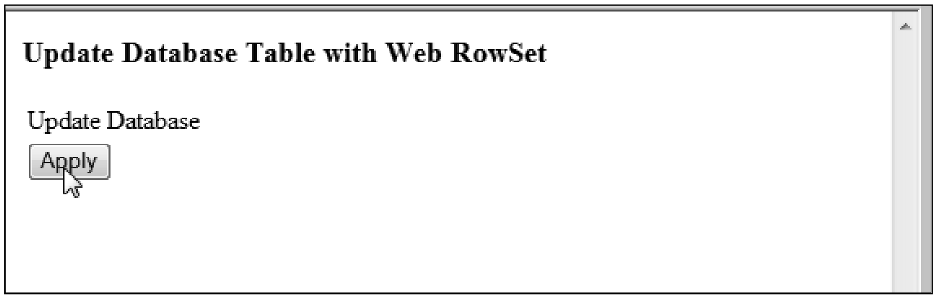
Delete the row with the `deleteRow()` method of the `OracleWebRowSet` object. The create, update, and delete operations are performed on the `OracleWebRowSet` object, not on the database table. The `DeleteRow.jsp` is as follows:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page contentType="text/html;charset=windows-1252"%>
<%@ page session="true"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
          charset=windows-1252">
    <title>Delete Database Table Row with Web RowSet</title>
  </head>
  <body>
    <form><h3>Delete Database Table Row with Web RowSet</h3>
      <table>
    <tr>
        <td><a href="ModifyWebRowSet.jsp">Modify Web RowSet
```

```
             Page</a></td>
       </tr>
   </table>
       </form>
       <%
           webrowset.WebRowSetQuery query=null;
           query=(
             webrowset.WebRowSetQuery)session.getAttribute("query");
           String deleteRow=request.getParameter("deleteRow");
       if((deleteRow!=null)){
      int delete_Row=Integer.parseInt(deleteRow);
       query.deleteRow(delete_Row);
       }
      %>
       <form name="query" action="DeleteRow.jsp" method="post">
         <table>
           <tr>
             <td><h4>Delete Row</h4></td>
           </tr>
           <tr>
             <td>
               <input name="deleteRow" type="text" size="25"
                  maxlength="50"/>
             </td>
           </tr>
           <tr>
             <td>
               <input class="Submit" type="submit" value="Apply"/>
             </td>
           </tr>
         </table>
       </form>
     </body>
   </html>
```

Updating Database Table

Next, we will update the database table with the modified `OracleWebRowSet` object. Click on the **Modify Web RowSet** link in the `DeleteRow` JSP. In the `ModifyWebRowSet` JSP, click on the **Update Database** link. In the `UpdateDatabase.jsp`, click on **Apply**.


Update Database Table with Web RowSet

In the `UpdateDatabase.jsp`, the `WebRowSetQuery` object is retrieved from the `session` object:

```
WebRowSetQuery query=(WebRowSet.WebRowSetQuery)
               session.getAttribute("query");
```

If the `WebRowSetQuery` object is not `null`, invoke the `updateDatabase()` method of the `WebRowSetQuery.java` class. Also output the XML document which represents the modifications made to the Web RowSet:

```
if(query!=null){
        query.updateDatabase();
    query.generateXMLDocument();
}
```

In the `updateDatabase()` method the database table is updated using the `acceptChanges()` method:

```
webRowSet.acceptChanges();
```

The database table, `Catalog`, is updated with the modifications made in the `OracleWebRowSet`. The `UpdateDatabase.jsp` JSP is listed below:

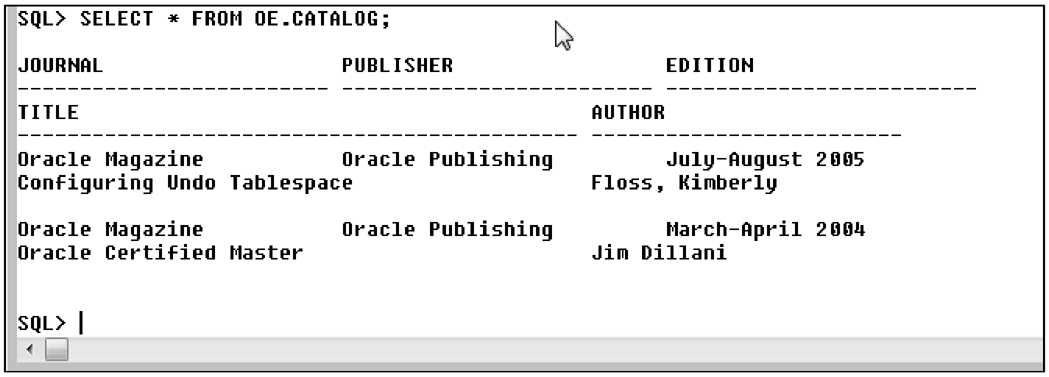
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page contentType="text/html;charset=windows-1252"%>
<%@ page session="true"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=windows-1252">
    <title>Update Database Table with Web RowSet</title>
  </head>
  <body>
    <h3>Update Database Table with Web RowSet</h3>
      <% webrowset.WebRowSetQuery query=null;%>
    <%
        String
updateDatabase=request.getParameter("updateDatabase");
        if(updateDatabase!=null)
        query
=( webrowset.WebRowSetQuery)session.getAttribute("query");
        if(query!=null)
{
        query.updateDatabase();
    query.generateXMLDocument();}
%>
   <form name="query" action="UpdateDatabase.jsp" method="post">
   <input type="hidden" name="updateDatabase" value=
                                "Update Database"/>
       <table>
        <tr>
          <td>Update Database
          </td>
      </tr>
      <tr>
          <td>
            <input class="Submit" type="submit" value="Apply"/>
          </td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

The XML document corresponding to the `OracleWebRowSet` object after the modifications are made is listed below. The modified XML document, as compared to the XML document before modifications has a row added, a row modified, and a row deleted.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RowSet PUBLIC '-//Sun Microsystems, Inc.//DTD RowSet//EN'
       'http://java.sun.com/j2ee/dtds/RowSet.dtd'>
<RowSet>
  <properties>
    <command>SELECT JOURNAL, PUBLISHER, EDITION, TITLE, AUTHOR FROM
        OE.Catalog</command>
    <concurrency>1007</concurrency>
    <datasource>jdbc/OracleDataSource</datasource>
    <escape-processing>true</escape-processing>
    <fetch-direction>1002</fetch-direction>
    <fetch-size>10</fetch-size>
    <isolation-level>2</isolation-level>
    <key-columns>
    </key-columns>
    <map></map>
    <max-field-size>0</max-field-size>
    <max-rows>3</max-rows>
    <query-timeout>0</query-timeout>
    <read-only>false</read-only>
    <rowset-type>1005</rowset-type>
    <show-deleted>false</show-deleted>
    <url>jdbc:oracle:thin:@localhost:1521:ORCL</url>
  </properties>
  <metadata>
    <column-count>5</column-count>
    <column-definition>
      <column-index>1</column-index>
      <auto-increment>false</auto-increment>
      <case-sensitive>true</case-sensitive>
      <currency>false</currency>
      <nullable>1</nullable>
      <signed>true</signed>
      <searchable>true</searchable>
      <column-display-size>25</column-display-size>
      <column-label>JOURNAL</column-label>
      <column-name>JOURNAL</column-name>
      <schema-name></schema-name>
      <column-precision>0</column-precision>
      <column-scale>0</column-scale>
      <table-name></table-name>
      <catalog-name></catalog-name>
      <column-type>12</column-type>
      <column-type-name>VARCHAR2</column-type-name>
    </column-definition>
    <column-definition>
      <column-index>2</column-index>
      <auto-increment>false</auto-increment>
      <case-sensitive>true</case-sensitive>
      <currency>false</currency>
      <nullable>1</nullable>
      <signed>true</signed>
      <searchable>true</searchable>
      <column-display-size>25</column-display-size>
      <column-label>PUBLISHER</column-label>
      <column-name>PUBLISHER</column-name>
      <schema-name></schema-name>
      <column-precision>0</column-precision>
      <column-scale>0</column-scale>
      <table-name></table-name>
      <catalog-name></catalog-name>
      <column-type>12</column-type>
      <column-type-name>VARCHAR2</column-type-name>
    </column-definition>
    <column-definition>
      <column-index>3</column-index>
      <auto-increment>false</auto-increment>
      <case-sensitive>true</case-sensitive>
      <currency>false</currency>
      <nullable>1</nullable>
      <signed>true</signed>
      <searchable>true</searchable>
      <column-display-size>25</column-display-size>
      <column-label>EDITION</column-label>
      <column-name>EDITION</column-name>
      <schema-name></schema-name>
      <column-precision>0</column-precision>
      <column-scale>0</column-scale>
      <table-name></table-name>
      <catalog-name></catalog-name>
      <column-type>12</column-type>
      <column-type-name>VARCHAR2</column-type-name>
    </column-definition>
    <column-definition>
      <column-index>4</column-index>
      <auto-increment>false</auto-increment>
      <case-sensitive>true</case-sensitive>
      <currency>false</currency>
      <nullable>1</nullable>
      <signed>true</signed>
      <searchable>true</searchable>
      <column-display-size>45</column-display-size>
      <column-label>TITLE</column-label>
      <column-name>TITLE</column-name>
      <schema-name></schema-name>
      <column-precision>0</column-precision>
      <column-scale>0</column-scale>
```

```
        <table-name></table-name>
        <catalog-name></catalog-name>
        <column-type>12</column-type>
        <column-type-name>VARCHAR2</column-type-name>
      </column-definition>
      <column-definition>
        <column-index>5</column-index>
        <auto-increment>false</auto-increment>
        <case-sensitive>true</case-sensitive>
        <currency>false</currency>
        <nullable>1</nullable>
        <signed>true</signed>
        <searchable>true</searchable>
        <column-display-size>25</column-display-size>
        <column-label>AUTHOR</column-label>
        <column-name>AUTHOR</column-name>
        <schema-name></schema-name>
        <column-precision>0</column-precision>
        <column-scale>0</column-scale>
        <table-name></table-name>
        <catalog-name></catalog-name>
        <column-type>12</column-type>
        <column-type-name>VARCHAR2</column-type-name>
      </column-definition>
    </metadata>
    <data>
      <row>
        <col>Oracle Magazine</col>
        <col>Oracle Publishing</col>
        <col>July-August 2005</col>
        <col>Configuring Undo Tablespace</col>
        <col>Floss, Kimberly</col>
      </row>
      <row>
        <col>Oracle Magazine</col>
        <col>Oracle Publishing</col>
        <col>March-April 2004</col>
        <col>Oracle Certified Master</col>
        <col>Jim Dillani</col>
      </row>
    </data>
  </RowSet>
```

Query the database table `Catalog`, the output produced. A new row has been added, a row modified, and a row deleted.



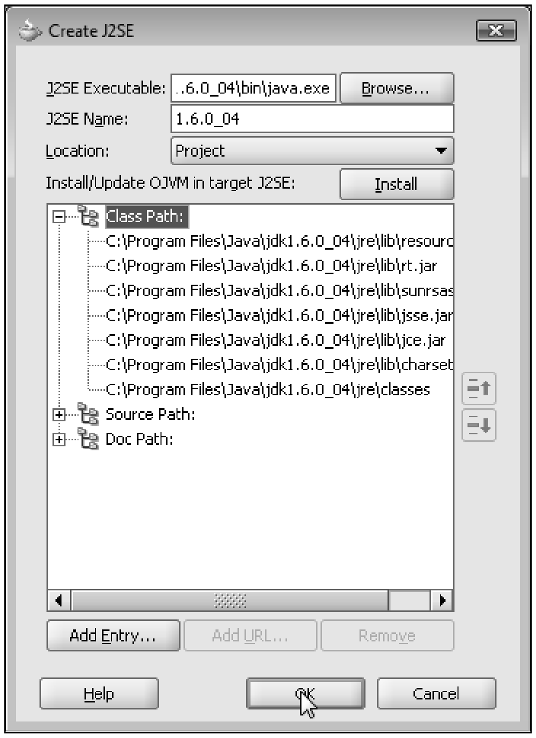
```
SQL> SELECT * FROM OE.CATALOG;

JOURNAL                   PUBLISHER                 EDITION
------------------------- ------------------------- -------------------------
TITLE                                              AUTHOR
-------------------------------------------------- -------------------------
Oracle Magazine           Oracle Publishing         July-August 2005
Configuring Undo Tablespace                        Floss, Kimberly

Oracle Magazine           Oracle Publishing         March-April 2004
Oracle Certified Master                            Jim Dillani


SQL> |
```

In this section a Web RowSet was generated from a database table, the WebRowSet was modified, and the database table updated with the modified Web RowSet.

JDBC 4.0 Version

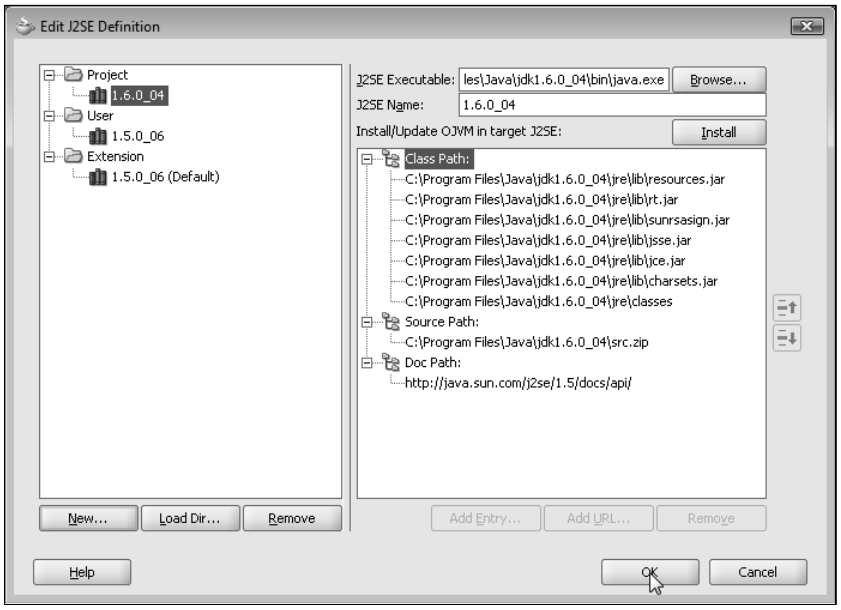
The OC4J embedded in JDeveloper 10g or JDeveloper 11g does not implement JDBC 4.0 specification. The new features in JDBC 4.0 may be availed of in a later version of JDeveloper that supports JDBC 4.0 specification.

In the JDBC 4.0 version of the web application, add the Oracle database 11g JDBC 4.0 drivers JAR file, `ojdbc6.jar`, to the `j2ee/home/applib` directory, which is in the runtime class path of a web applications running in OC4J server. Also add `ojdbc6.jar` to the project libraries by selecting **Tools | Project Properties** and subsequently selecting **Libraries | Add Jar/Directory**. For the JDBC 4.0 drivers we need to set the JDK version to JDK 6.0. Select the project node in **Applications-Navigator** and select **Tools | Project Properties**. Select the **Libraries** node in the **Project Properties** window and click on **J2SE Version** field's **Change** button to set the JDK version. In the **Edit J2SE Definition** window, click on **New**. In the **Create J2SE** window, select a JDK 6.0 **J2SE Executable** and click on **OK**.



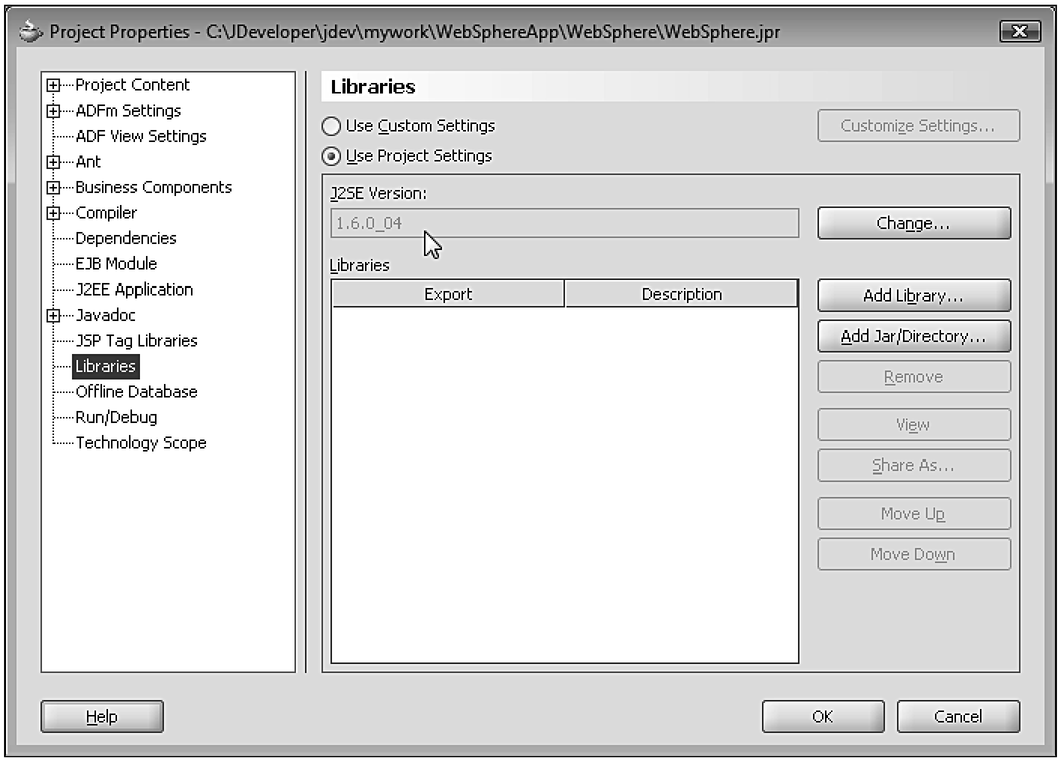
```
J2SE Executable: ..6.0_04\bin\java.exe
J2SE Name: 1.6.0_04
Location: Project
Install/Update OJVM in target J2SE:
Class Path:
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\resourc
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\rt.jar
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\sunrsas
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\jsse.jar
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\jce.jar
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\charset
    C:\Program Files\Java\jdk1.6.0_04\jre\classes
Source Path:
Doc Path:
```

In the **Edit J2SE Definition** window, select the JDK 6.0 **J2SE Definition** and click **OK**.



```
Project
    1.6.0_04
User
    1.5.0_06
Extension
    1.5.0_06 (Default)

J2SE Executable: les\Java\jdk1.6.0_04\bin\java.exe
J2SE Name: 1.6.0_04
Install/Update OJVM in target J2SE:
Class Path:
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\resources.jar
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\rt.jar
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\sunrsasign.jar
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\jsse.jar
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\jce.jar
    C:\Program Files\Java\jdk1.6.0_04\jre\lib\charsets.jar
    C:\Program Files\Java\jdk1.6.0_04\jre\classes
Source Path:
    C:\Program Files\Java\jdk1.6.0_04\src.zip
Doc Path:
    http://java.sun.com/j2se/1.5/docs/api/
```

In the **Project Properties** window the **J2SE Version** gets set to JDK 6.0.



J2SE Version: 1.6.0_04

JDBC 4.0 provides enhanced connection management. JDBC 4.0 has added support for connection state tracking using which unusable connections can be identified and closed. The connection state tracking is implemented by the connection pool manager using a new method in the `Connection` interface, `isValid()`. The connection pool manager determines if a `Connection` object is valid by invoking the `isValid()` method on the `Connection` object. If the `Connection` object is not valid the connection pool manager closes the connection. Prior to the new feature, to track connection state the connection pool manager typically had to close all the connections in a connection pool and reinitiate the connection pool if the connection pool performance got reduced due to unusable connections. A connection pool manager implements the connection state tracking as follows:

```
if(!connection.isValid())
connection.close();
```

An `SQLException` in a JDBC application might be chained to other `SQLExceptions` and a developer would be interested in retrieving the chained exceptions. In JDBC 3.0 the chained exceptions and the chained causes of the exceptions had to be retrieved by invoking the `getNextException()` and `getCause()` methods recursively.

```
catch(SQLException e)
 {
while(e != null)
 {
System.out.println("SQLException Message:" + e.getMessage());
Throwable t = e.getCause();
while(t != null)
 {
System.out.println("SQLException Cause:" + t);
t = t.getCause();
}
e = e.getNextException();
}
}
```

JDBC 4.0 has added support for the Java SE chained exception facility also called the **cause facility**. The support for the Java SE chained exception facility is implemented with following new features.

- Four new constructors in the `SQLException` class that have the `Throwable` cause as one of the parameters.
- `SQLException` class supports the enhanced `for-each` loop introduced in J2SE 5.0 to retrieve the chained exceptions and chained causes without invoking the `getNextException()` and `getCause()` methods recursively.
- The `getCause()` method supports non-`SQLExceptions`.

Chained exceptions and chained causes may be retrieved using the enhanced `for-each` loop as follows:

```
catch(SQLException sqlException)
 {
for(Throwable e : sqlException )
 {
System.out.println("Error encountered: " + e);
}
}
```

JDBC 4.0 drivers have also added support for SQL data types ROWID and National Character Set data types NCHAR, NVARCHAR, LONGNVARCHAR, and NCLOB in the RowSet interface.

The `WebRowSetQuery` class used in this chapter with the chained exceptions retrieved using the enhanced `for-each` loop is listed below:

```
package webrowset;
import oracle.jdbc.rowset.*;
import java.io.*;
import java.sql.SQLException;
public class WebRowSetQuery
 {
    public OracleWebRowSet webRowSet;
    public String selectQuery;
    public WebRowSetQuery() {
    }
    public WebRowSetQuery(OracleWebRowSet webRowSet)
 {
        this.webRowSet = webRowSet;
    }
    public void generateWebRowSet(String selectQuery)
 {
        try
 {
            webRowSet = new OracleWebRowSet();
            webRowSet.setDataSourceName("jdbc/OracleDataSource");
            webRowSet.setCommand(selectQuery);
            webRowSet.setUsername("oe");
            webRowSet.setPassword("pw");
            webRowSet.setReadOnly(false);
            webRowSet.setFetchSize(5);
            webRowSet.setMaxRows(3);
            webRowSet.execute();

        }
 catch (SQLException sqlException)
 {
            for (Throwable e: sqlException)
 {
                System.out.println("Error encountered: " + e);
            }
        }
    }
    public void generateXMLDocument()
```

```
   {
          try
   {
              OutputStreamWriter output =
                  new OutputStreamWriter(new FileOutputStream(new
                      File("c:/output/output.xml")));
              webRowSet.writeXml(output);
          } catch (SQLException sqlException)
   {
              for (Throwable e: sqlException)
   {
                  System.out.println("Error encountered: " + e);
              }
          }
          catch (IOException e)
   {
          }
      }
      public void deleteRow(int row)
   {
          try
   {
              webRowSet.absolute(row);
              webRowSet.deleteRow();
          } catch (SQLException sqlException)
   {
              for (Throwable e: sqlException)
   {
                  System.out.println("Error encountered: " + e);
              }
          }
      }
  public void insertRow(String journal, String publisher, String
                  edition,
                            String title, String author)
   {
          try
   {
              webRowSet.moveToInsertRow();
              webRowSet.updateString(1, journal);
              webRowSet.updateString(2, publisher);
              webRowSet.updateString(3, edition);
              webRowSet.updateString(4, title);
              webRowSet.updateString(5, author);
```

```
                webRowSet.insertRow();
            } catch (SQLException sqlException)
    {
                for (Throwable e: sqlException)
    {
                    System.out.println("Error encountered: " + e);
                }
            }
        }
   public void updateRow(int rowUpdate, String journal, String
                    publisher,
                              String edition, String title, String author)
    {
            try
    {
                webRowSet.absolute(rowUpdate);
                webRowSet.updateString(1, journal);
                webRowSet.updateString(2, publisher);
                webRowSet.updateString(3, edition);
                webRowSet.updateString(4, title);
                webRowSet.updateString(5, author);
                webRowSet.updateRow();
            } catch (SQLException sqlException)
    {
                for (Throwable e: sqlException)
    {
                    System.out.println("Error encountered: " + e);
                }
            }
        }
        public String[] readRow(int rowRead)
    {
            String[] resultSet = null;
            try
    {
                resultSet = new String[5];
                webRowSet.absolute(rowRead);
                resultSet[0] = webRowSet.getString(1);
                resultSet[1] = webRowSet.getString(2);
                resultSet[2] = webRowSet.getString(3);
                resultSet[3] = webRowSet.getString(4);
                resultSet[4] = webRowSet.getString(5);
            } catch (SQLException sqlException)
    {
            for (Throwable e: sqlException)
    {
                System.out.println("Error encountered: " + e);
            }
        }
        return resultSet;
    }
    public void updateDatabase()
    {
        try {
            webRowSet.acceptChanges();
        } catch (SQLException sqlException)
    {
            for (Throwable e: sqlException)
    {
                System.out.println("Error encountered: " + e);
            }
        }
    }
}
```

Summary

A persistent connection with a database is required to make updates to the database with the `ResultSet` interface. The `RowSet` extends the `ResultSet` interface. `RowSet` has the advantage of not requiring a persistent JDBC database connection for the modification of data. `WebRowSet` interface further extends the `RowSet` interface and represents a `RowSet` object as an XML document, thus facilitating the transfer of data for query and modification by remote clients who are not connected to the database. JDBC 4.0 features such as connection state tracking and Java SE chained exceptions facility may be availed of in a Web RowSet application in a server that supports JDBC 4.0.