Generating Large XML Documents from Databases

Purpose

This module describes how to easily and efficiently generate very large XML documents from a database query using the Oracle XDK Java Components

Topics

This module will discuss the following topics:

Place the cursor on this icon to display all screenshots. You can also place the cursor on each icon to see only the screenshot associated with it.

Overview

Back to List

What is XML?

XML stands for e **X** tensible **M** arkup **L** anguage. It is a markup language similar to HTML. XML is a markup language for structuring data rather than formatting information. You use XML to create a document that contains structured data that can be used or interpreted by other applications. The format or structure is straightforward and can be used by any person or program that can read text.

The Benefits of Using XML

One significant benefit of XML is that you can exchange data between applications in a simple, nonproprietary format. Because the language is extensible, you create tags that are specific to your business. For example, your document may contain tags to structure information about books. Those tags may include <title>, <author>, and <desc>. With these tags, other applications can parse the document for specific information, ignore some information, or even search for a specific title.

```
<?xml version='1.0'?>
  <LIBRARY>
   <BOOK>
    <TITLE>Is This Your Child</TITLE>
    <AUTHOR>Doris Rapp M.D.</AUTHOR>
    <DESC>Discovering and treating unrecognized allergies
```

```
        in children and adults.</DESC>
</BOOK>
</LIBRARY>
```

XML and Parsing

In order to programmatically access an XML document, you need to parse it. There are several standards-based ways to do so with the most common one being DOM (Document Object Model). DOM parsing is a W3C standard and while available in the Oracle XML Parser, it is not an optimum choice when working with large XML documents because it constructs an in-memory document tree that can be more than 10 times the size of the original document.

There is an alternative to DOM called SAX (Simple API for XML), which is an event-driven form of parsing and thus consumes very few resources regardless of the document size. SAX parsing is also available in the Oracle XML Parser. While SAX parsing does not support in-memory manipulation of a document, it can effectively be used to do simple transformations that do not require navigating backwards or significant structure change of the entire document.

Obviously, preparing a document to be printed is a simple transformation that is well suited for SAX parsing. Instead of requiring you to create the necessary event handlers to implement printing, the Oracle SAX Parser provide this new class built-in **XMLSAXSerializer** .

XML, Databases and the XDK

The ability to extract data with its meta-data providing the context is a common XML function. Unfortunately, from a programmer's viewpoint, this data can many times consist of hundreds if not thousands of rows that then need to be encapsulated into XML. The standard way to implement this would be to use the DOM to construct this document, however because of the need to construct it entirely in memory before it could be serialized, this approach is not practical.

The Oracle XDK includes the XML SQL Utility, which simplifies the extraction of the data already tagged in XML, and with its ability to generate SAX events as its XML output along with the new XMLSAXSerializer class makes this implementation easy and efficient.

Prerequisites

Back to List

Before starting this module, you should have:

☐ **1.** Completed the Configuring Linux for the Installation of Oracle Database 10g lesson

☐ **2.** Completed the Installing the Oracle Database 10g on Linux lesson

☐ **3.** Completed the Installing Oracle9i JDeveloper on Linux lesson.

Configuring a JDeveloper Project to use the XDK

Back to List

Having installed Oracle Database 10g as part of the pre-requisites, you will have installed the XDK. The main part of the XDK can be found at <ORACLE_HOME>/xdk, the Java XDK library (xmlparserv2.jar) and the Java XSU library (xsu12.jar) in <ORACLE_HOME>/lib. Oracle's JDBC driver can also be found at <ORACLE_HOME>/jdbc/lib/classes12.jar.

In order to build an XML Application using JDeveloper , you need to perform the following steps:

☐ **1.** Create a JDeveloper Project.

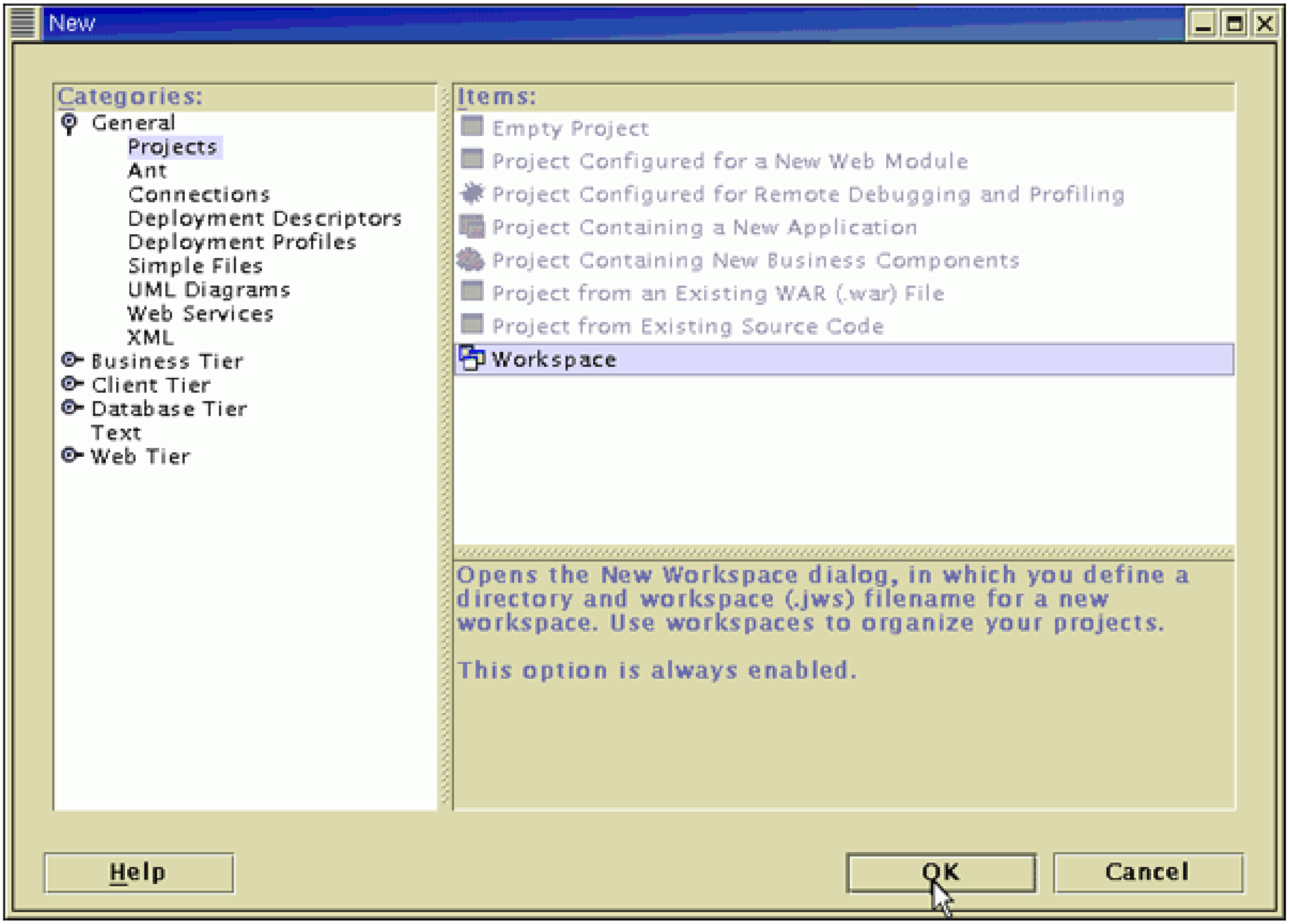
☐ **2.** Configure the project for XML

Create a JDeveloper Project

You need to create a workspace and project before you start creating the application.

☐ **1.** Open JDeveloper.

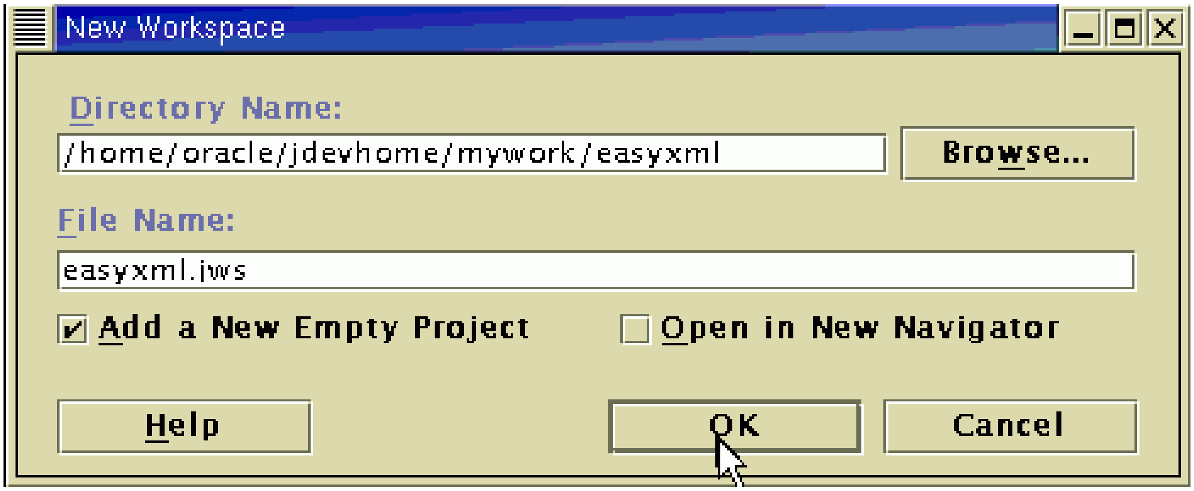
☐ **2.** Click **File > New** . Select **Projects** from the category and select **Workspace** . Click **OK** .


New
Categories:
General
Projects
Ant
Connections
Deployment Descriptors
Deployment Profiles
Simple Files
UML Diagrams
Web Services
XML
Business Tier
Client Tier
Database Tier
Text
Web Tier
Items:
Empty Project
Project Configured for a New Web Module
Project Configured for Remote Debugging and Profiling
Project Containing a New Application
Project Containing New Business Components
Project from an Existing WAR (.war) File
Project from Existing Source Code
Workspace
Help
OK
Cancel

Opens the New Workspace dialog, in which you define a directory and workspace (.jws) filename for a new workspace. Use workspaces to organize your projects.

This option is always enabled.

☐ 3. Change the Directory and Filename as **easyxml** . Make sure **Add a New Empty Project** is checked . Click **OK** .



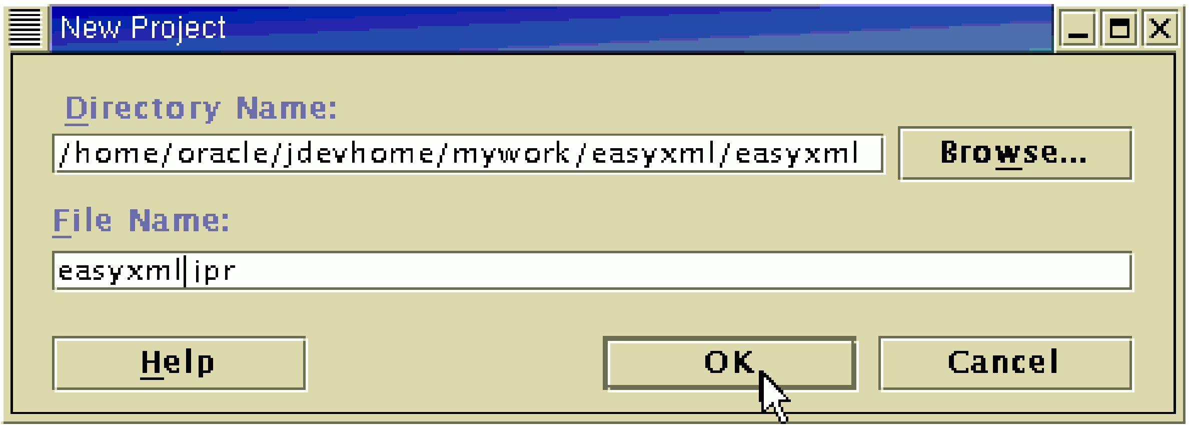
Directory Name: /home/oracle/jdevhome/mywork/easyxml

File Name: easyxml.jws

☑ Add a New Empty Project

☐ Open in New Navigator

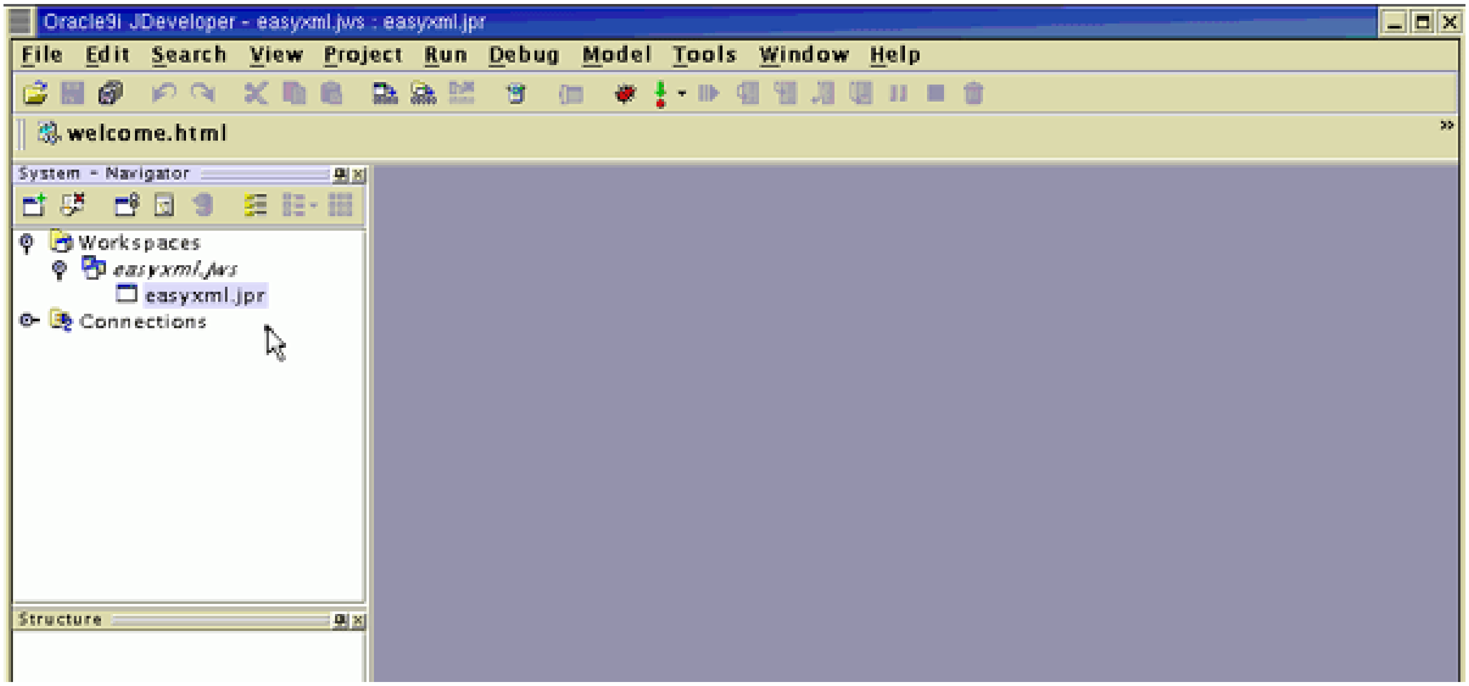
☐ 4. Enter the project directory and file name as **easyxml** and click **OK** .



Directory Name: /home/oracle/jdevhome/mywork/easyxml/easyxml

File Name: easyxml.jpr

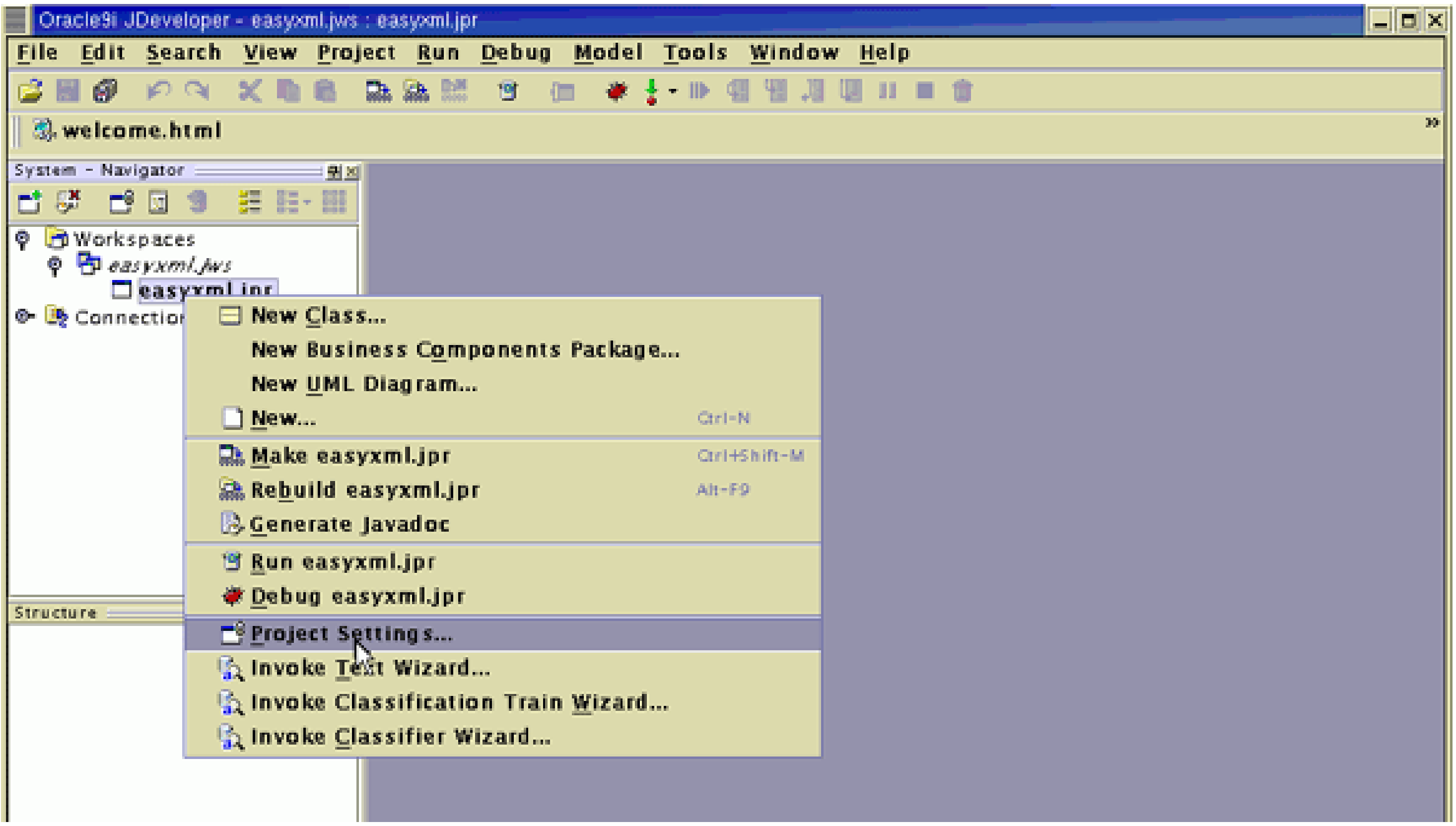
☐ **5.** Once the Project is created you will see it in the System Navigator.


Oracle9i JDeveloper - easyxml.jws : easyxml.jpr
File Edit Search View Project Run Debug Model Tools Window Help
welcome.html
System - Navigator
Workspaces
easyxml.jws
easyxml.jpr
Connections
Structure

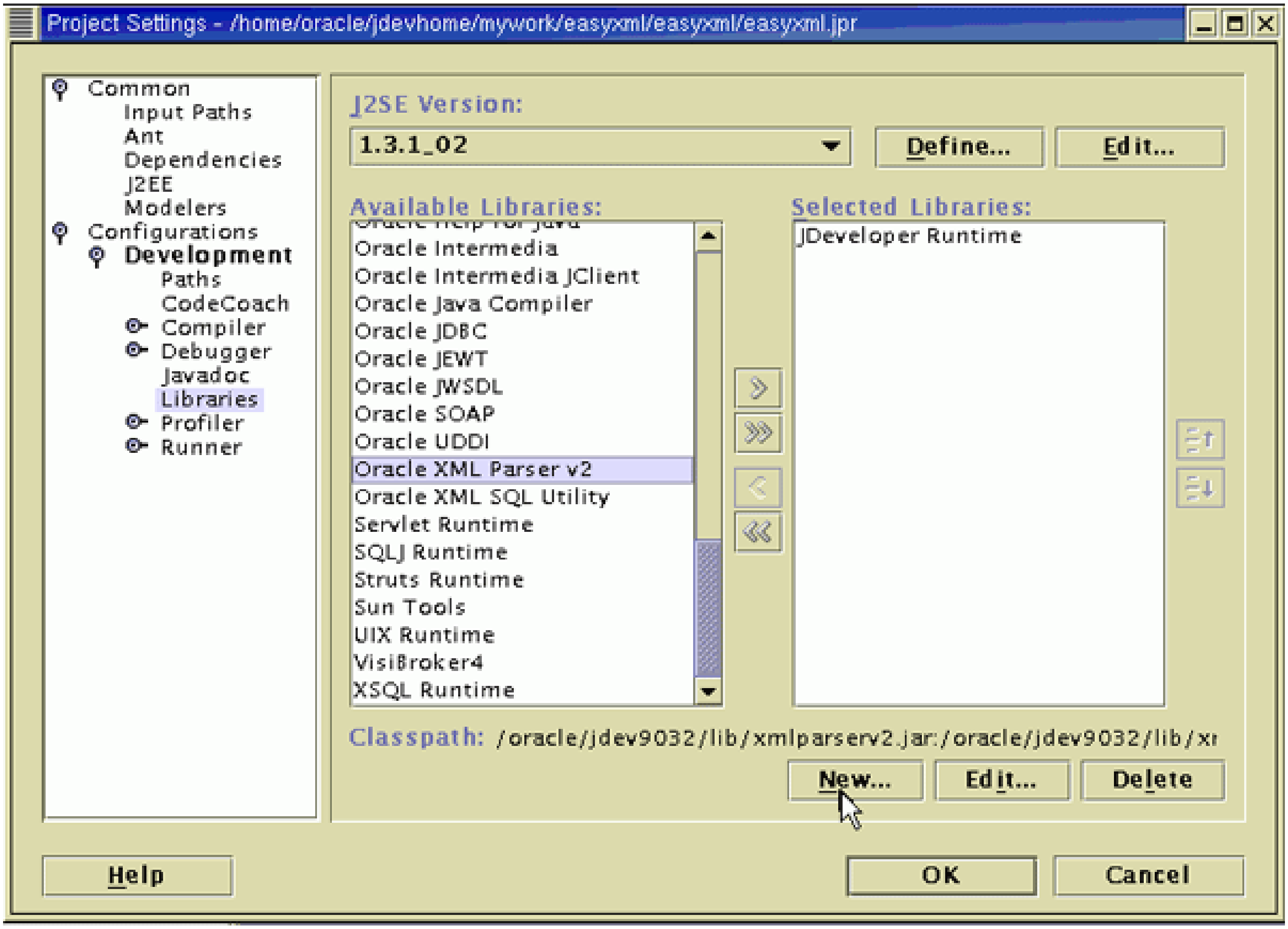
Configure the project for XML

You need set the projects classpath to include the XML libraries. Perform the following:

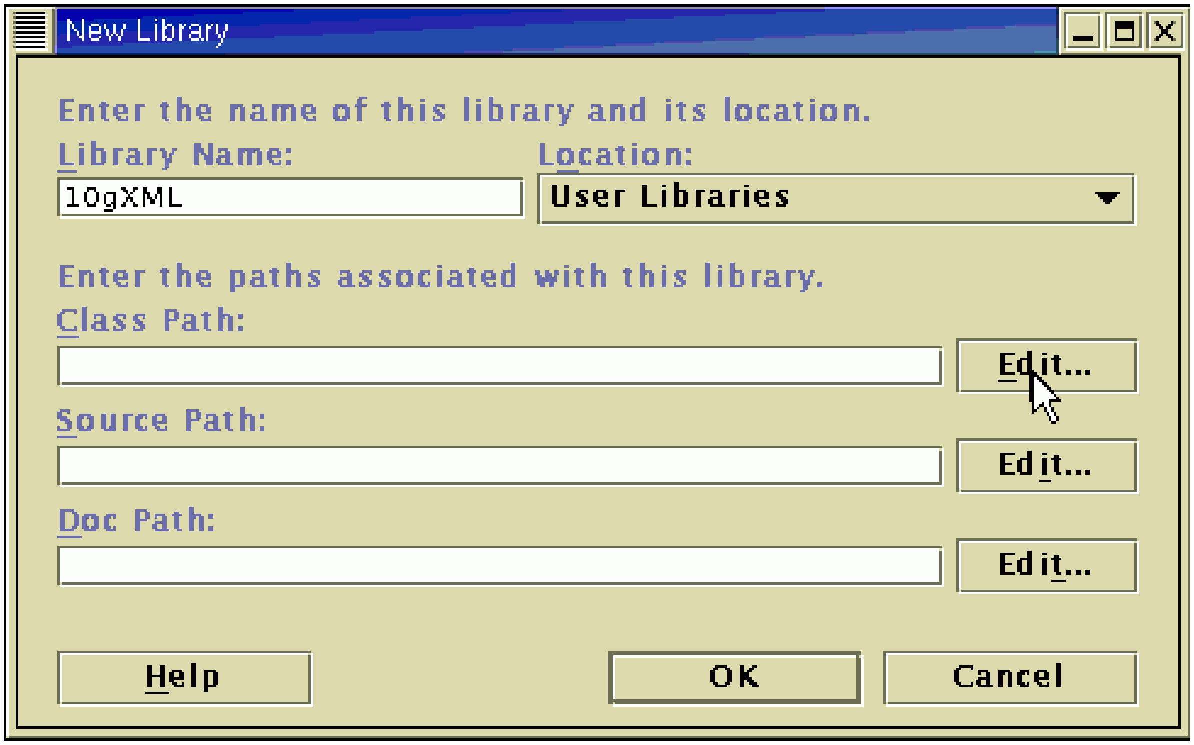
☐ **1.** Right click on the easyxml.jpr project and select **Project Settings** .


Oracle9i JDeveloper - easyxml.jws : easyxml.jpr
File Edit Search View Project Run Debug Model Tools Window Help
welcome.html
System - Navigator
Workspaces
easyxml.jws
easyxml.jpr
Connections
Structure
New Class...
New Business Components Package...
New UML Diagram...
New... Ctrl-N
Make easyxml.jpr Ctrl+Shift-M
Rebuild easyxml.jpr Alt-F9
Generate Javadoc
Run easyxml.jpr
Debug easyxml.jpr
Project Settings...
Invoke Text Wizard...
Invoke Classification Train Wizard...
Invoke Classifier Wizard...

☐ **2.** Select the **Libraries** entry in the settings navigator. Here you can configure the CLASSPATH of your project. In the Available Libraries you might already have entries for the **Oracle XML Parser V2** and the **Oracle XML SQL Utility** . Since you are using an older version of JDeveloper these are not the 10g libraries and you need to create some new librarys. Click the **New** button.



☐ **3.** Name the library **10gXML** and press the **Edit** button for the Class Path.



Enter the name of this library and its location.

Library Name: 10gXML

Location: User Libraries

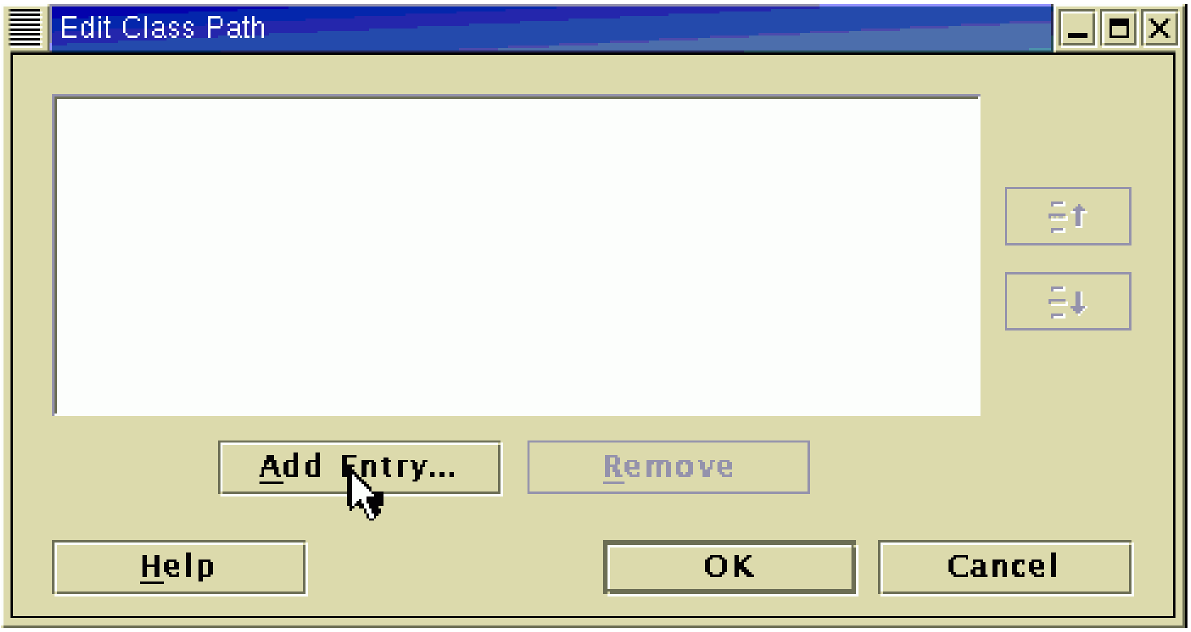
Enter the paths associated with this library.

Class Path:

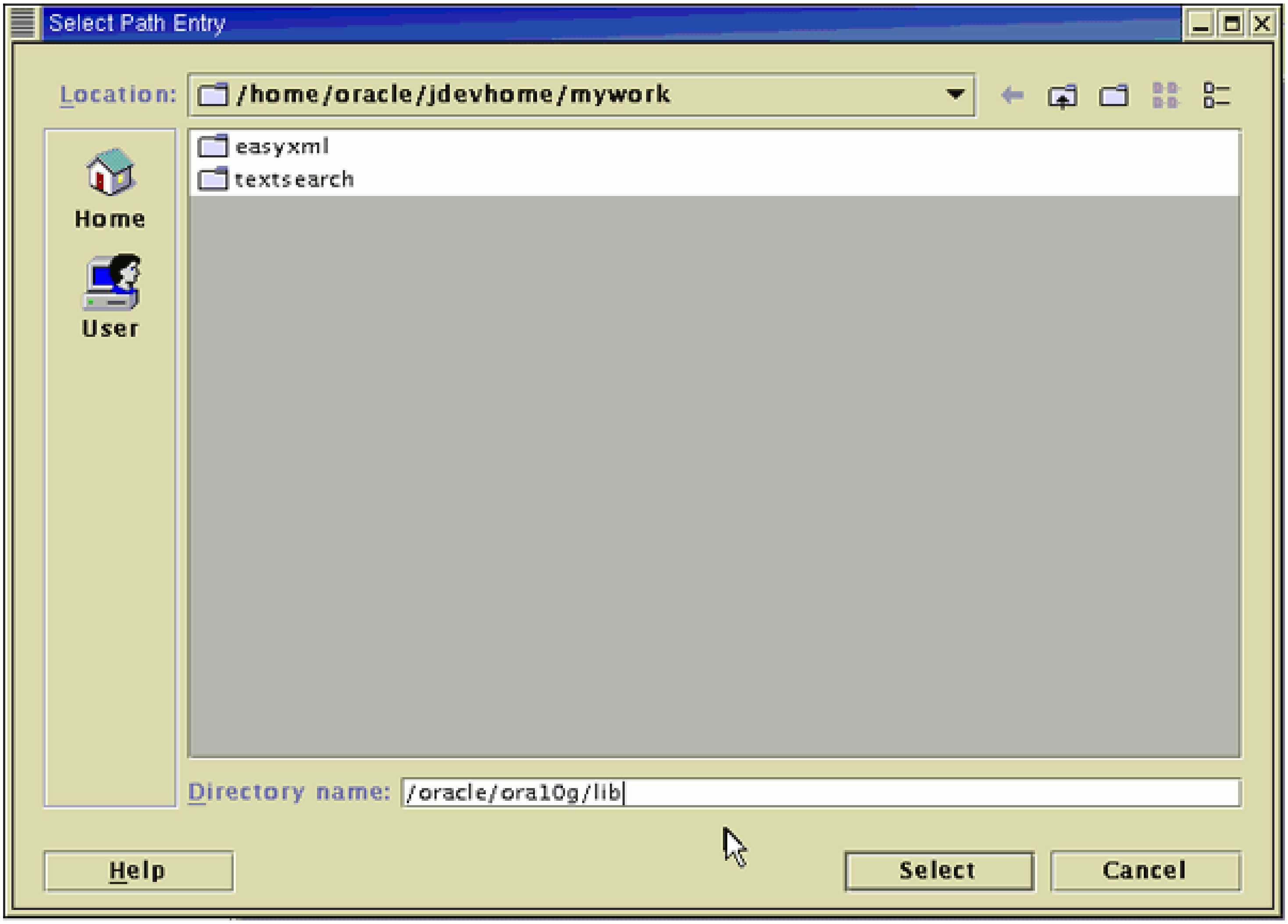
Source Path:

Doc Path:

☐ **4.** Click **Add Entry** .

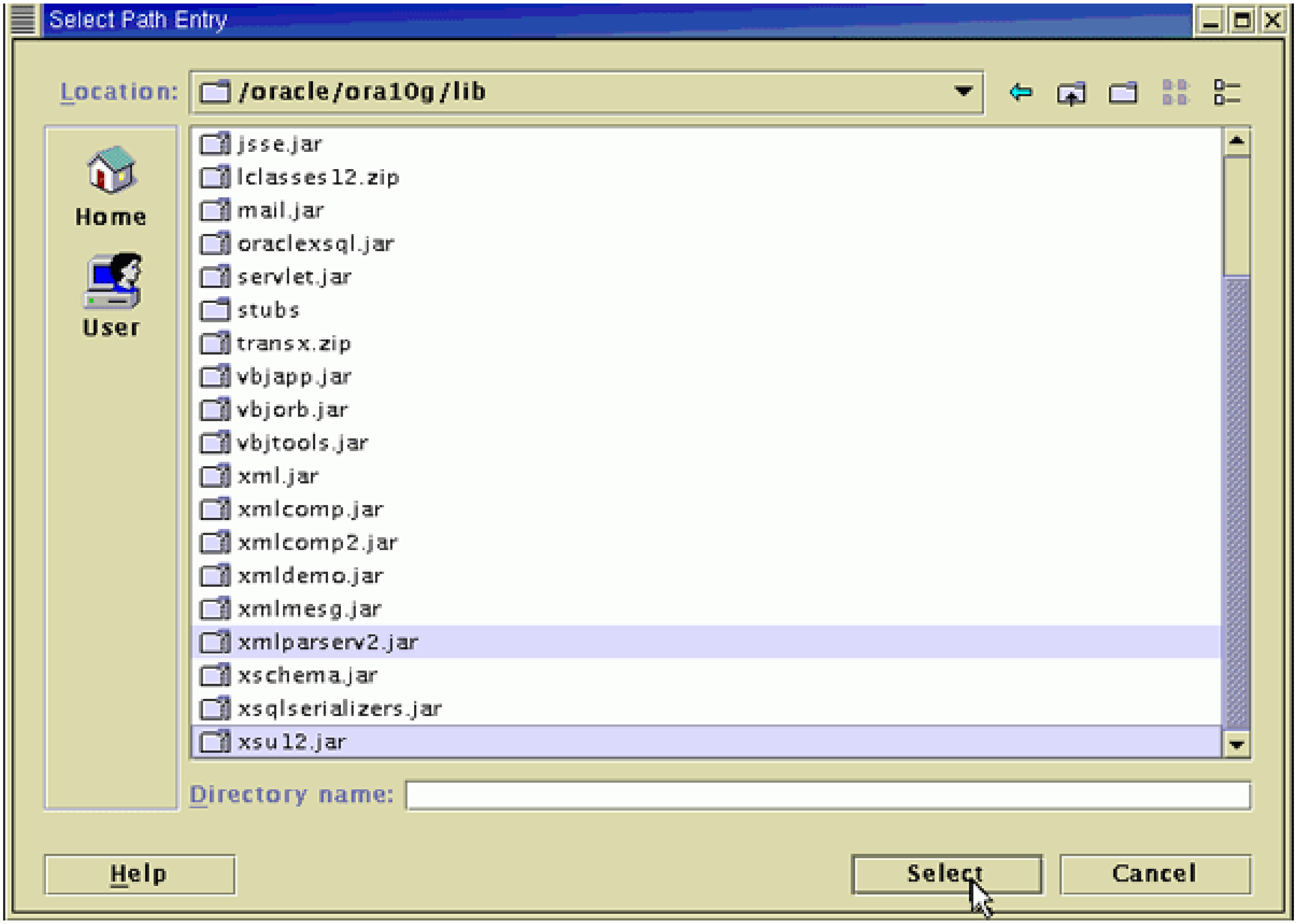


☐ **5.** In the Directory Name field enter **/oracle/ora10g/lib** and press **enter** .

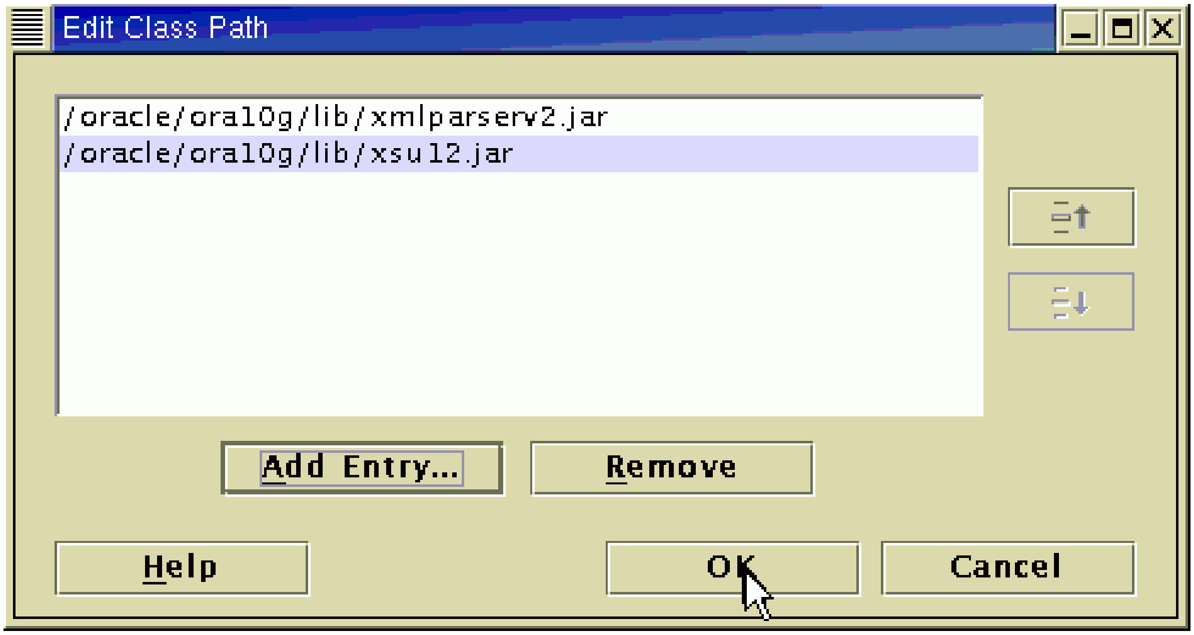


Directory name: /oracle/ora10g/lib

☐ **6.** Select **`xmlparserv2.jar`** and **`xsu12.jar`** and click **Select** .

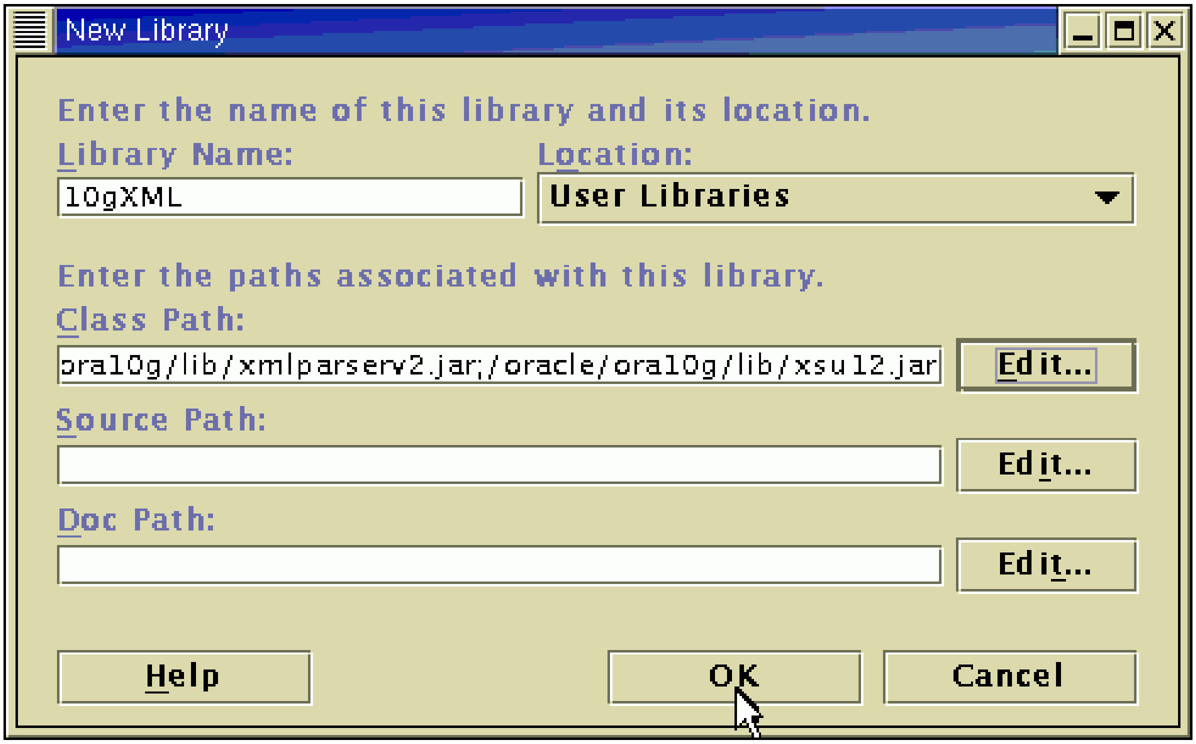


☐ **7.** Click **OK** to add the jar files to the classpath.



Edit Class Path
/oracle/ora10g/lib/xmlparserv2.jar
/oracle/ora10g/lib/xsu12.jar
Add Entry...
Remove
Help
OK
Cancel

☐ **8.** Click **OK** to create the library.



Enter the paths associated with this library.

Library Name: 10gXML

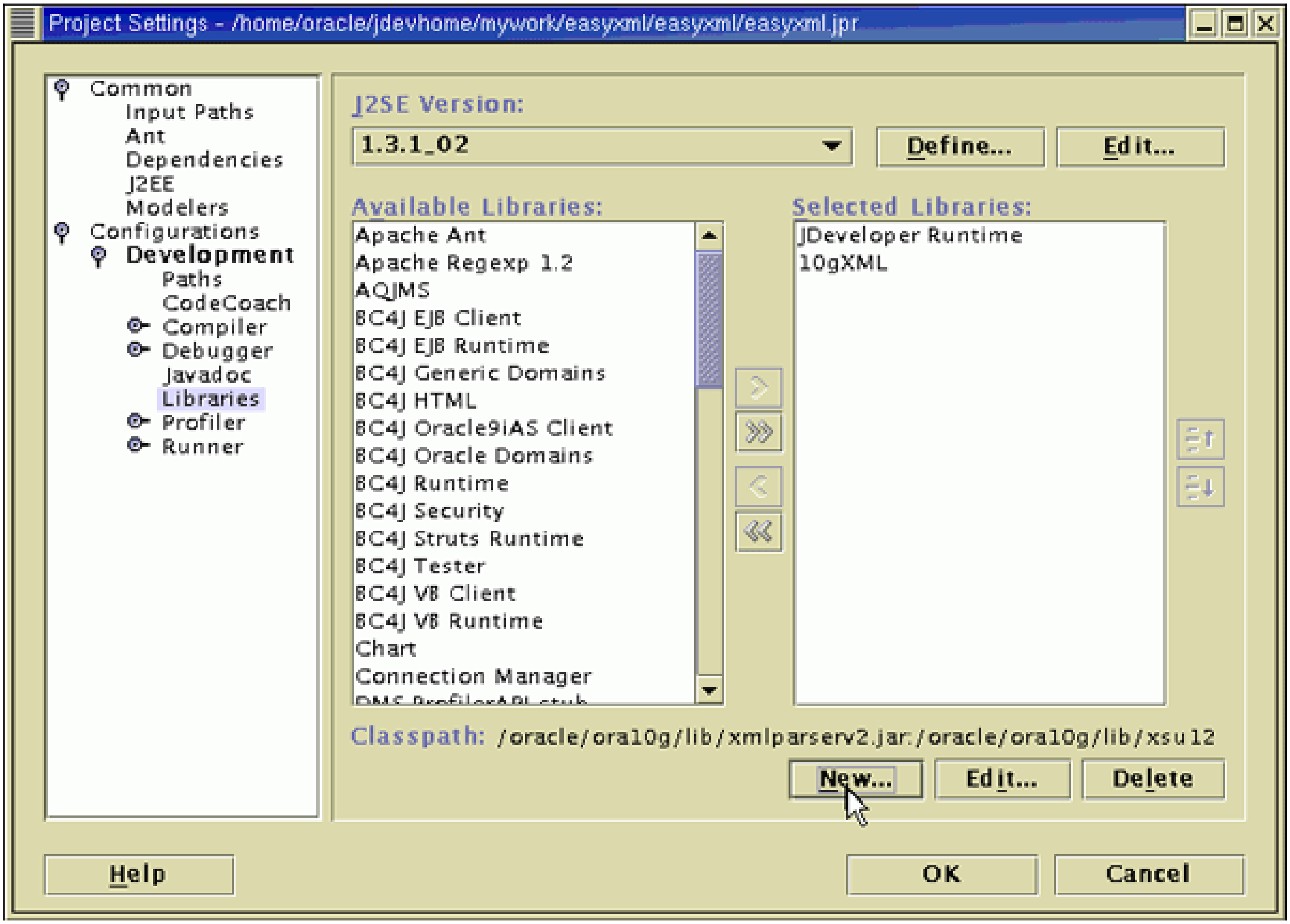
Location: User Libraries

Class Path: ora10g/lib/xmlparserv2.jar;/oracle/ora10g/lib/xsu12.jar

Source Path:

Doc Path:

☐ **9.** You need to also create a library for JDBC. Click the **New** button again.



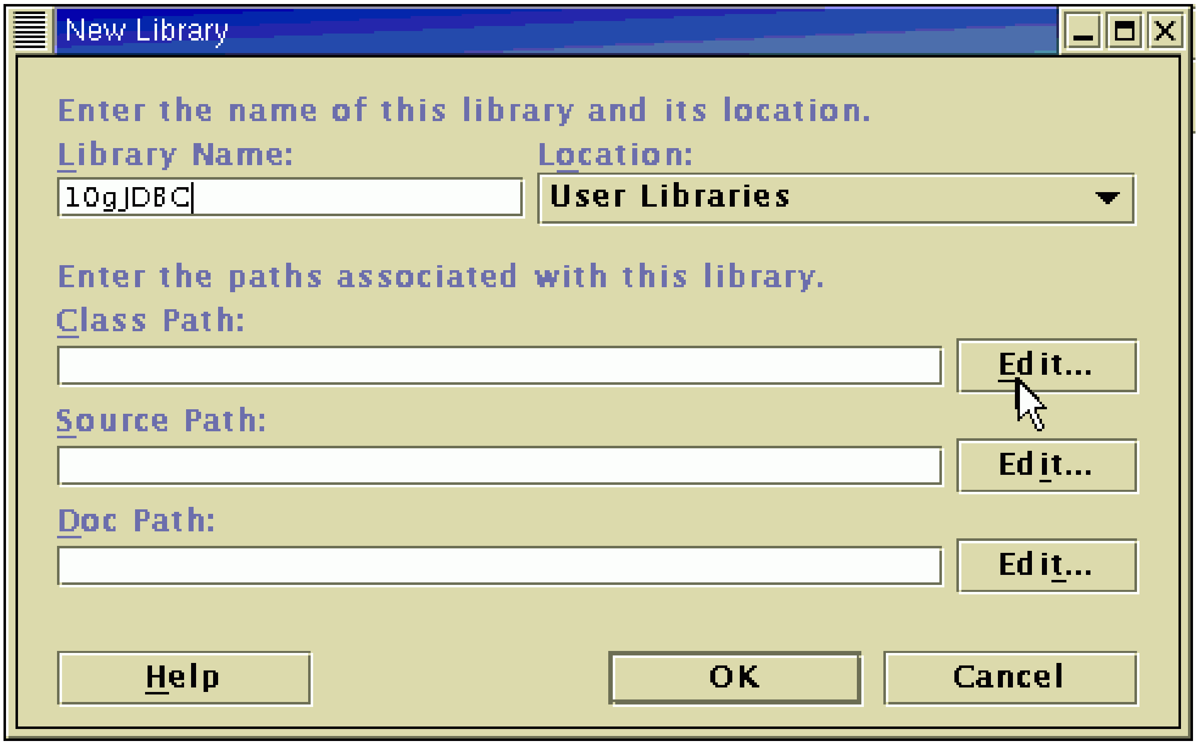
J2SE Version: 1.3.1_02

Available Libraries:

Selected Libraries:
JDeveloper Runtime
10gXML

Classpath: /oracle/ora10g/lib/xmlparserv2.jar:/oracle/ora10g/lib/xsu12

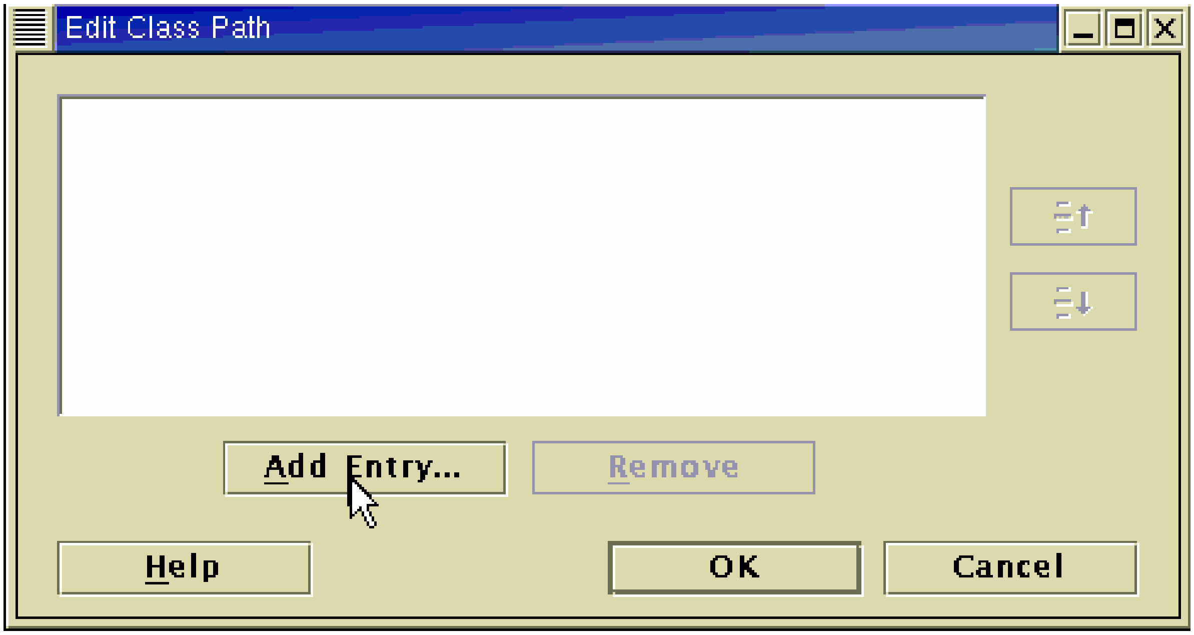
☐ **10.** Name the library **10gJDBC** and press the **Edit** button for the Class Path.



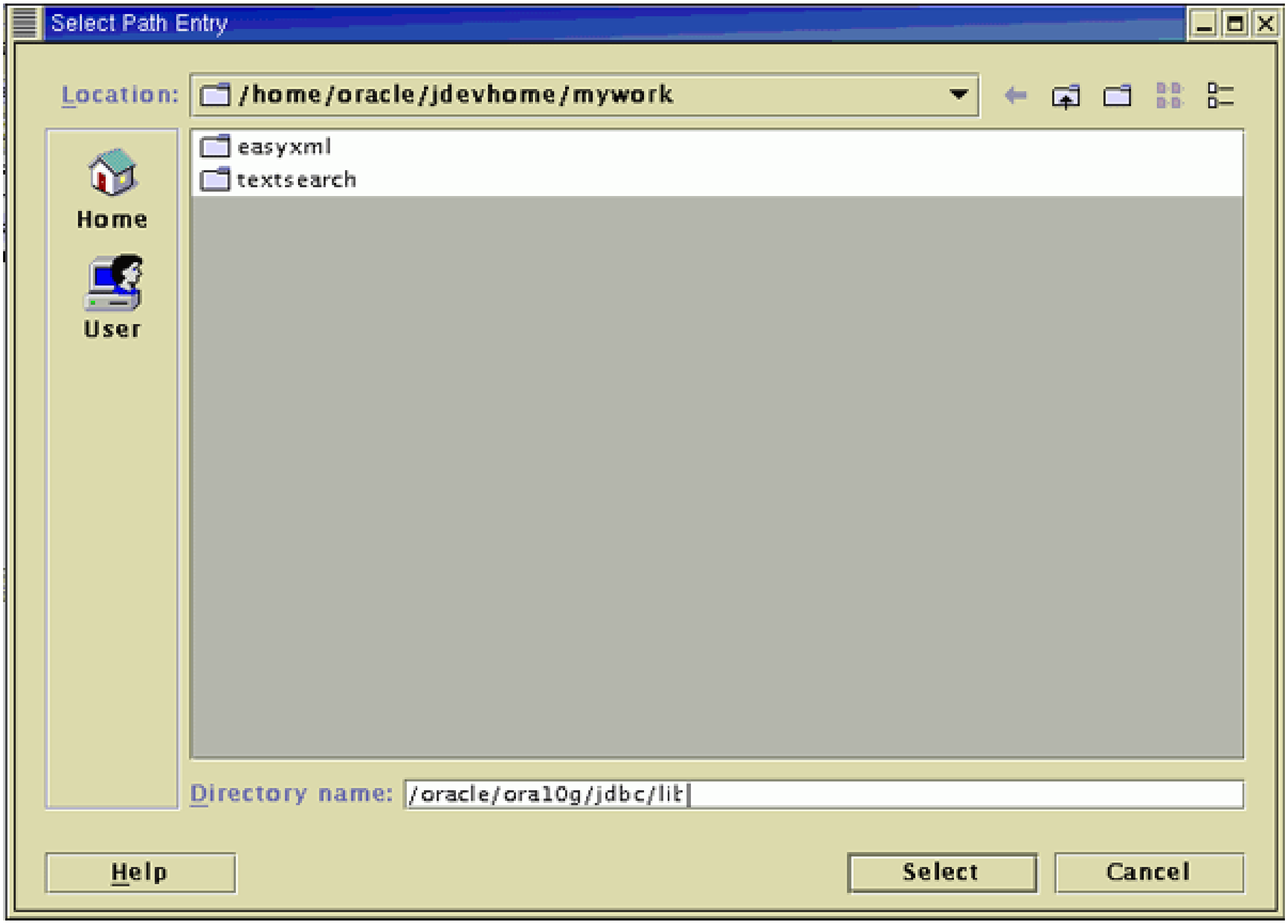
Library Name: 10gJDBC

Location: User Libraries

☐ **11.** Click **Add Entry** .

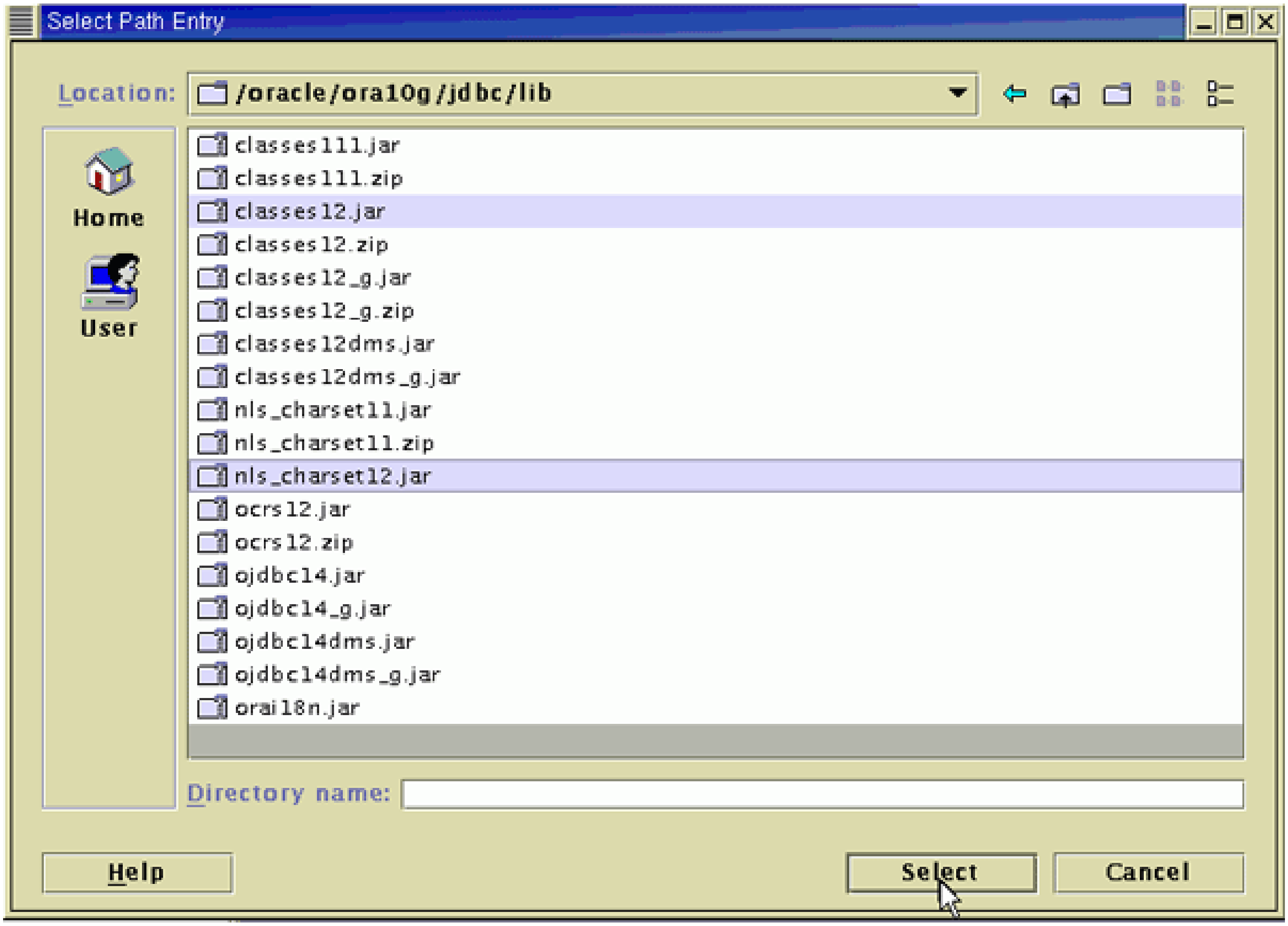


☐ 12. In the Directory Name field enter **/oracle/ora10g/jdbc/lib** and press **enter** .

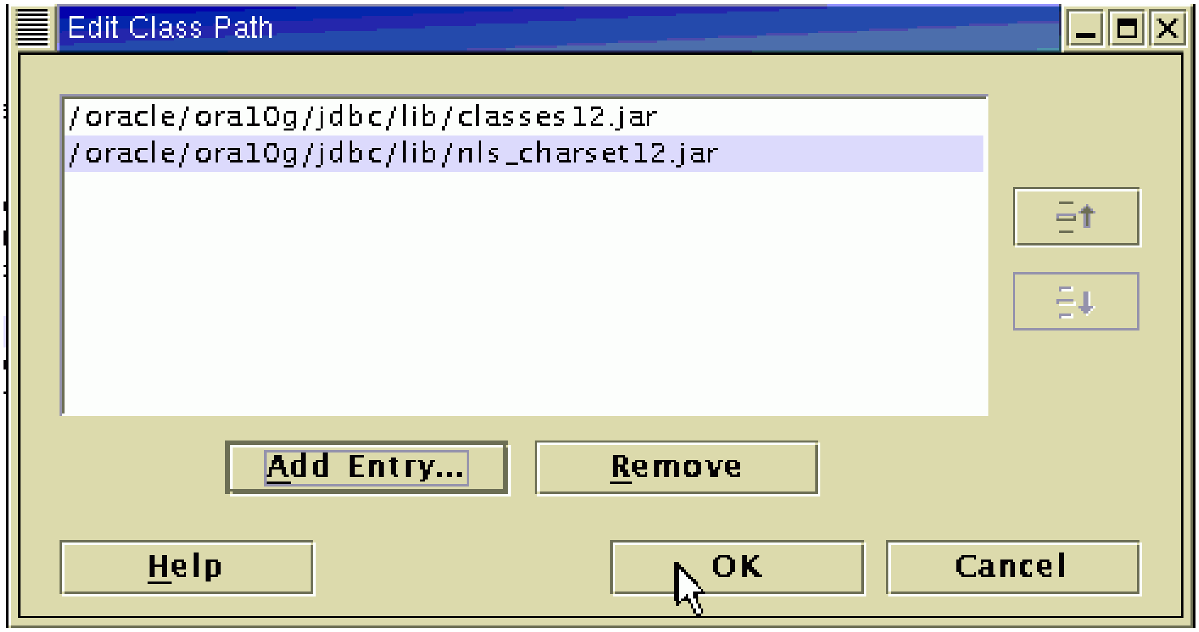


Directory name: /oracle/ora10g/jdbc/lib

☐ 13. Select **`classes12.jar`** and **`nls_charset12.jar`** and click **Select** .

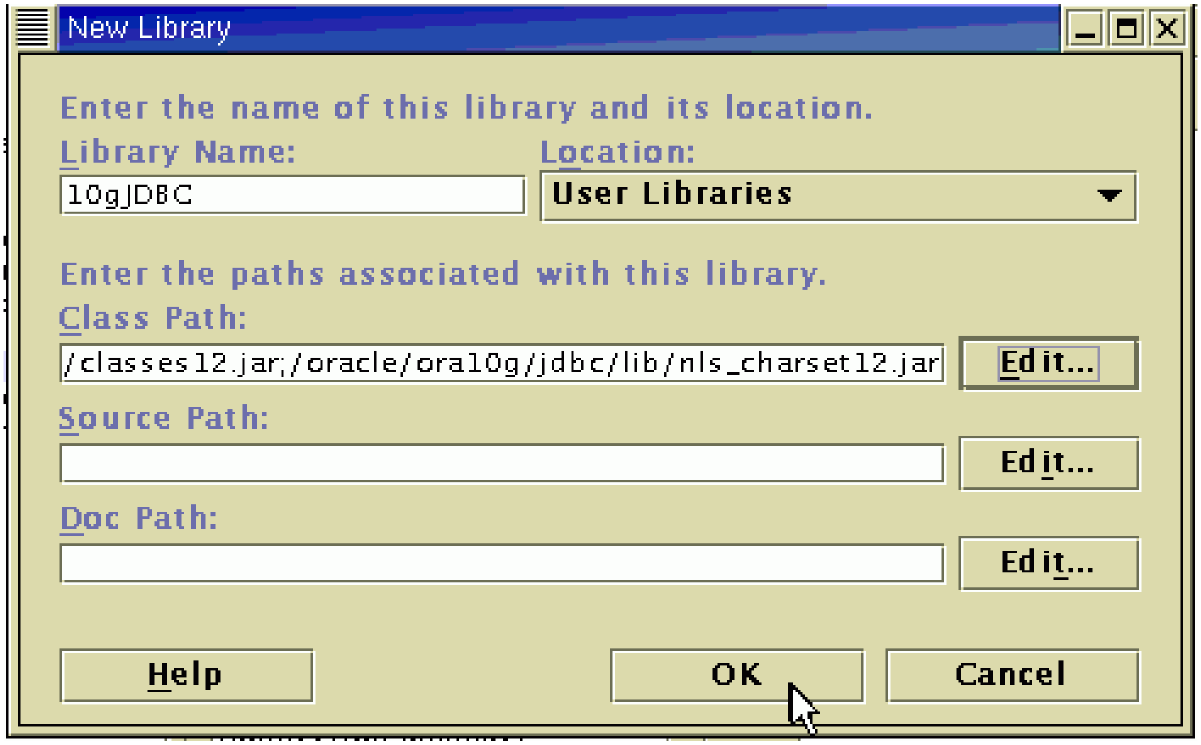


☐ **14.** Click **OK** to add the jar files to the classpath.



```
/oracle/ora10g/jdbc/lib/classes12.jar
/oracle/ora10g/jdbc/lib/nls_charset12.jar
```

☐ **15.** Click **OK** to create the library.

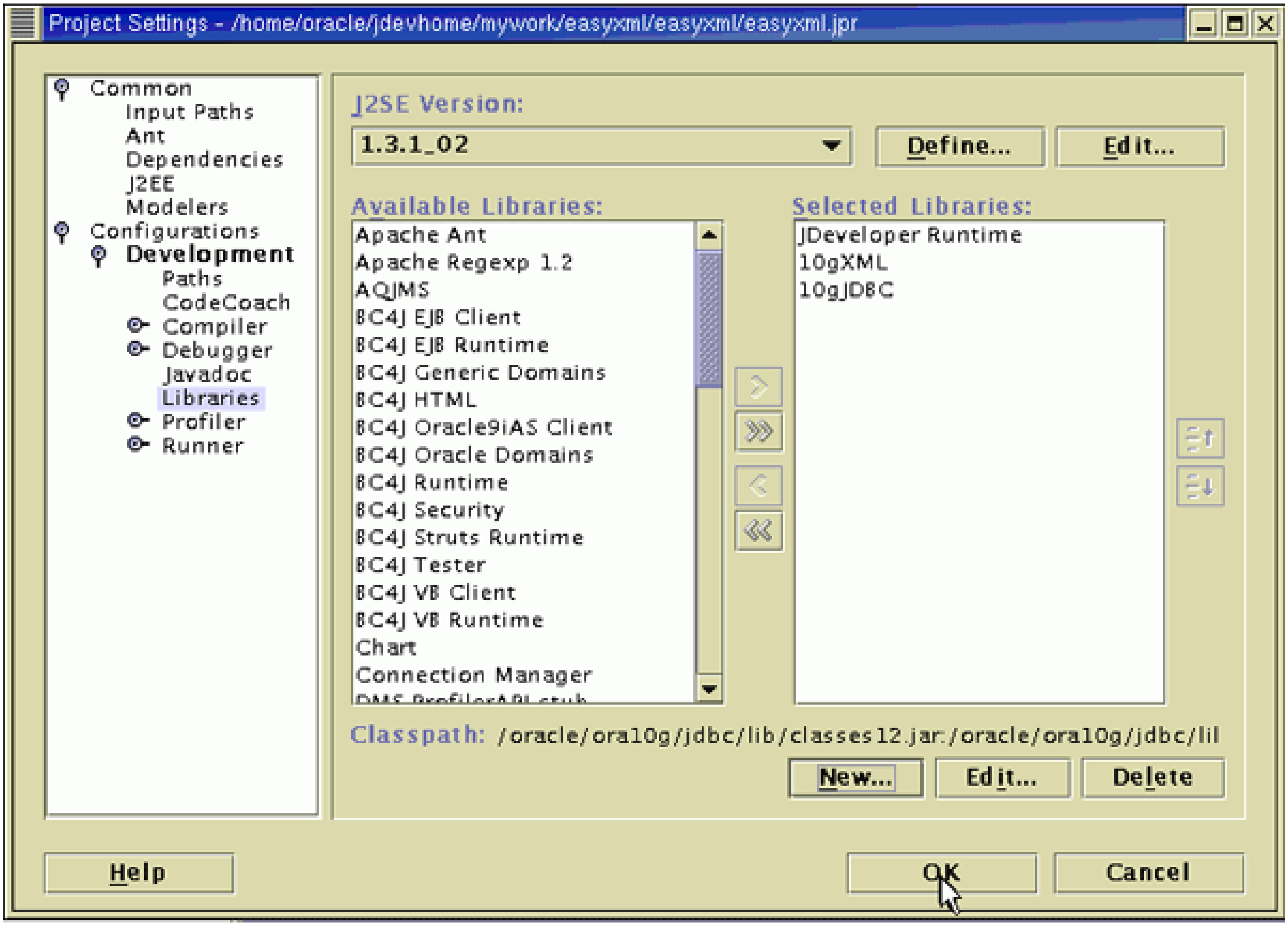


Library Name: 10gJDBC

Location: User Libraries

Class Path: /classes12.jar;/oracle/ora10g/jdbc/lib/nls_charset12.jar

☐ **16.** Click **OK** to close the Project Settings window.



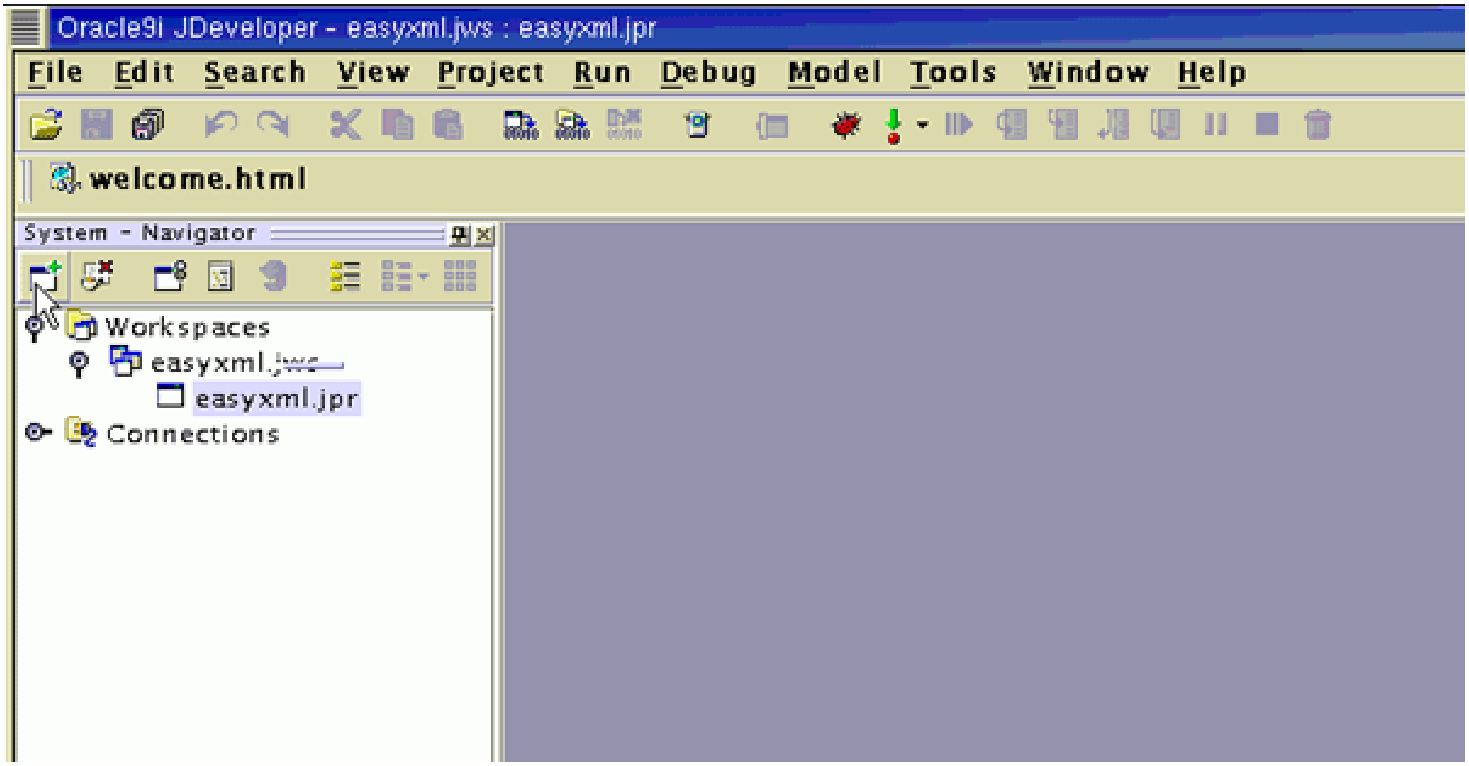
Retrieving Database Data as an XML Document

Back to List

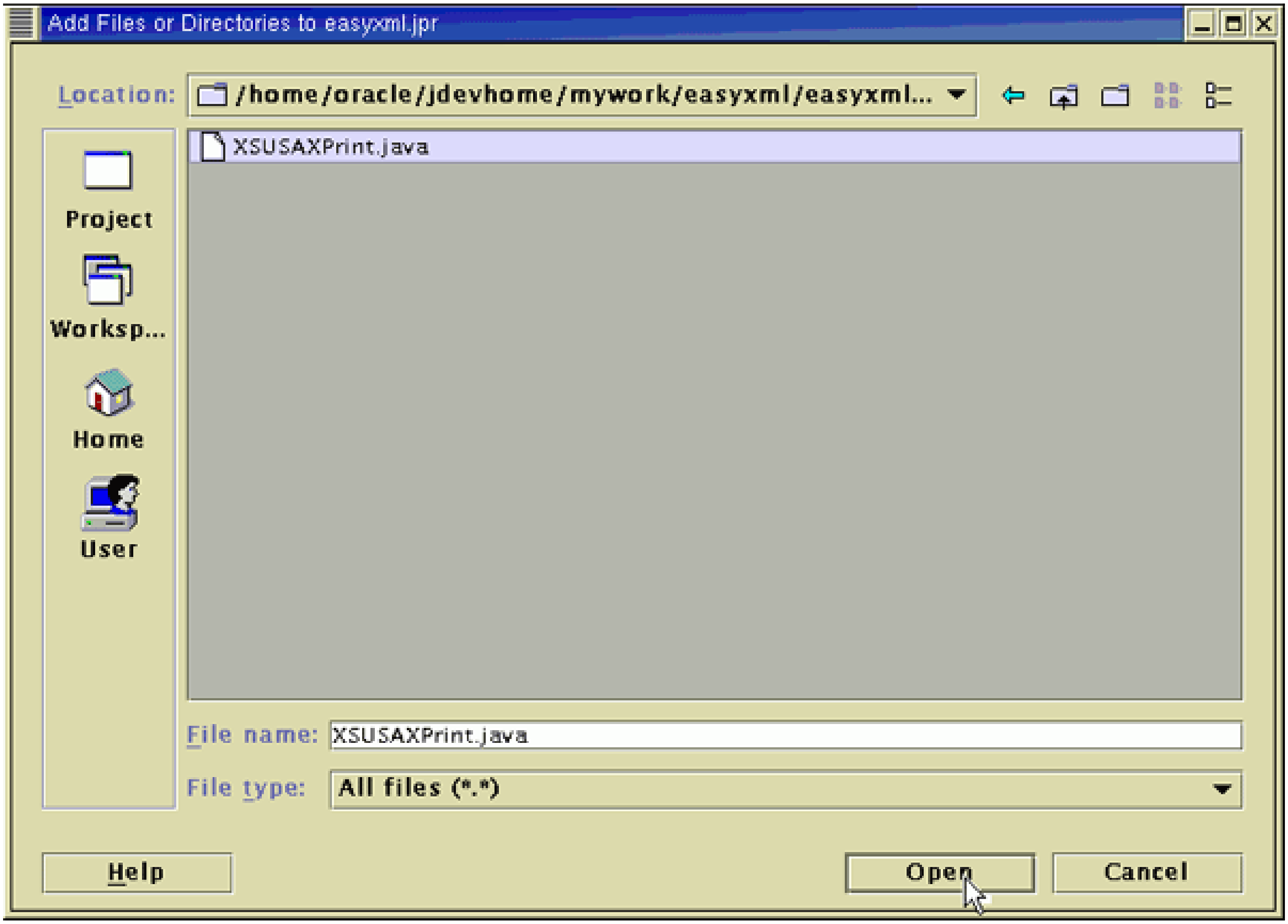
You can not retrieve the database data as an XML Document by performing the following:

☐ **1.** Download the example source file available here . Unzip it into the easyxml project directory **`/home/oracle/jdevhome/easyxml/easyxml`** .

☐ **2.** In JDeveloper select the **easyxml** project in the navigator and press the add file button in the System Navigator pane.


Oracle9i JDeveloper - easyxml.jws : easyxml.jpr
File Edit Search View Project Run Debug Model Tools Window Help
welcome.html
System - Navigator
Workspaces
easyxml.jws
easyxml.jpr
Connections

☐ **3.** Browse to the **`src/oracle/xml/sample`** directory and add the **`XSUSAXPrint.java`** file to the project.



Location: /home/oracle/jdevhome/mywork/easyxml/easyxml...

XSUSAXPrint.java

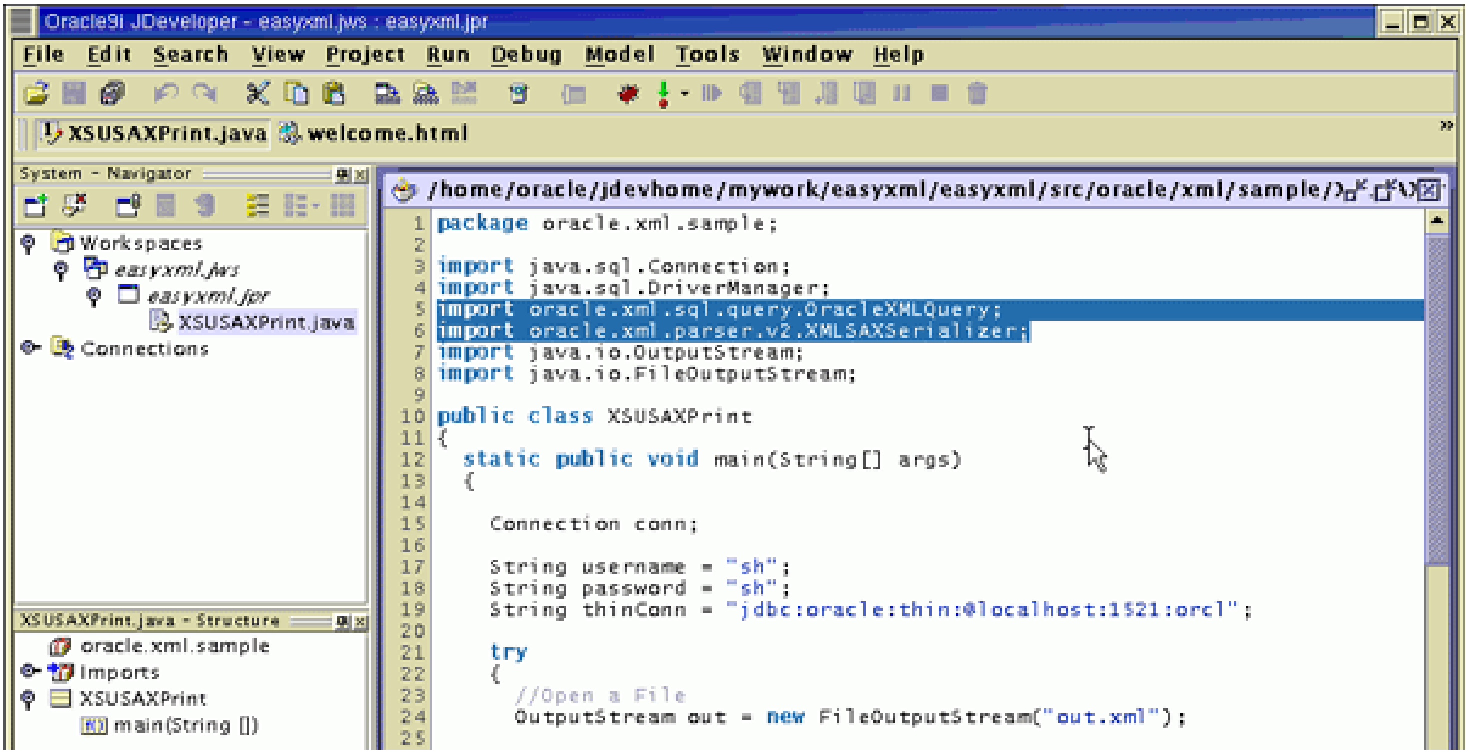
File name: XSUSAXPrint.java

☐ **4.** You can now look at the source code and see how the database data is retrieved by SAX streaming. You need to make sure that the following import statements are included:

import oracle.xml.parser.v2.XMLSAXSerializer
(SAX output)

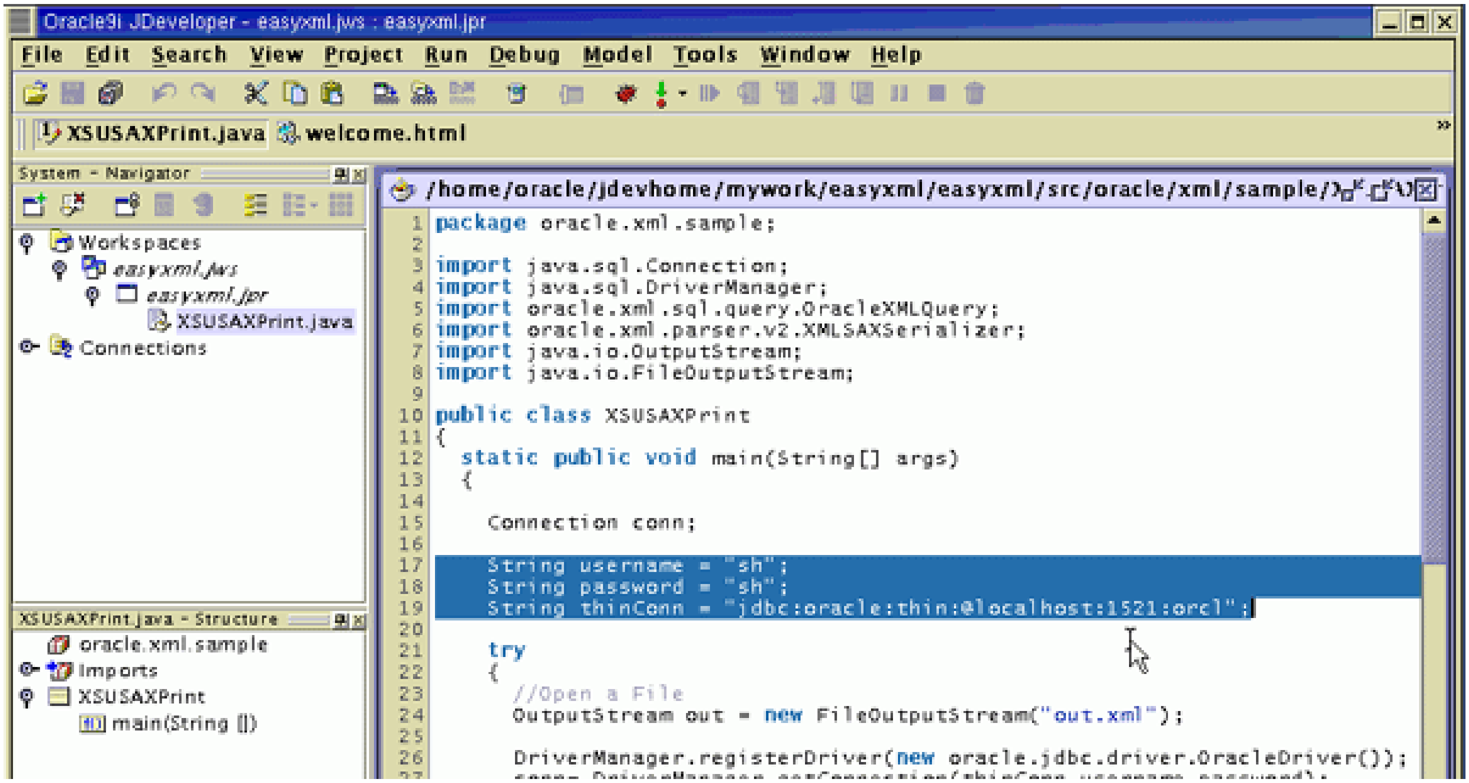
import

oracle.xml.sql.query.OracleXMLQuery
(XQL utility queries)


Oracle9i JDeveloper - easyxml.jws : easyxml.jpr
File Edit Search View Project Run Debug Model Tools Window Help
XSUSAXPrint.java
welcome.html
System - Navigator
Workspaces
easyxml.jws
easyxml.jpr
XSUSAXPrint.java
Connections
XSUSAXPrint.java - Structure
oracle.xml.sample
Imports
XSUSAXPrint
main(String [])
/home/oracle/jdevhome/mywork/easyxml/easyxml/src/oracle/xml/sample/

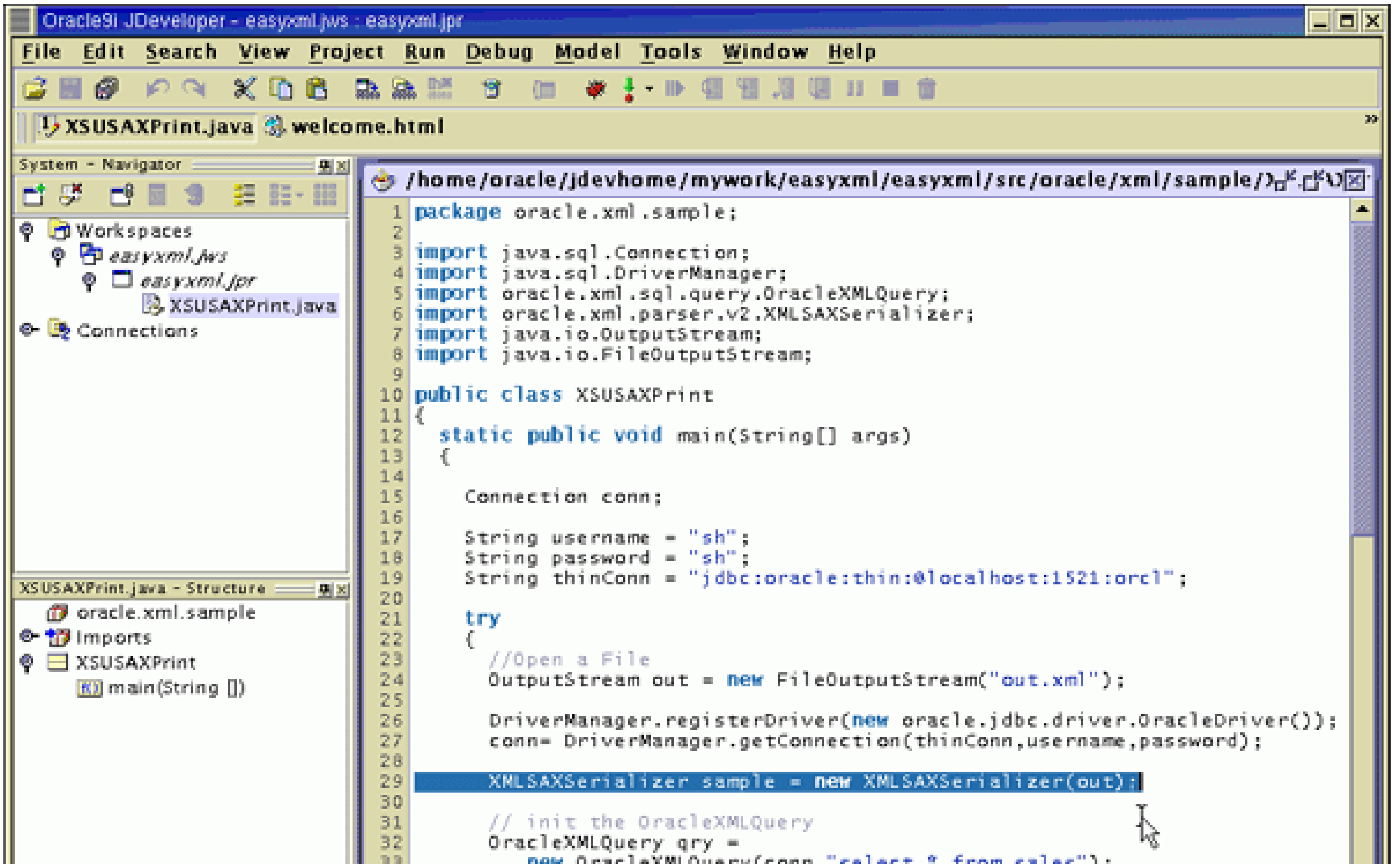
```
package oracle.xml.sample;

import java.sql.Connection;
import java.sql.DriverManager;
import oracle.xml.sql.query.OracleXMLQuery;
import oracle.xml.parser.v2.XMLSAXSerializer;
import java.io.OutputStream;
import java.io.FileOutputStream;

public class XSUSAXPrint
{
  static public void main(String[] args)
  {

    Connection conn;

    String username = "sh";
    String password = "sh";
    String thinConn = "jdbc:oracle:thin:@localhost:1521:orcl";

    try
    {
      //Open a File
      OutputStream out = new FileOutputStream("out.xml");

```

☐ **5.** You need to set up the JDBC connection to the database. In this example, you will connect to the SH sample schema assuming the password for SH is "SH".


Oracle9i JDeveloper - easyxml.jws : easyxml.jpr
File Edit Search View Project Run Debug Model Tools Window Help
XSUSAXPrint.java
welcome.html
System - Navigator
Workspaces
easyxml.jws
easyxml.jpr
XSUSAXPrint.java
Connections
XSUSAXPrint.java - Structure
oracle.xml.sample
Imports
XSUSAXPrint
main(String [])
/home/oracle/jdevhome/mywork/easyxml/easyxml/src/oracle/xml/sample/

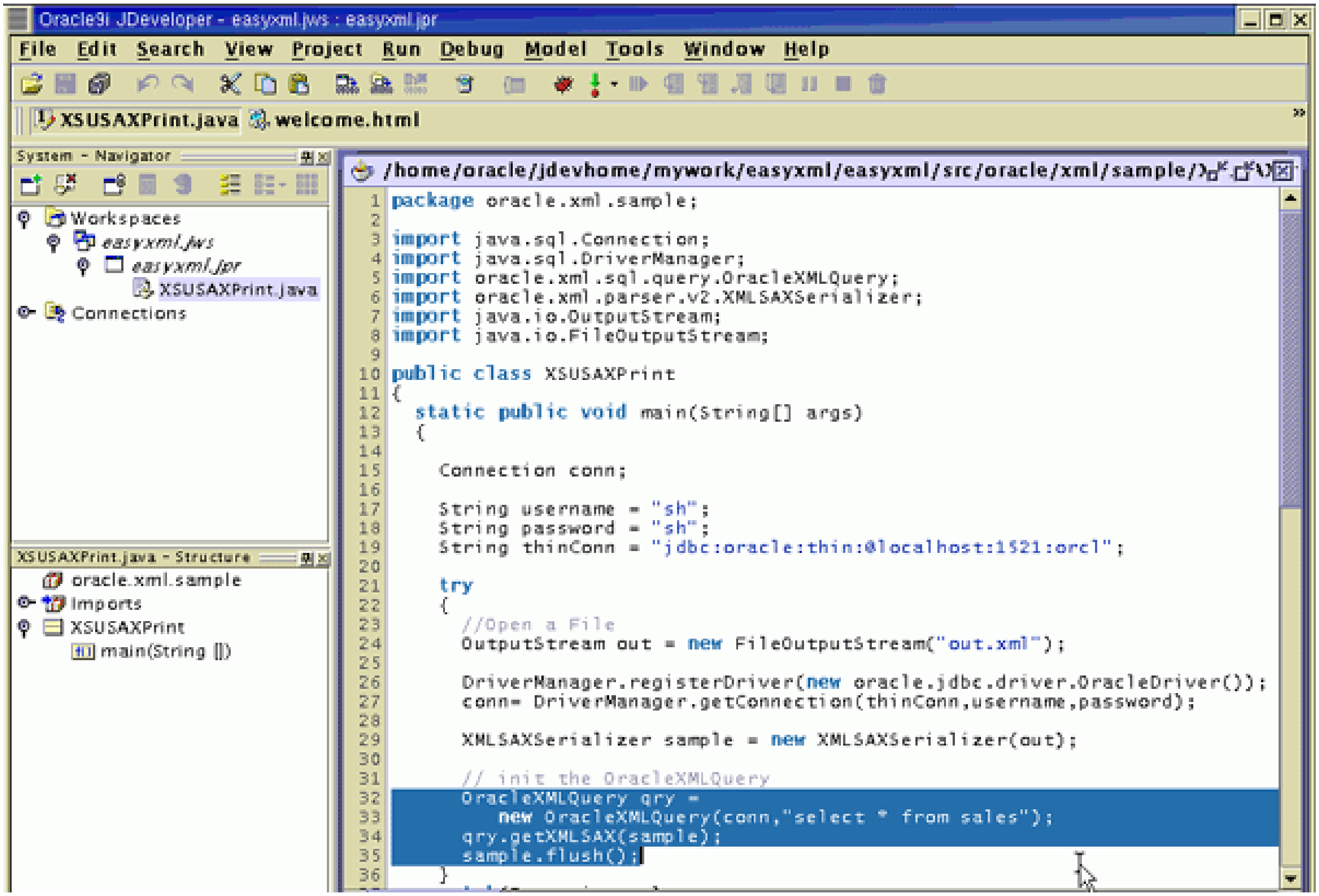
```
package oracle.xml.sample;

import java.sql.Connection;
import java.sql.DriverManager;
import oracle.xml.sql.query.OracleXMLQuery;
import oracle.xml.parser.v2.XMLSAXSerializer;
import java.io.OutputStream;
import java.io.FileOutputStream;

public class XSUSAXPrint
{
  static public void main(String[] args)
  {

    Connection conn;

    String username = "sh";
    String password = "sh";
    String thinConn = "jdbc:oracle:thin:@localhost:1521:orcl";

    try
    {
      //Open a File
      OutputStream out = new FileOutputStream("out.xml");

      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
```

☐ **6.** You need to create an instance of the SAXSerializer.


Oracle9i JDeveloper - easyxml.jws : easyxml.jpr
File Edit Search View Project Run Debug Model Tools Window Help
XSUSAXPrint.java
welcome.html
System - Navigator
Workspaces
easyxml.jws
easyxml.jpr
XSUSAXPrint.java
Connections
XSUSAXPrint.java - Structure
oracle.xml.sample
Imports
XSUSAXPrint
main(String [])
/home/oracle/jdevhome/mywork/easyxml/easyxml/src/oracle/xml/sample/

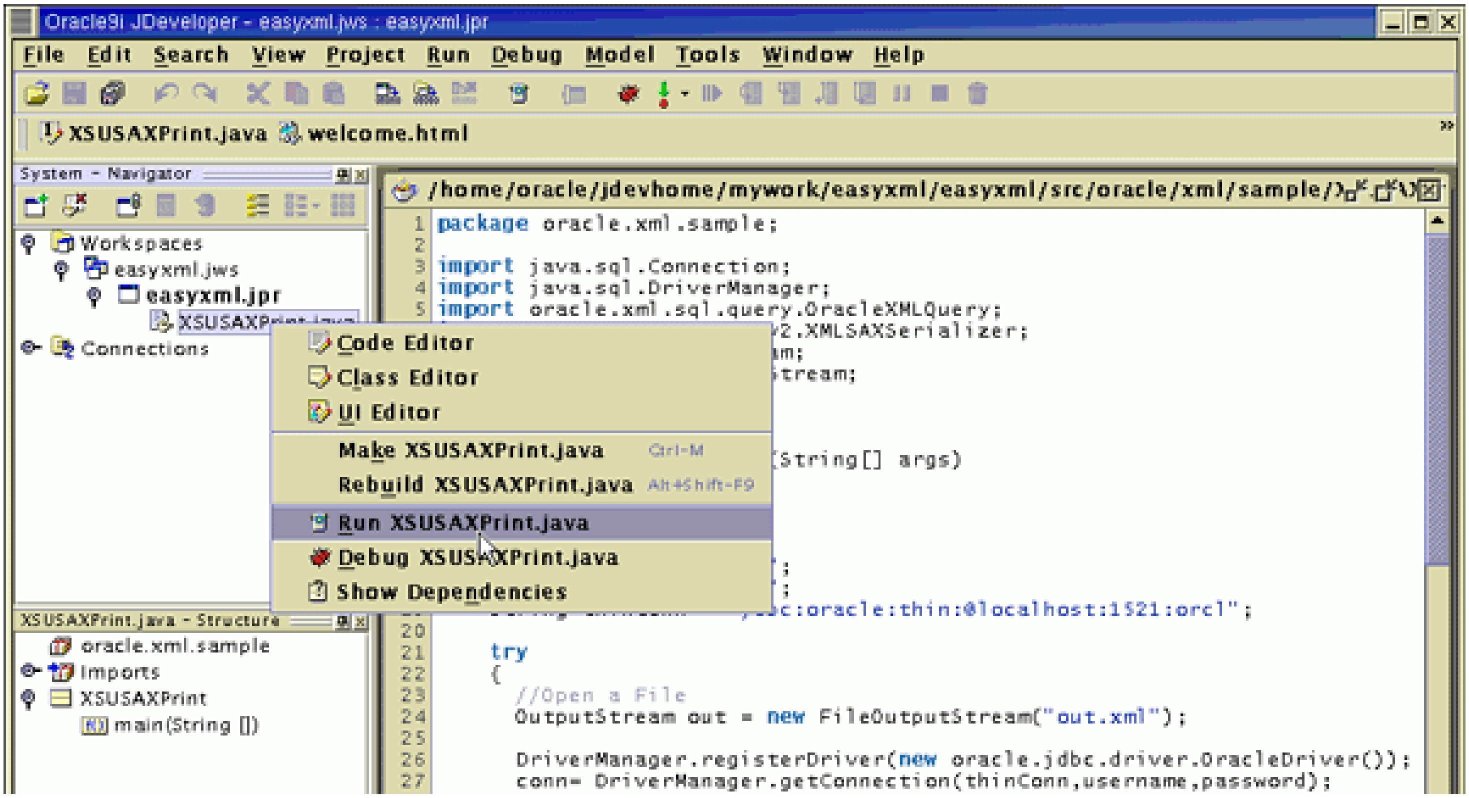
```
package oracle.xml.sample;

import java.sql.Connection;
import java.sql.DriverManager;
import oracle.xml.sql.query.OracleXMLQuery;
import oracle.xml.parser.v2.XMLSAXSerializer;
import java.io.OutputStream;
import java.io.FileOutputStream;

public class XSUSAXPrint
{
  static public void main(String[] args)
  {

    Connection conn;

    String username = "sh";
    String password = "sh";
    String thinConn = "jdbc:oracle:thin:@localhost:1521:orcl";

    try
    {
      //Open a File
      OutputStream out = new FileOutputStream("out.xml");

      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
      conn= DriverManager.getConnection(thinConn,username,password);

      XMLSAXSerializer sample = new XMLSAXSerializer(out);

      // init the OracleXMLQuery
      OracleXMLQuery qry =
         new OracleXMLQuery(conn,"select * from sales");
```

☐ **7.** To set up the XSU to send out SAX events you call the OracleXMLQuery.getXMLSAX() interface and pass in the SAX content handler as input. In this example, you pass in the SAXSerializer instance "sample".


Oracle9i JDeveloper - easyxml.jws : easyxml.jpr
File Edit Search View Project Run Debug Model Tools Window Help
XSUSAXPrint.java
welcome.html
System - Navigator
Workspaces
easyxml.jws
easyxml.jpr
XSUSAXPrint.java
Connections
XSUSAXPrint.java - Structure
oracle.xml.sample
Imports
XSUSAXPrint
main(String [])
/home/oracle/jdevhome/mywork/easyxml/easyxml/src/oracle/xml/sample/

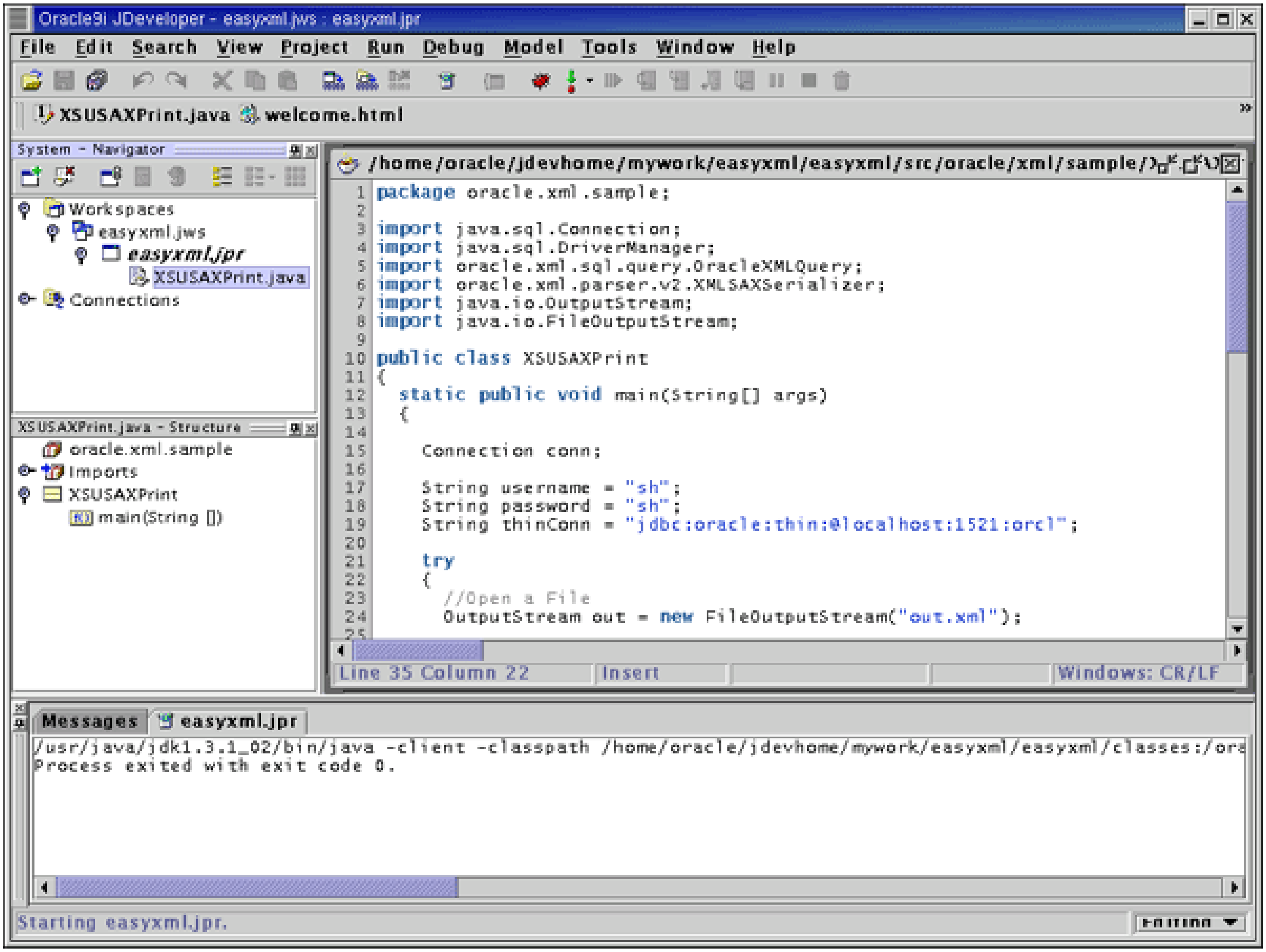
```
package oracle.xml.sample;

import java.sql.Connection;
import java.sql.DriverManager;
import oracle.xml.sql.query.OracleXMLQuery;
import oracle.xml.parser.v2.XMLSAXSerializer;
import java.io.OutputStream;
import java.io.FileOutputStream;

public class XSUSAXPrint
{
  static public void main(String[] args)
  {

    Connection conn;

    String username = "sh";
    String password = "sh";
    String thinConn = "jdbc:oracle:thin:@localhost:1521:orcl";

    try
    {
      //Open a File
      OutputStream out = new FileOutputStream("out.xml");

      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
      conn= DriverManager.getConnection(thinConn,username,password);

      XMLSAXSerializer sample = new XMLSAXSerializer(out);

      // init the OracleXMLQuery
      OracleXMLQuery qry =
         new OracleXMLQuery(conn,"select * from sales");
      qry.getXMLSAX(sample);
      sample.flush();
    }
```

☐ **8.** You are now ready to run the easyxml project. Right click on easyxml.jpr and select **Run XSUSAXPrint.java** .


Oracle9i JDeveloper - easyxml.jws : easyxml.jpr
File Edit Search View Project Run Debug Model Tools Window Help
XSUSAXPrint.java
welcome.html
System - Navigator
Workspaces
easyxml.jws
easyxml.jpr
XSUSAXPrint.java
Connections
Code Editor
Class Editor
UI Editor
Make XSUSAXPrint.java Ctrl-M
Rebuild XSUSAXPrint.java Alt+Shift-F9
Run XSUSAXPrint.java
Debug XSUSAXPrint.java
Show Dependencies
XSUSAXPrint.java - Structure
oracle.xml.sample
Imports
XSUSAXPrint
main(String [])
/home/oracle/jdevhome/mywork/easyxml/easyxml/src/oracle/xml/sample/

```
package oracle.xml.sample;

import java.sql.Connection;
import java.sql.DriverManager;
import oracle.xml.sql.query.OracleXMLQuery;
                                          v2.XMLSAXSerializer;
                                          am;
                                          tream;



                                          (String[] args)



                                          ;
                                          ;
                                          jdbc:oracle:thin:@localhost:1521:orcl";

    try
    {
      //Open a File
      OutputStream out = new FileOutputStream("out.xml");

      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
      conn= DriverManager.getConnection(thinConn,username,password);
```

☐ **8.** You should receive an exit code 0 in the log window.



```
package oracle.xml.sample;

import java.sql.Connection;
import java.sql.DriverManager;
import oracle.xml.sql.query.OracleXMLQuery;
import oracle.xml.parser.v2.XMLSAXSerializer;
import java.io.OutputStream;
import java.io.FileOutputStream;

public class XSUSAXPrint
{
  static public void main(String[] args)
  {

    Connection conn;

    String username = "sh";
    String password = "sh";
    String thinConn = "jdbc:oracle:thin:@localhost:1521:orcl";

    try
    {
      //Open a File
      OutputStream out = new FileOutputStream("out.xml");
```

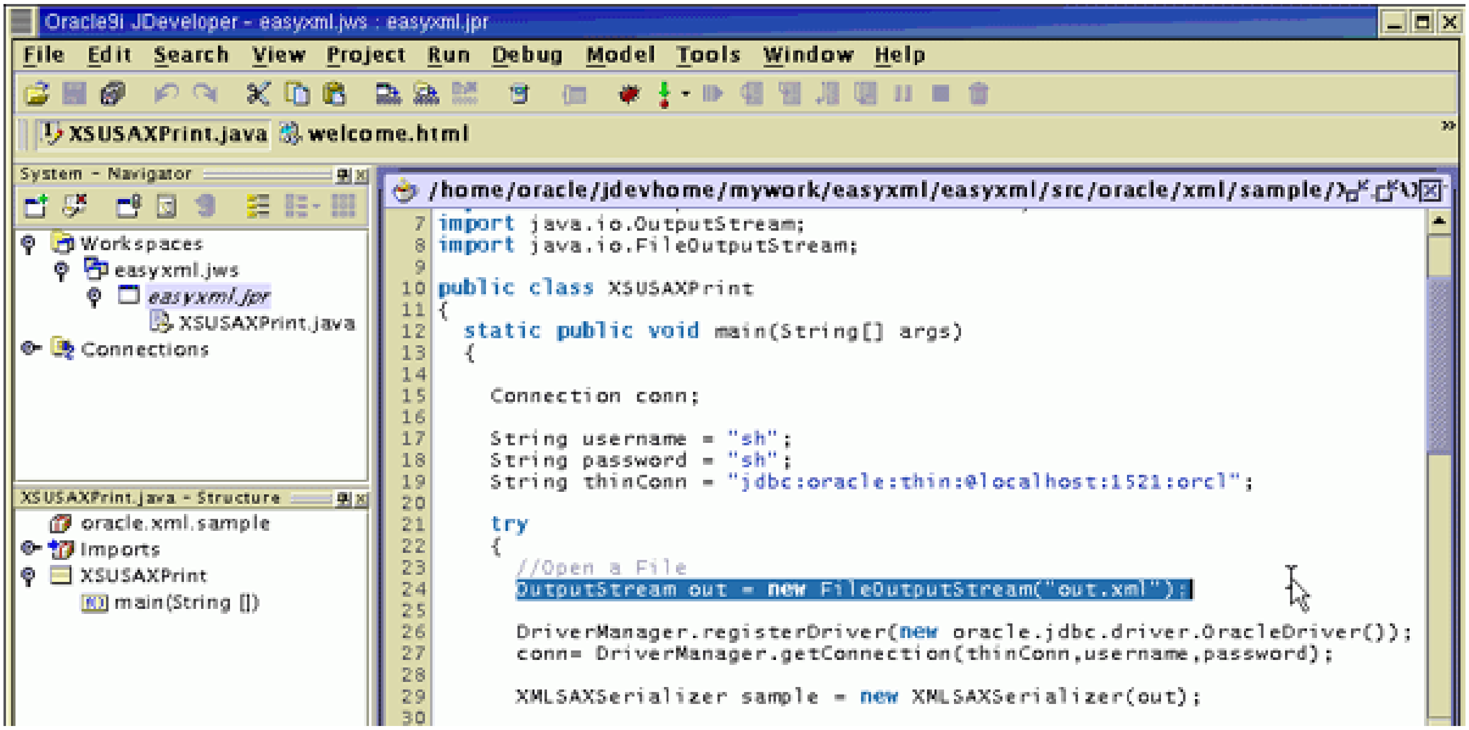
```
/usr/java/jdk1.3.1_02/bin/java -client -classpath /home/oracle/jdevhome/mywork/easyxml/easyxml/classes:/ora
Process exited with exit code 0.
```

Printing Database Data as an XML Document

Back to List

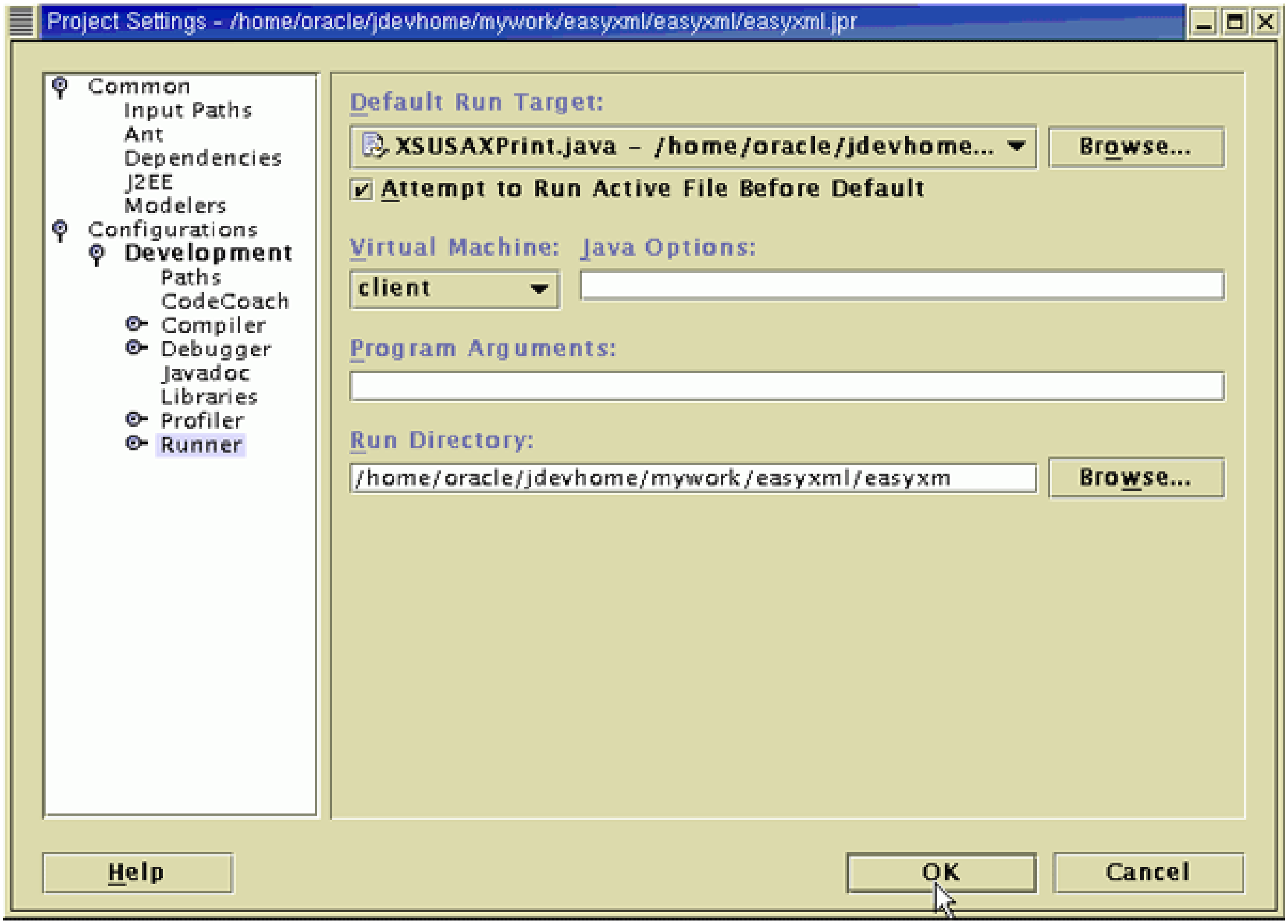
In this lesson, you will use the **SAXSerializer** to print out the query results to a file. However the **SAXSerializer** allows you to print out the SAX output into any output stream supported in Java. To print database data as an XML document, perform the following:

☐ **1.** You open a **FileOutputStream** and pass it to the **SAXSerializer.**


Oracle9i JDeveloper - easyxml.jws : easyxml.jpr
File Edit Search View Project Run Debug Model Tools Window Help
XSUSAXPrint.java
welcome.html
System - Navigator
Workspaces
easyxml.jws
easyxml.jpr
XSUSAXPrint.java
Connections
XSUSAXPrint.java - Structure
oracle.xml.sample
Imports
XSUSAXPrint
main(String [])
/home/oracle/jdevhome/mywork/easyxml/easyxml/src/oracle/xml/sample/

```
import java.io.OutputStream;
import java.io.FileOutputStream;

public class XSUSAXPrint
{
  static public void main(String[] args)
  {

    Connection conn;

    String username = "sh";
    String password = "sh";
    String thinConn = "jdbc:oracle:thin:@localhost:1521:orcl";

    try
    {
      //Open a File
      OutputStream out = new FileOutputStream("out.xml");

      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
      conn= DriverManager.getConnection(thinConn,username,password);

      XMLSAXSerializer sample = new XMLSAXSerializer(out);
```

☐ **2.** Right Click on the easyxml.jpr project and select **Project Settings** . Select the **Runner** option. Add Add the Run Directory **`/home/oracle/jdevhome/mywork/easyxml/easyxml`** . Then click **OK** .



Default Run Target: XSUSAXPrint.java – /home/oracle/jdevhome...

☑ Attempt to Run Active File Before Default

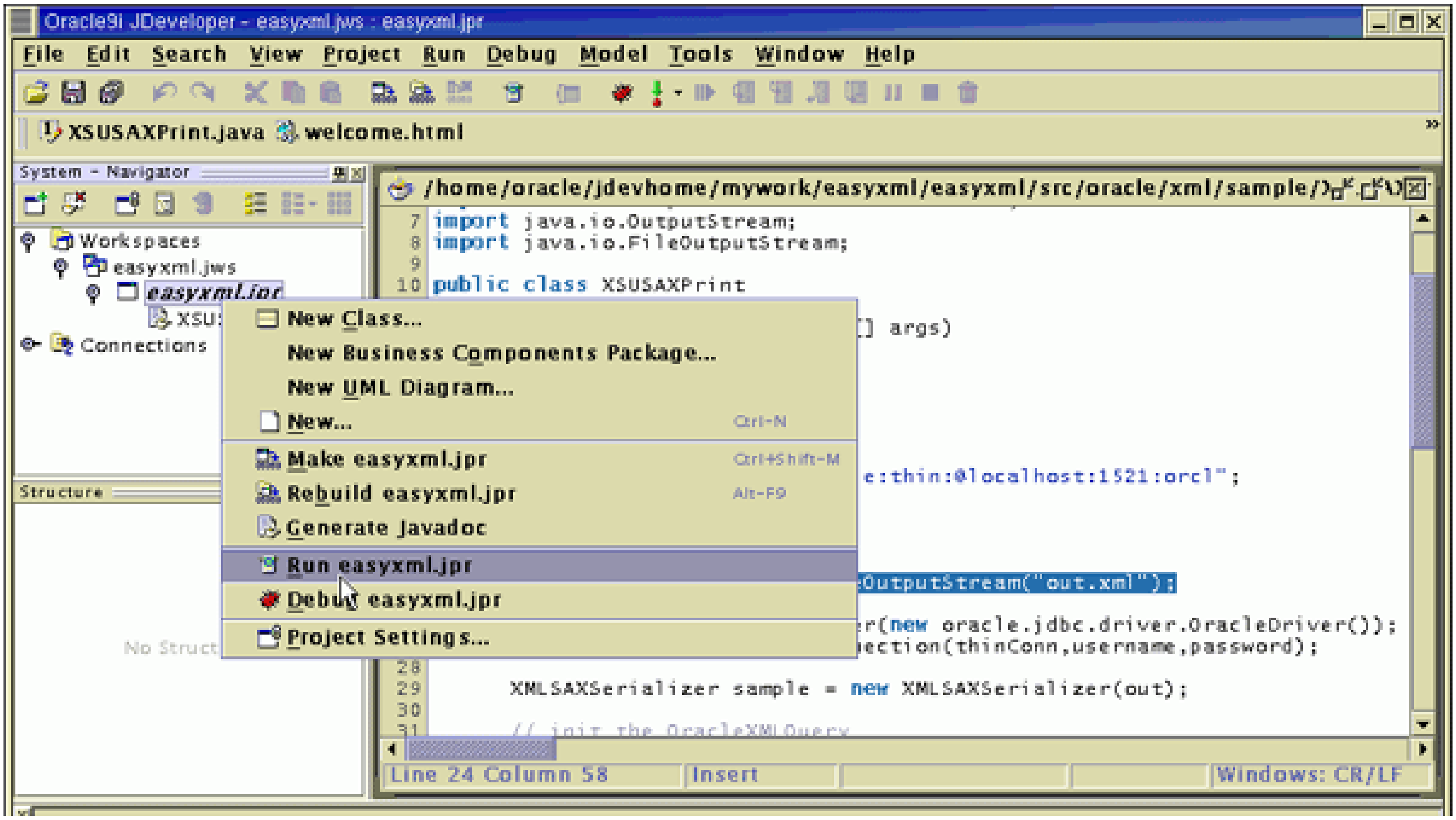
Virtual Machine: client

Java Options:

Program Arguments:

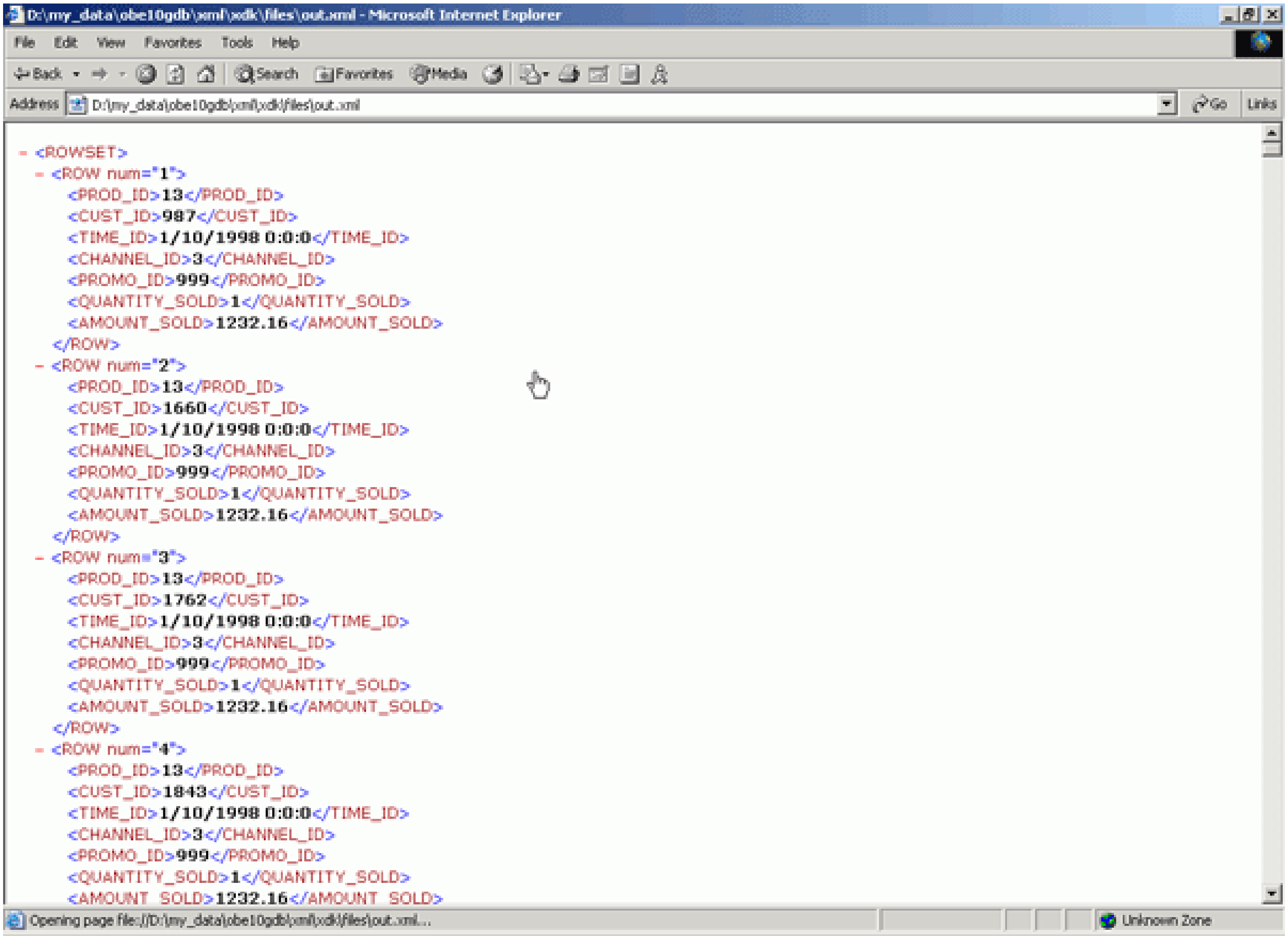
Run Directory: /home/oracle/jdevhome/mywork/easyxml/easyxm

☐ **3.** Run the program by right clicking the project and select **Run easyxml.jpr**


Oracle9i JDeveloper - easyxml.jws : easyxml.jpr
File Edit Search View Project Run Debug Model Tools Window Help
XSUSAXPrint.java
welcome.html
System - Navigator
Workspaces
easyxml.jws
easyxml.jpr
XSU
Connections
Structure
No Struct
/home/oracle/jdevhome/mywork/easyxml/easyxml/src/oracle/xml/sample/
New Class...
New Business Components Package...
New UML Diagram...
New... Ctrl-N
Make easyxml.jpr Ctrl+Shift-M
Rebuild easyxml.jpr Alt-F9
Generate Javadoc
Run easyxml.jpr
Debug easyxml.jpr
Project Settings...
Line 24 Column 58
Insert
Windows: CR/LF

```
import java.io.OutputStream;
import java.io.FileOutputStream;

public class XSUSAXPrint
                                        [] args)
                                        e:thin:@localhost:1521:orcl";
                                        OutputStream("out.xml");
                                        r(new oracle.jdbc.driver.OracleDriver());
                                        ection(thinConn,username,password);

        XMLSAXSerializer sample = new XMLSAXSerializer(out);

        // init the OracleXMLQuery
```

☐ **4.** Take a look at the Run Directory, you will see a large XML document out.xml in the directory. Open the file in your browser to review the contents.


D:\my_data\obe10gdb\xml\xdk\files\out.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address D:\my_data\obe10gdb\xml\xdk\files\out.xml
Go Links
Opening page file://D:\my_data\obe10gdb\xml\xdk\files\out.xml...
Unknown Zone

```
- <ROWSET>
  - <ROW num="1">
      <PROD_ID>13</PROD_ID>
      <CUST_ID>987</CUST_ID>
      <TIME_ID>1/10/1998 0:0:0</TIME_ID>
      <CHANNEL_ID>3</CHANNEL_ID>
      <PROMO_ID>999</PROMO_ID>
      <QUANTITY_SOLD>1</QUANTITY_SOLD>
      <AMOUNT_SOLD>1232.16</AMOUNT_SOLD>
    </ROW>
  - <ROW num="2">
      <PROD_ID>13</PROD_ID>
      <CUST_ID>1660</CUST_ID>
      <TIME_ID>1/10/1998 0:0:0</TIME_ID>
      <CHANNEL_ID>3</CHANNEL_ID>
      <PROMO_ID>999</PROMO_ID>
      <QUANTITY_SOLD>1</QUANTITY_SOLD>
      <AMOUNT_SOLD>1232.16</AMOUNT_SOLD>
    </ROW>
  - <ROW num="3">
      <PROD_ID>13</PROD_ID>
      <CUST_ID>1762</CUST_ID>
      <TIME_ID>1/10/1998 0:0:0</TIME_ID>
      <CHANNEL_ID>3</CHANNEL_ID>
      <PROMO_ID>999</PROMO_ID>
      <QUANTITY_SOLD>1</QUANTITY_SOLD>
      <AMOUNT_SOLD>1232.16</AMOUNT_SOLD>
    </ROW>
  - <ROW num="4">
      <PROD_ID>13</PROD_ID>
      <CUST_ID>1843</CUST_ID>
      <TIME_ID>1/10/1998 0:0:0</TIME_ID>
      <CHANNEL_ID>3</CHANNEL_ID>
      <PROMO_ID>999</PROMO_ID>
      <QUANTITY_SOLD>1</QUANTITY_SOLD>
      <AMOUNT_SOLD>1232.16</AMOUNT_SOLD>
```

Place the cursor on this icon to hide all screenshots.