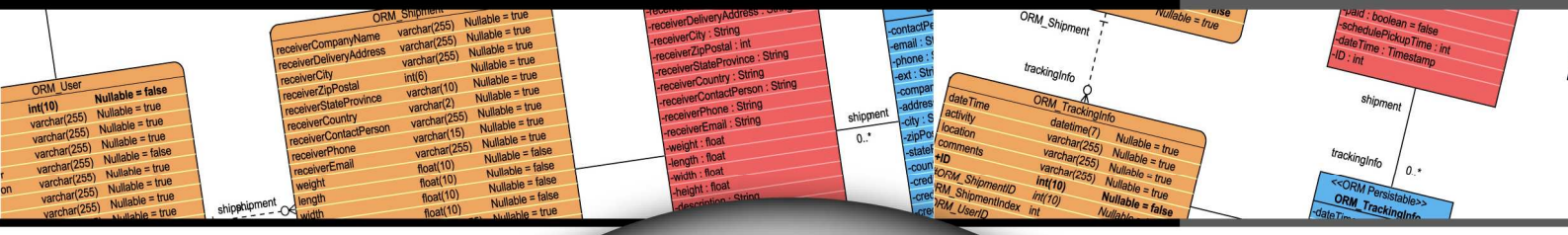
Visual Paradigm

Build Quality Applications Faster, Better and Cheaper

Database Visual ARCHITECT

Deployment Guide for Java

Access database with Object-Oriented technology



DB Visual ARCHITECT 4.0 Deployment Guide for Java

The software and documentation are furnished under the DB Visual ARCHITECT license agreement and may be used only in accordance with the terms of the agreement.

DB Visual ARCHITECT License Agreement

THE USE OF THE SOFTWARE LICENSED TO YOU IS SUBJECT TO THE TERMS AND CONDITIONS OF THIS SOFTWARE LICENSE AGREEMENT. BY INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE, YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUNDED BY ALL OF THE TERMS AND CONDITIONS OF THIS SOFTWARE LICENSE AGREEMENT.

1. **Limited License Grant.** Visual Paradigm grants to you ("the Licensee") a personal, non-exclusive, non-transferable, limited, perpetual, revocable license to install and use Visual Paradigm Products ("the Software" or "the Product"). The Licensee must not re-distribute the Software in whole or in part, either separately or included with a product.
2. **Restrictions.** The Software is confidential copyrighted information of Visual Paradigm, and Visual Paradigm and/or its licensors retain title to all copies. The Licensee shall not modify, adapt, decompile, disassemble, decrypt, extract, or otherwise reverse engineer the Software. Software may not be leased, rented, transferred, distributed, assigned, or sublicensed, in whole or in part. The Software contains valuable trade secrets. The Licensee promises not to extract any information or concepts from it as part of an effort to compete with the licensor, nor to assist anyone else in such an effort. The Licensee agrees not to remove, modify, delete or destroy any proprietary right notices of Visual Paradigm and its licensors, including copyright notices, in the Software.
3. **Disclaimer of Warranty.** The software and documentation are provided "AS IS," WITH NO WARRANTIES WHATSOEVER. ALL EXPRESS OR IMPLIED REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. THE ENTIRE RISK AS TO SATISFACTORY QUALITY, PERFORMANCE, ACCURACY AND EFFORT IS WITH THE LICENSEE. THERE IS NO WARRANTY THE DOCUMENTATION, Visual Paradigm's EFFORTS OR THE LICENSED SOFTWARE WILL FULFILL ANY OF LICENSEE'S PARTICULAR PURPOSES OR NEEDS. IF THESE WARRANTIES ARE UNENFORCEABLE UNDER APPLICABLE LAW, THEN Visual Paradigm DISCLAIMS SUCH WARRANTIES TO THE MAXIMUM EXTENT PERMITTED BY SUCH APPLICABLE LAW.
4. **Limitation of Liability.** Visual Paradigm AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY THE LICENSEE OR ANY THIRD PARTY AS A RESULT OF USING OR DISTRIBUTING SOFTWARE. IN NO EVENT WILL Visual Paradigm OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, EXEMPLARY, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF Visual Paradigm HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

5. **Termination.** The Licensee may terminate this License at any time by destroying all copies of Software. Visual Paradigm will not be obligated to refund any License Fees, if any, paid by the Licensee for such termination. This License will terminate immediately without notice from Visual Paradigm if the Licensee fails to comply with any provision of this License. Upon such termination, the Licensee must destroy all copies of the Software. Visual Paradigm reserves all rights to terminate this License.

SPECIFIC DISCLAIMER FOR HIGH-RISK ACTIVITIES. The SOFTWARE is not designed or intended for use in high-risk activities including, without restricting the generality of the foregoing, on-line control of aircraft, air traffic, aircraft navigation or aircraft communications; or in the design, construction, operation or maintenance of any nuclear facility. Visual Paradigm disclaims any express or implied warranty of fitness for such purposes or any other purposes.

NOTICE. The Product is not intended for personal, family or household use; rather, it is intended exclusively for professional use. Its utilization requires skills that differ from those needed to use consumer software products such as word processing or spreadsheet software.

GOVERNMENT RIGHTS. If the Software is licensed by or on behalf of a unit or agency of any government, the Licensee agrees that the Software is "commercial computer software", "commercial computer software documentation" or similar terms and that, in the absence of a written agreement to the contrary, the Licensee's rights with respect to the Software are limited by the terms of this Agreement.

Acknowledgements

Table of Contents

Chapter 1 - Deploying Standalone Java Application
Compiling using Javac ..... 1 -2
Compiling using Apache Ant ..... 1 -3
Compiling using Script ..... 1 -6
Executing application ..... 1 -8

Chapter 2 - Deploying Enterprise Java Web Application to BEA WebLogic
Introduction ..... 2 -2
Preparing to Deploy to WebLogic ..... 2 -2
Deploying Web Application to WebLogic 8.1 ..... 2 -3
Deploying Web Application to WebLogic 9.0 ..... 2 -4
Configuring Datasource on WebLogic 8.1 ..... 2 -5
Configuring Datasource on WebLogic 9.0 ..... 2 -9
Configuring DataSource Connection on DB-VA ..... 2 -14

Chapter 3 - Deploying Enterprise Java Web Application to IBM WebSphere
Introduction ..... 3 -2
Preparing to Deploy to WebSphere ..... 3 -2
Deploying Web Application to WebSphere ..... 3 -3
Configuring Datasource on WebSphere ..... 3 -6
Configuring the Datasource on DB-VA ..... 3 -12

Chapter 4 - Deploying Enterprise Web Application to IBM WebSphere Community Edition
Introduction ..... 4 -2
Preparing to Deploy to WebSphere Community Edition ..... 4 -2
Deploying Web Application to WebSphere Community Edition ..... 4 -3

Chapter 5 - Deploying Enterprise Java Web Application to JBoss
Introduction ..... 5 -2
Preparing to Deploy to JBoss ..... 5 -2
Deploying Web Application to JBoss ..... 5 -3
Configuring Datasource on JBoss ..... 5 -4
Configuring Datasource Connection in DB-VA ..... 5 -5

Chapter 6 - Deploying Enterprise Java Web Application to Oracle Application Server
Introduction ..... 6 -2
Preparing to Deploy to Oracle Application Server ..... 6 -2
Deploying Web Application to Oracle Application Server ..... 6 -3
Configuring Datasource on Oracle Application Server ..... 6 -6
Configuring Datasource Connection on DB-VA ..... 6 -7

1 Deploying Standalone Java Application

Chapter 1 - Deploying Standalone Java Application

DB Visual ARCHITECT (DB-VA) can generate all Java code for accessing database, you do not need to write any SQL to insert, query, update or delete the record, hence you can develop quality standalone Java application much faster, better and cheaper. In this chapter we focus on deploying standalone Java application. If you want to know how to develop a quality standalone Java application, you can refer to the Programmer's Guide for Java - Chapter 1 Developing Standalone Java Application.

In this chapter:

- Compiling using Javac
- Compiling using Apache Ant
- Compiling using Script
- Executing Application

Compiling using Javac

After you have developed a standalone Java application in Eclipse or other tools, you may wonder how you can execute it. First of all, you need to compile all the classes in your Java application and then execute the class that contains the main method. In the following step, we assume you have installed Java SDK and make sure you have modified your PATH environment variable to include JAVA SDK bin directory. In the following steps of compile and execute, we will use the example in the Programmer's Guide for Java - Chapter 4 Developing Standalone Java Application.

1. When DB-VA generates Java codes, "**classes**" , "**lib**" and "**src**" folders are created. The classes folder is empty, the lib folder contains the persistent library (orm.jar), and the src folder contains the configuration files and ORM-Persistable Java Classes. You can develop your code inside the src folder.

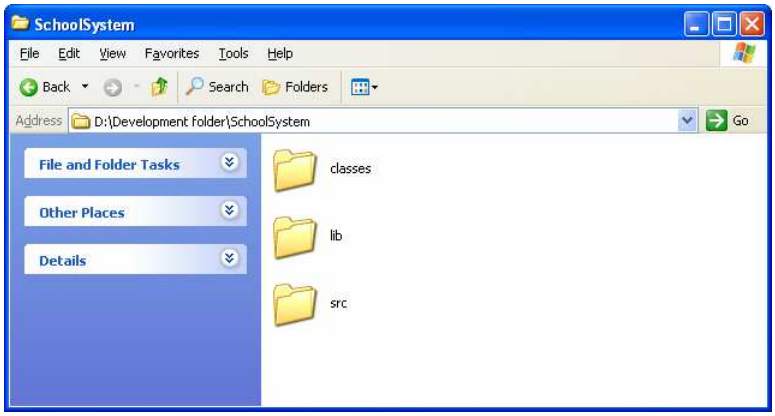


Figure 1.1 - The generated Source code

2. Use the "javac" command to compile .java files inside the src folder into intermediate bytecode files.

   Example of compiling the School System:

```
D:\Development folder\SchoolSystem>javac -d classes -cp lib\orm.jar src\school\*
.java src\system\*.java src\system\dialog\*.java src\ormsamples\*.java
```

Figure 1.2 - The compile script

```
javac - d classes - cp lib\orm.jar src\school\*.java src\system\*.java
src\system\dialog\*.java src\ormsamples\*.java
```

-d - specify where to place the generated class files

-cp - specify where to find the user class files

The orm.jar must be including in the classpath when compiling the application.

3. After executing the "javac" command, class files are generated in the "classes" folder.

Figure 1.3 - The compiled classes

Compiling using Apache Ant

DB-VA provides the option to generate the Apache Ant file for compiling the Java code and copy the configuration files to the classes folder. We assume that you have installed Ant.

1. From menu bar, select **Tools > Object Relational Mapping (ORM) > Generate Code ...**to open the Database Code Generation dialog box.

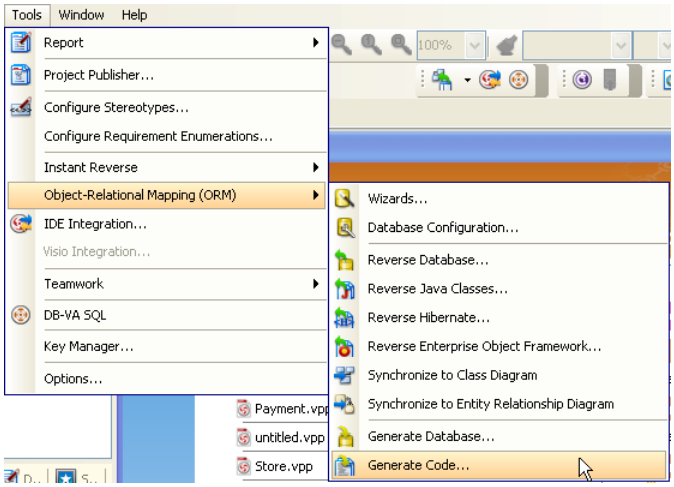


Figure 1.4 - To generate code

2. Select generate **Ant File** in **Code** tab.

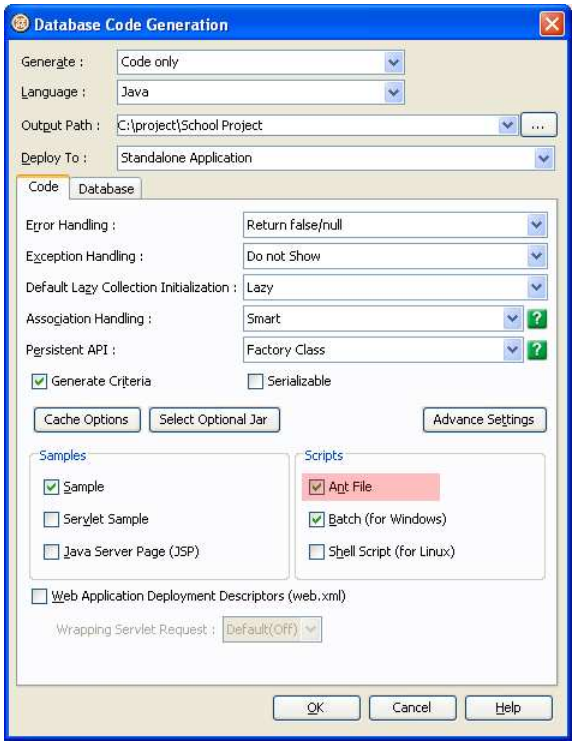


Figure 1.5 - Click the generate Ant file option

3. Click OK to generate the Java code and Ant file call **build-dbva.xml**.

   The sample of generated Ant file (build-dbva.xml):

```
<?xml version="1.0"?>
<!--
Licensee: Demo
License Type: Purchased
-->
<project name="SchoolSystem" default="compile">
	<description>SchoolSystem</description>

	<property name="src.dir" location="src" />
	<property name="lib.dir" location="lib" />
	<property name="classes.dir" location="classes" />
	<property name="javac.debug" value="on" />

	<target name="compile" description="Compile SchoolSystem">
		<mkdir dir="${classes.dir}" />
		<javac srcdir="${src.dir}" destdir="${classes.dir}"
		debug="${javac.debug}">
			<classpath>
				<fileset dir="${lib.dir}" includes="*.jar"/>
			</classpath>
		</javac>
		<copy todir="${classes.dir}">
			<fileset dir="${src.dir}">
				<exclude name="**/*.java" />
			</fileset>
		</copy>
	</target>

	<target name="clean" description="Clean SchoolSystem">
		<delete dir="${classes.dir}" />
	</target>

	<target name="createSchema" description="Run create data schema sample">
		<java classname="ormsamples.CreateSchoolSystemDatabaseSchema" fork="true">
			<classpath>
```

```
                        <fileset dir="${lib.dir}" includes="*.jar"/>
                        <pathelement location="${classes.dir}"/>
                    </classpath>
                </java>
            </target>

            <target name="createdata" description="Run create test data sample">
                <java classname="ormsamples.CreateSchoolSystemData" fork="true">
                    <classpath>
                        <fileset dir="${lib.dir}" includes="*.jar"/>
                        <pathelement location="${classes.dir}"/>
                    </classpath>
                </java>
            </target>

            <target name="retrievedata" description="Run retrieve data sample">
                <java classname="ormsamples.RetrieveSchoolSystemData" fork="true">
                    <classpath>
                        <fileset dir="${lib.dir}" includes="*.jar"/>
                        <pathelement location="${classes.dir}"/>
                    </classpath>
                </java>
            </target>

            <target name="deletedata" description="Run delete data sample">
                <java classname="ormsamples.DeleteSchoolSystemData" fork="true">
                    <classpath>
                        <fileset dir="${lib.dir}" includes="*.jar"/>
                        <pathelement location="${classes.dir}"/>
                    </classpath>
                </java>
            </target>

            <target name="dropschema" description="Run drop schema sample">
                <java classname="ormsamples.DropSchoolSystemDatabaseSchema" fork="true">
                    <classpath>
                        <fileset dir="${lib.dir}" includes="*.jar"/>
                        <pathelement location="${classes.dir}"/>
                    </classpath>
                </java>
            </target>

            <target name="listdata" description="Run list data sample">
                <java classname="ormsamples.ListSchoolSystemData" fork="true">
                    <classpath>
                        <fileset dir="${lib.dir}" includes="*.jar"/>
                        <pathelement location="${classes.dir}"/>
                    </classpath>
                </java>
            </target>
        </project>
```

The Ant file contains different targets to perform different tasks for the project such as compile the source code, clean the generated bytecode class files in the classes folder, and execute the sample generated by DB-VA.

4. Use Command Prompt to go to the location of the build-dbva.xml file.

   Type **ant - version** to confirm the Ant can be executed in Command Prompt.

Figure 1.6 - The result of execute "ant -version" command

5. Type **ant - file build-dbva.xml clean** to delete all the bytecode class file in classes folder.

```
D:\project\School Project>ant -file build-dbva.xml clean
Buildfile: build-dbva.xml

clean:
   [delete] Deleting directory D:\project\School Project\classes

BUILD SUCCESSFUL
Total time: 0 seconds
```

Figure 1.7 - The result of execute "ant - file build-dbva.xml clean" command

6. Type **ant - file build-dbva.xml compile** to compile all the source code in src folder.

```
D:\project\School Project>ant -file build-dbva.xml compile
Buildfile: build-dbva.xml

compile:
    [mkdir] Created dir: D:\project\School Project\classes
    [javac] Compiling 30 source files to D:\project\School Project\classes
    [javac] Note: Some input files use unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.
     [copy] Copying 5 files to D:\project\School Project\classes

BUILD SUCCESSFUL
Total time: 2 seconds
```

Figure 1.8 - The result of execute "ant - file build-dbva.xml compile" command

Compiling using Script

DB-VA not only generates the Ant File for compiling source code and executing sample, it can also generate the batch and shell script file.

Batch file - a file that allow MS-DOS and Microsoft Windows user to create a lists of command and/or programs to run once the batch file is executed. Batch file refer to scripts written for DOS and Windows.

Shell Script - a file containing operating system commands that are processed in a batch method, one at a time, until complete. Shell Script refers to scripts written for Unix shell.

1. Select generate **Batch** and **Shell Script** in **Code** tab of Database Code Generation dialog box.

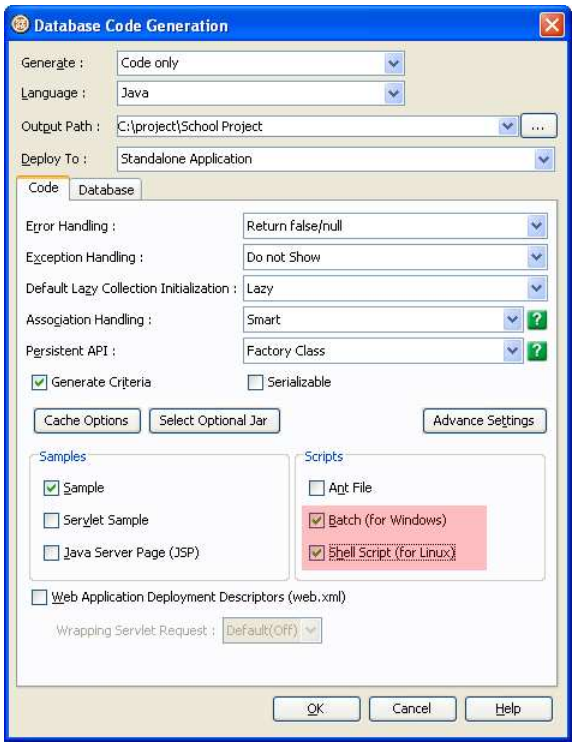


Figure 1.9 - Generate Batch and shell script options

2. The Batch and Shell Script files are generated at the Output Path. The Batch and Shell Script can be used to compile Java source and execute the samples generated by DB-VA.

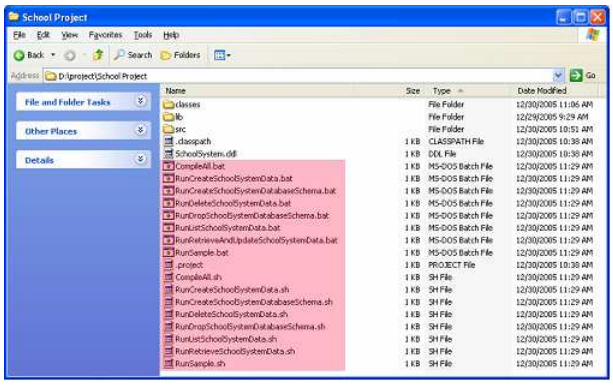


Figure 1.10 - The generated script files

3. The CompileAll.bat and CompileAll.sh is used for compile the source code generated by DB-VA.
4. Use Command Prompt to go to the Output Path location.

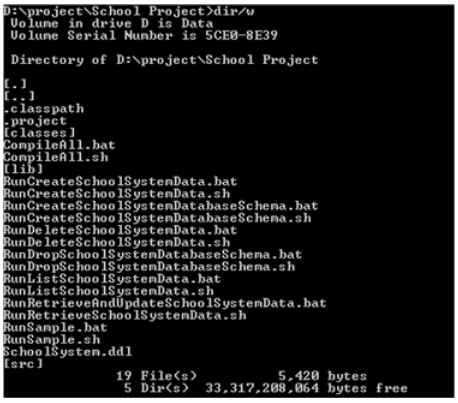


Figure 1.11 - The directory of the script files

5. In Windows, type CompileAll.bat to compile all the source code in the src folder and copy the configure files to the classes folder.

Figure 1.12 - Execute the script file

Executing application

After you have compiled the java files to class files, you still cannot execute the application yet because the configuration and mapping files for connecting the database is missing.

1. Copy the "**ormmapping**" folder in "**src**" folder to "**classes**" folder. The "**ormmapping**" folder contains the configuration files and the mapping files of the classes with database.

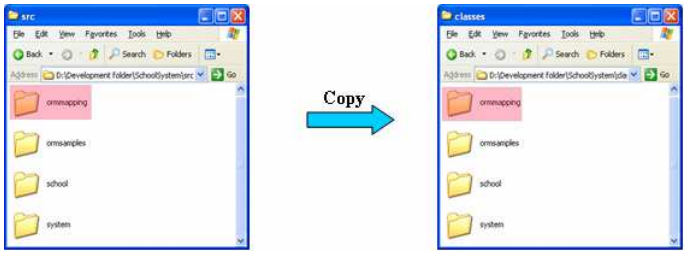


Figure 1.13 - copy the ormmapping folder to classes folder

2. Use the "**java**" command to execute the class which contains the "`public static void main (String args[])`" method.

   Example of executing School System:

   - **Start > Run** > type **cmd** and click **OK** to open the command prompt.
   - Go to the development directory.
   - Execute the java command

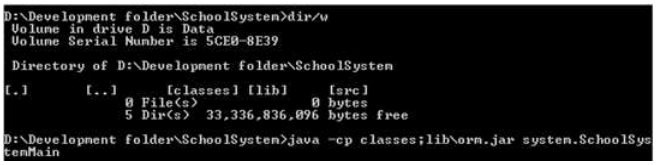


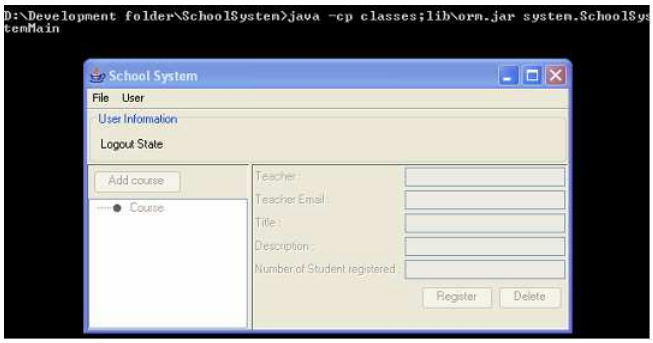
Figure 1.14 - Execute the sample program

```
>java - cp classes;lib\orm.jar system.SchoolSystemMain
```

"-cp" is a flag that tell the Java command to look for java class files.

system.SchoolSystemMain is the class that contains the "`public static void main (String args[])`" method.

3. The standalone application is executing now.



2 Deploying Enterprise Java Web Application to BEA WebLigic

Chapter 2 - Deploying Enterprise Java Web Application to BEA WebLogic

DB Visual ARCHITECT (DB-VA) provides different kinds of template for user to generate Java code. The template will optimize the configuration of generated Java Code and select jar files for different application server or standalone Java application. In this chapter, we will deploy enterprise Java web application to BEA WebLogic. When you deploy the web application on application server, you can select to use the JDBC or datasource to connect the database.

In this chapter:

- Introduction
- Preparing to Deploy to WebLogic 8.1/9.0
- Deploying Web Application to WebLogic 8.1
- Deploying Web Application to WebLogic 9.0
- Configuring Datasource on WebLogic 8.1
- Configuring Datasource on WebLogic 9.0
- Configuring Datasource Connection on DB-VA

Introduction

This document is based on the Programmer's Guide for Java - Chapter 3 Developing Java Enterprise Web Application example to demonstrate the deployment steps on a WebLogic Server. The Example of Programmer's Guide for java - Chapter 3 is deployed on a JBoss Server, we need to modify some configuration before deploy on a WebLogic Server. Finally, we will configure the web application to use the datasource connection provide by WebLogic server to connect to database.

Preparing to Deploy to WebLogic

Suppose you have downloaded the example of the Programmer's Guide for Java - Chapter 3 Developing Java Enterprise Web Application. You need to change the template of generate code before deploying on the WebLogic 8.1 Server.

1. From the menu bar, select **Tools > Object Relational Mapping (ORM) > Generate Code ...** to open Database Code Generation dialog box.

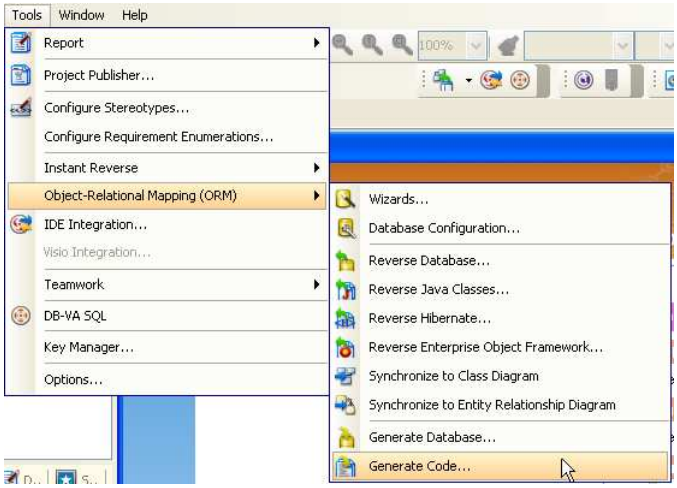


Figure 2.1 - To generate code

2. Change Deploy To option from **JBoss Application Server** to **WebLogic Application Server 8.1/9.0**

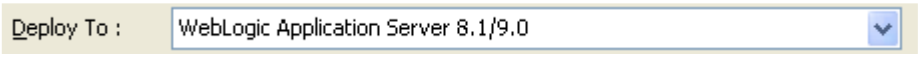


Figure 2.2 - Enter the deployment directory

DB-VA helps you to select the corresponding Optional Jar files and set datasource options.

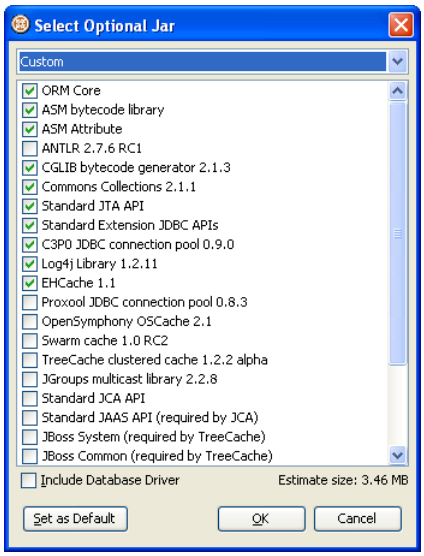


Figure 2.3 - Select Optional Jar dialog

3. Click **OK** to regenerate code.

Deploying Web Application to WebLogic 8.1

Now the orm.jar is updated and generate in JBoss Server deploy folder. You can copy the schoolsystem.war folder in JBoss Server deploy folder to the WebLogic 8.1 Server. The following is the steps to deploy on a WebLogic 8.1 Server.

1. Find the antlr.jar file from DB-VA installation Directory\ormlib folder.
2. Modify **startWebLogic.cmd** in WebLogic server domain (**WEBLOGIC _HOME\user_projects\domains\mydomain)** to add the antlr.jar to server classpath.

   The mydomain is a domain created by user to use Configuration Wizard provided by WebLogic.

   This sample adds antlr.jar to classpath:

```
set CLASSPATH=C:\bea\user_projects\domains\mydomain\antlr.jar;%WEBLOGIC_CLASSPATH%;
%POINTBASE_CLASSPATH%;%JAVA_HOME%\jre\lib\rt.jar;%WL_HOME%\server\lib\webservices.jar;
C:\bea\user_projects\domains\mydomain\applications\SchoolProject\lib\mysql.jar;%CLASSPA
TH%
```

3. Copy the schoolsystem.war folder from JBoss server's deploy folder to the WEBLOGIC_HOME\user_projects\domains\mydomain\applications folder.

Figure 2.4 - The deployment directory

4. Start WebLogic Server. **Start menu > All Program > BEA WebLogic Platform 8.1 > User Projects > mydomain > Start Server.**

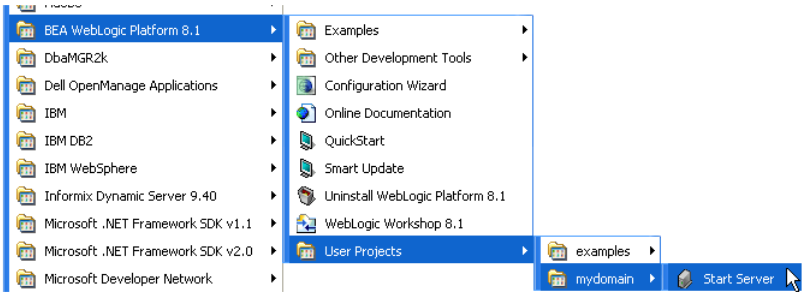


Figure 2.5 - To start the Application Server

5. Visit the School System application on WebLogic Server.

   http://localhost:7001/schoolsystem

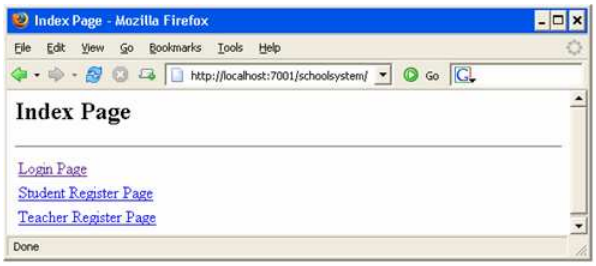


Figure 2.6 - the index page of the project

Deploying Web Application to WebLogic 9.0

1. Find **antlr.jar file** in the **DB-VA installation Directory\ormlib** folder.

   Modify setDomainEnv.cmd in WebLogic server domain "bin" folder (WEBLOGIC_HOME\user_projects\domains\base_domain\bin) and add the antlr.jar to classpath.

   The mydomain is a domain created by user to use Configuration Wizard provided by WebLogic. This sample adds antlr.jar to classpath:

```
set CLASSPATH=DBVA_HOME\ormlib\antlr.jar;%PRE_CLASSPATH%;%WEBLOGIC_CLASSPATH%;
%POST_CLASSPATH%;%WLP_POST_CLASSPATH%;%WL_HOME%\integration\lib\util.jar;
C:\bea9.0\user_projects\domains\base_domain\autodeploy\SchoolProject\lib\mysql.jar;
```

2. Copy the schoolsystem.war folder from JBoss server deploy folder to WEBLOGIC_HOME\user_projects\domains\base_domain\autodeploy folder.

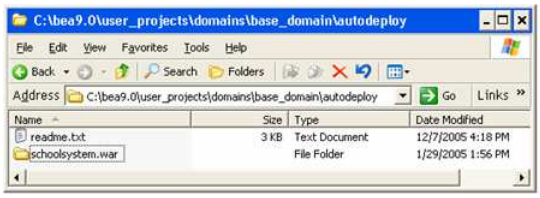


Figure 2.7 - The deployment directory

3. Start WebLogic 9.0 Server. Windows **Start > All Program > BEA Products > User Products > base_domain > Start Admin Server for WebLogic Server Domain**

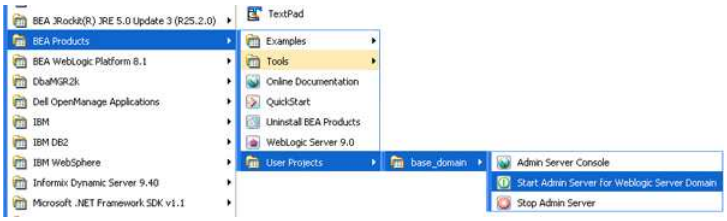


Figure 2.8 - To start the Application Server

4. Visit the SchoolProject application on WebLogic Server.

   http://localhost:7001/schoolsystem

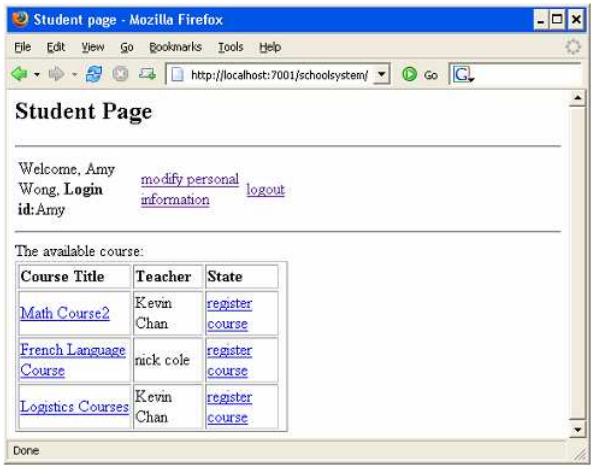


Figure 2.9 - The student Page

Configuring Datasource on WebLogic 8.1

WebLogic 8.1/9.0 server can provide a datasource for application to share the JDBC connection within the server. The following steps teach you how to configure datasource on WebLogic server.

1. Add the JDBC driver for the classpath of the startWebLogic.cmd in WebLogic server domain (**WEBLOGIC _HOME\user_projects\domains\mydomain**). You can find the JDBC driver in **schoolsystem.war\WEB-INF\lib**. In this example we use the MySQL database, so you need to get the mysql.jar JDBC driver in the lib folder.

   This Sample adds the JDBC driver to classpath:

```
set CLASSPATH= C:\bea\user_projects\domains\mydomain\antlr.jar;%WEBLOGIC_CLASSPATH%;
%POINTBASE_CLASSPATH%;%JAVA_HOME%\jre\lib\rt.jar;%WL_HOME%\server\lib\webservices.jar;
C:\bea\user_projects\domains\mydomain\applications\schoolsystem.war\lib\mysql.jar;%CLAS
SPATH%
```

2. Go to **WebLogic Server Administration Console** (http://localhost:7001/console/login/LoginForm.jsp), type in username and password to login.

Figure 2.10 - The Login page of BEA WebLogic

3. Select **mydomain > Services > JDBC > Connection Pools**

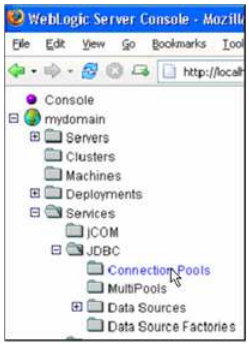


Figure 2.11 - Select the connection pool

4. Select **Configure a new JDBC Connection Pool**

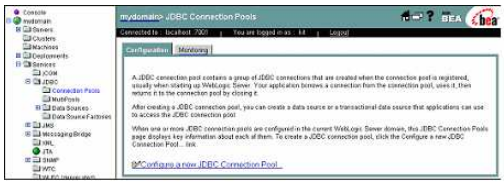


Figure 2.12 - Configure the connection pool

5. Select Database Type and Database Driver and click **Continue**

Database Type MySQL

Database Driver using com.mysql.jdbc.Driver

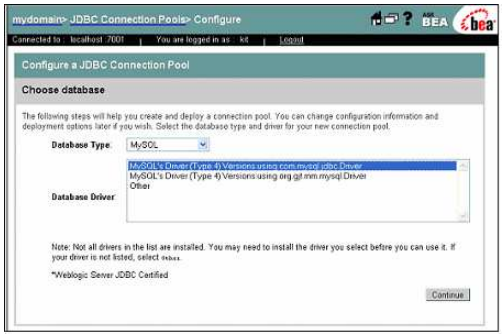


Figure 2.13 - Select the Database Driver

6. Fill in the Connection Properties and click **Continue**

localhost

Name	MySQL Connection Pool
Database Name	schoolproject
Host Name	
Port	3306
Database User Name	root

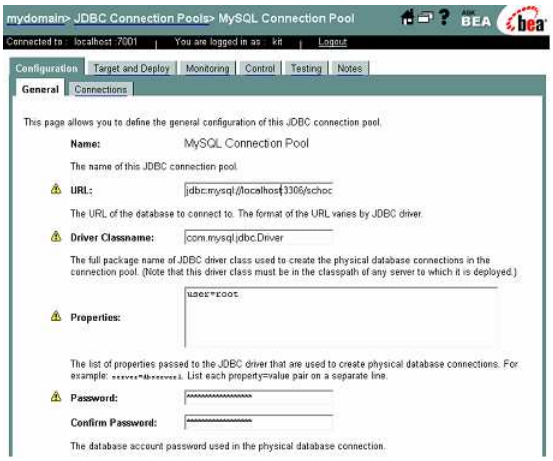


Figure 2.14 - Enter the Database Configuration

7. Click **Test Driver Configuration**. It will show the success message. Click **Create and Deploy** button to finish setup. Now the **MySQL Connection Pool** is created.

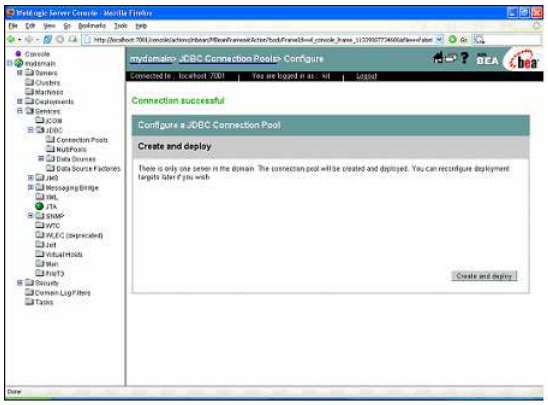


Figure 2.15 - The database connection successful message

8. Select the **mydomain > Services > JDBC > Data Sources**

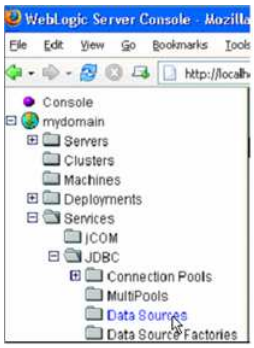


Figure 2.16 - Select the Data Sources

9. Click **Configure a new JDBC Data Sources**

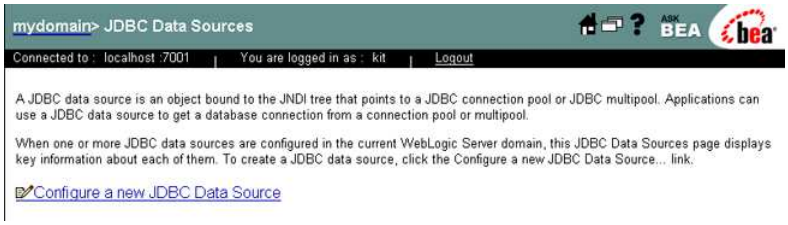


Figure 2.17 - Configuration a new JDBC Data Sources

10. Enter the Data Source information and click **Continue**

Name MySQL Data Source
JNDI Name app/schoolsystem
JNDI path bound Honor Global Transactions

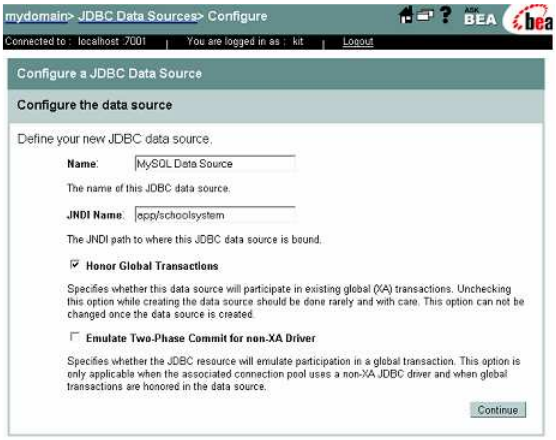


Figure 2.18 - Configure the Data Sources

11. Select the previously created **MySQL Connection Pool** and click **Continue**

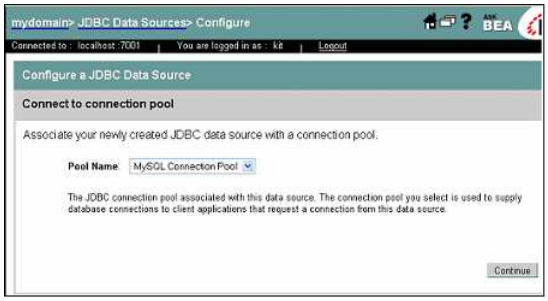


Figure 2.19 - Connect to the connection pool

12. Select this Data Source for **myserver**. Click **Create** to finish setup.

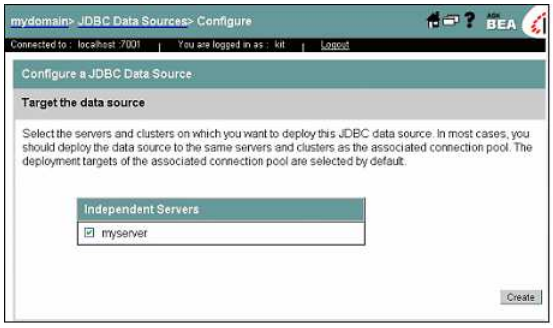


Figure 2.20 - Select the target Data Sources

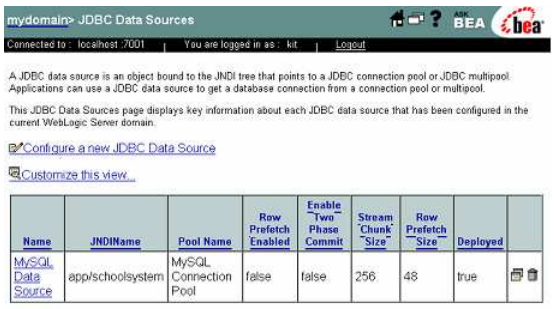


Figure 2.21 - Manage the Data Source

Configuring Datasource on WebLogic 9.0

1. Add the JDBC driver for the classpath of the **setDomainEnv.cmd** in WebLogic server domain "bin" folder (**WEBLOGIC_HOME\user_projects\domains\base_domain\bin**). In this example, we use the MySQL database; you can find the JDBC driver in **schoolsystem.war\WEB-INF\lib\mysql.jar**.

   This sample adds the JDBC driver to classpath:

   ```
   set CLASSPATH=DBVA_HOME\ormlib\antlr.jar;%PRE_CLASSPATH%;%WEBLOGIC_CLASSPATH%;
   %POST_CLASSPATH%;%WLP_POST_CLASSPATH%;%WL_HOME%\integration\lib\util.jar;
   C:\bea9.0\user_projects\domains\base_domain\autodeploy\schoolsystem.war\lib\mysql.jar;
   ```

2. Go to **WebLogic Server Administration Console** (http://localhost:7001/console/login/LoginForm.jsp), type in username and password to login.

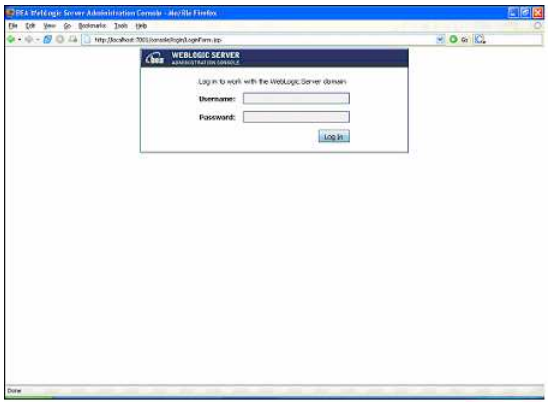


Figure 2.22 - The Login page

3. Before configure any setting, you need to click the **Lock & Edit** button to modify, add or delete items in this domain

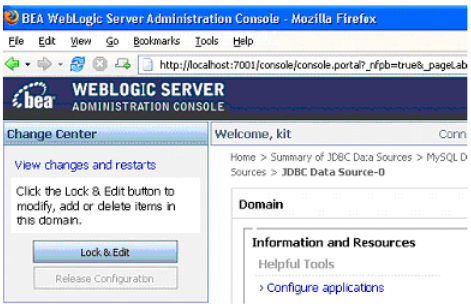


Figure 2.23 - Lock an Edit the site

4. Select **JDBC > Data Sources**

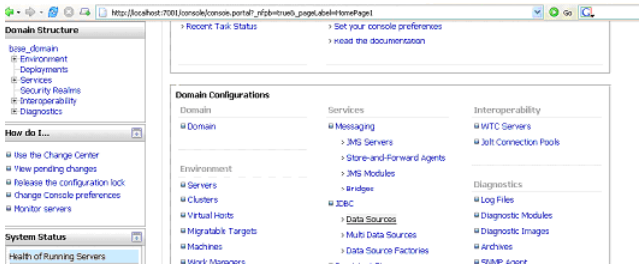


Figure 2.24 - Select Data Sources

5. Click **New** to create data source

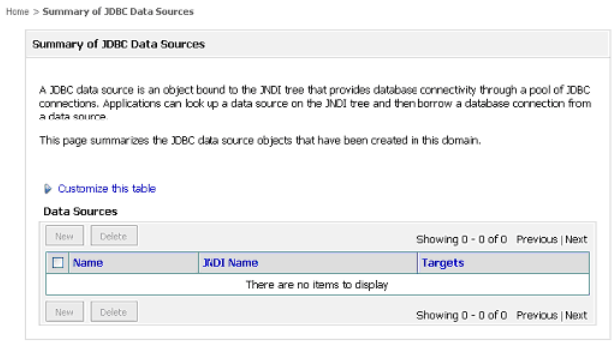


Figure 2.25 - The Data Sources summary

6. Enter the New JDBC Data Source Properties

Name MySQL Data Source
JNDI app/schoolsystem
Database Type MySQL
Database Driver using com.mysql.jdbc.Driver

Figure 2.26 - Create a New Data Sources

7. Select the transaction of the data source to follow the diagram below:

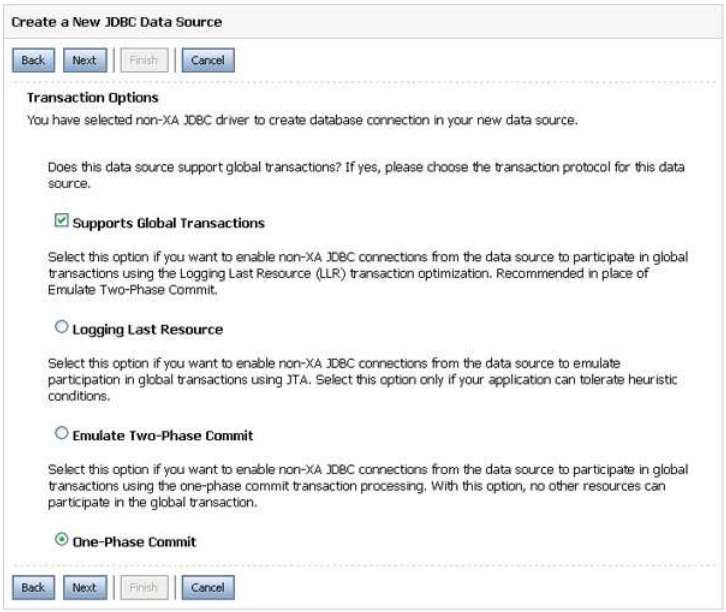


Figure 2.27 - Transaction options

8. Enter the Connection Properties

Database Name schoolsystem
Host Name localhost
Port 3306
Database User root

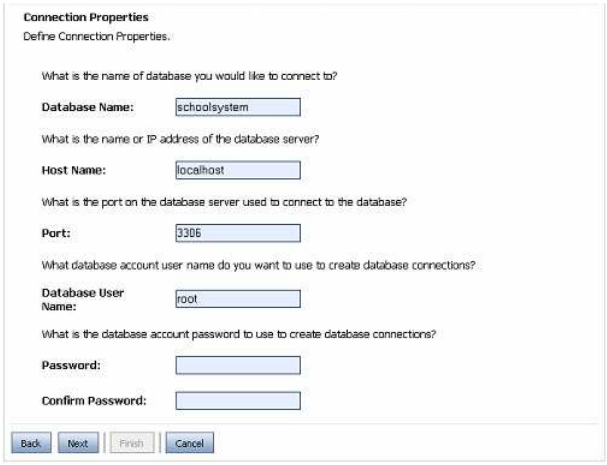


Figure 2.28 - The Connection Properties

9. Click **Test Configuration** to test the connection properties and settings. If the settings are correct, it will show the succeeded message.

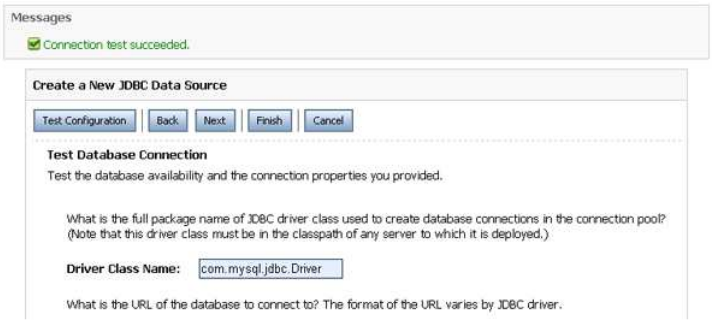


Figure 2.29 - The Connection successful message

10. Select the target of the setting data source; click **Finish** to create JDBC Data Source.

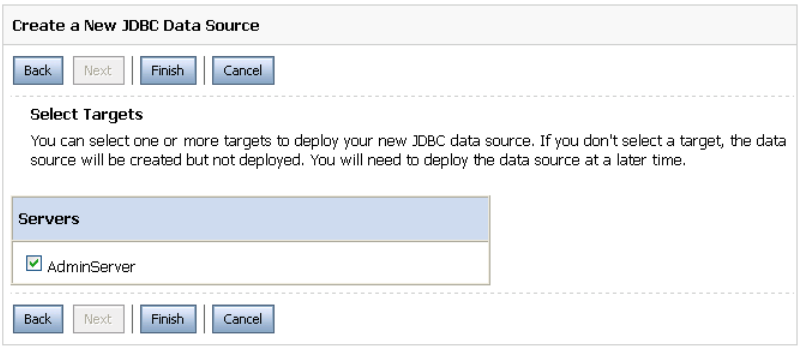


Figure 2.30 - Select target server

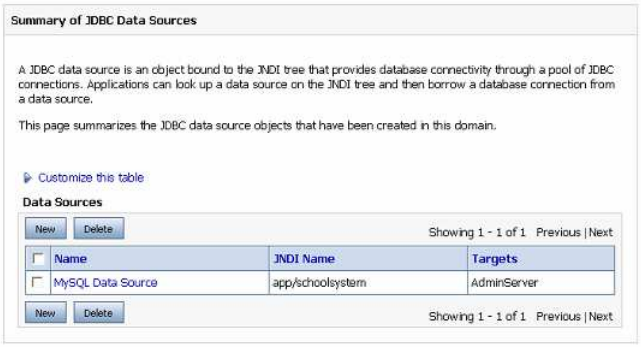


Figure 2.31 - The Data source is added

11. Click **Activate Changes** button to apply the New Data Source Change setting for WebLogic Server

Figure 2.32 - Activate Change

Configuring DataSource Connection on DB-VA

You have setup the datasource on the WebLogic 8.1 and 9.0 and JNDI name called "app/schoolsystem" . You can configure the datasource connection to make the web application to use the datasource on the server.

1. From the menu bar, select **Tools > Object Relational Mapping (ORM) > Generate Code ...** to open the Database Code Generation dialog box.

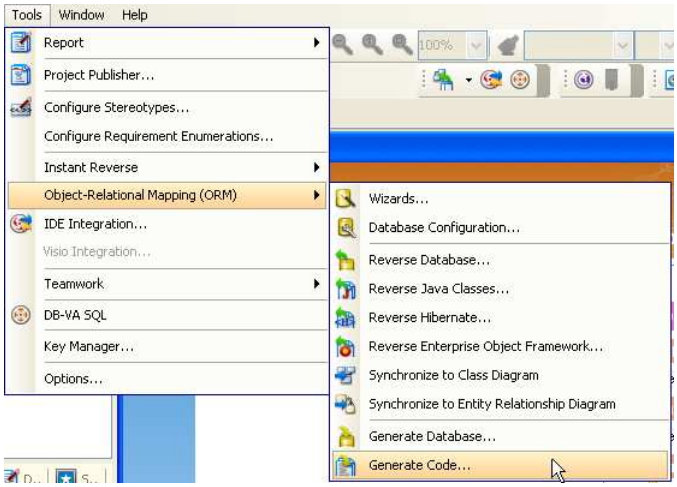


Figure 2.33 - To generate code

2. Select the **Database** tab and select the **Connection** option from **JDBC** to **Datasource**.

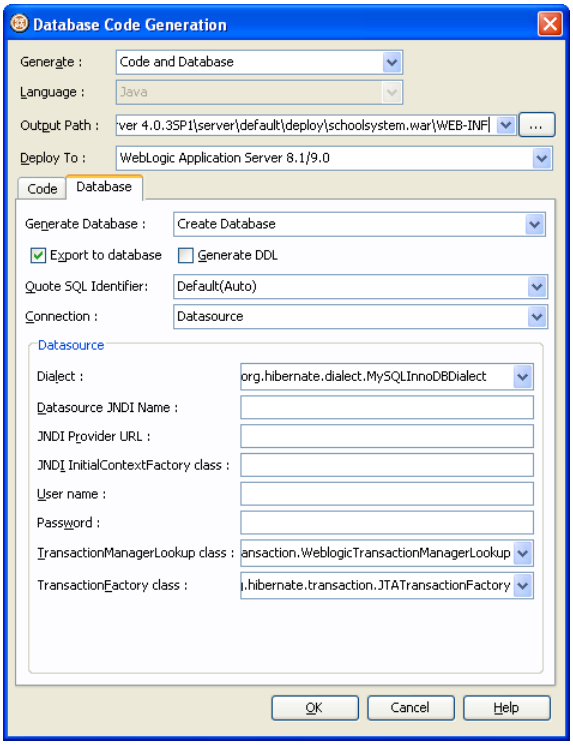
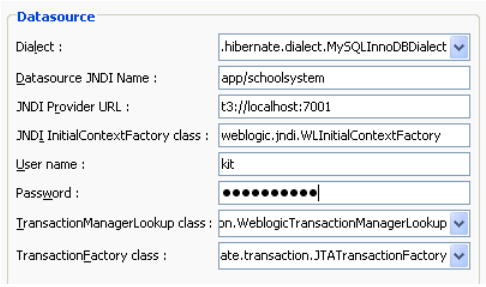


Figure 2.34 - The generate database configure for Data source

3. Enter the following information to configure the datasource of WebLogic Server

Dialect	org.hibernate.dialect.MySQLInnoDBDialect
Datasource JNDI Name	app/schoolsystem
JNDI Provider URL	t3://localhost:7001
JNDI InitialContextFactory class	weblogic.jndi.WLInitialContextFactory
User name	(WebLogic Server login user name)
Password	(WebLogic Server login password)
TransactionManagerLookup class	org.hibernate.transaction.WeblogicTransactionManagerLookup
TransactionFactory class	org.hibernate.transaction.JTATransactionFactory



Click **OK** to regenerate the code.

4. Copy the new configuration files from **schoolsystem.war\src\ormmapping** folder to **schoolsystem.war\classes** folder.

Follow the steps of **Deploy Web Application to WebLogic 8.1** or **Deploy Web Application to WebLogic 9.0** to redeploy the web application. After that it will use the datasource to connect to database in the WebLogic server.

3 Deploying Enterprise Java Web application to IBM WebSphere

Chapter 3 - Deploying Enterprise Java Web Application to IBM WebSphere

DB Visual ARCHITECT (DB-VA) provides different kinds of templates for users to generate Java code. The template will optimize the configuration of generated Java code and select jar files for different application servers or standalone Java application. In this chapter, we will deploy enterprise Java web application to IBM WebSphere. If the application server supports datasource, you can also configure DB-VA to use datasource connection to connect the database on the server.

In this chapter:

- Introduction
- Preparing to Deploy to WebSphere
- Deploying Web Application to WebSphere
- Configuring Datasource on WebSphere
- Configuring Datasource Connection on DB-VA

Introduction

This document is based on the Programmer's Guide for Java - Chapter 3 Developing Java Enterprise Web Application example to demonstrate the deployment step on the WebSphere Server. The Example of Programmer's Guide for java - Chapter 3 is deployed on the JBoss Server, we need to modify some configuration before deploy on the WebSphere Server. Finally, we will configure the web application to use the datasource connection provide by WebSphere server to connect to database.

Preparing to Deploy to WebSphere

Suppose you have downloaded the example of the Programmer's Guide for Java - Chapter 3 Developing Java Enterprise Web Application. You need to change the template of generated code before deploy on the WebSphere Server.

1. From the menu bar, select **Tools > Object Relational Mapping (ORM) > Generate Code ...** to open Database Code Generation dialog box.

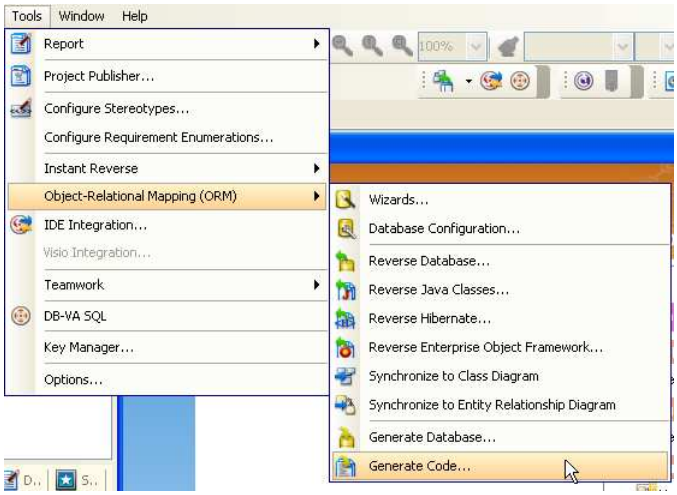


Figure 3.1 - To generate code

2. Change Deploy To option from **JBoss Application Server** to **Generic Application Server**.

Figure 3.2 - Deploy directory

DB-VA helps you to select the corresponding Optional Jar files and set datasource options.

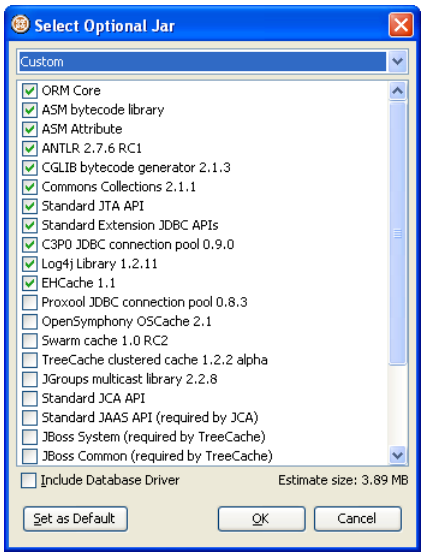


Figure 3.3 - Select Optional Jar

3. Click **OK** to regenerate code.

Copy the **schoolsystem.war\src\ormmapping** folder to **schoolsystem.war\classes** folder to make sure the configuration files are also updated.

Deploying Web Application to WebSphere

Now the orm.jar is updated and generated in the JBoss Server deploy folder.

1. Open the command prompt to the schoolsystem.war folder in the JBoss deploy folder.
2. Execute jar to create a war file in command prompt.

The command to create a war file:

```
jar - cvf schoolsystem.war .
```

The schoolsystem.war file is created inside the schoolsystem.war folder

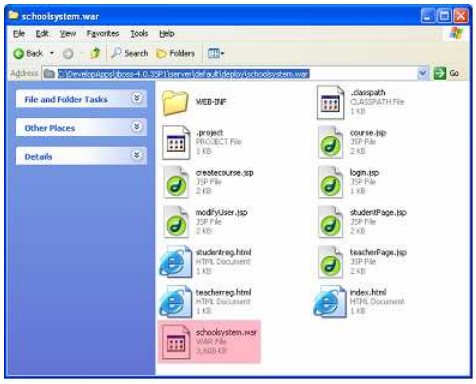


Figure 3.4 - The war file

3. Startup WebSphere Application Server 6. **Start menu > All Program > IBM WebSphere > Application Server v6 > Profiles > AppSrv01 > Start the server**.

Figure 3.5 - Start the Server

4. Go to Administration Console (http://localhost:9061/ibm/console) to login (note that the port number for some WebSphere users may not be 9061, so replace it with the correct one in such case).

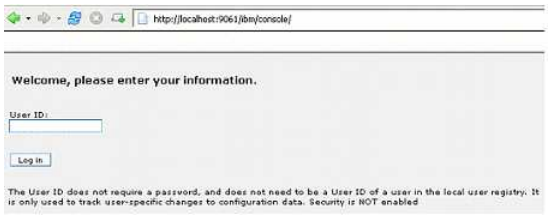


Figure 3.6 - Login page

5. Select **Applications > Install New Application** to install the schoolsystem.war file

Figure 3.7 - Install New Application

6. Select the War file and fill in the **Context Root (schoolsystem)**

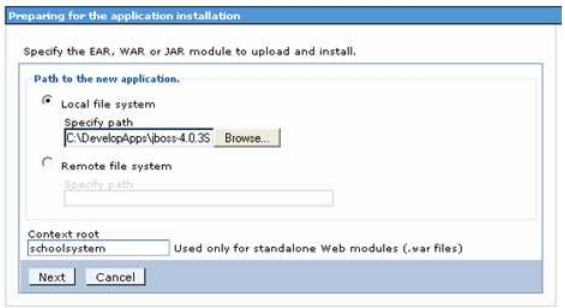


Figure 3.8 - Select the file

7. Click **Next** to finish the installation.

8. Click **Save to Master Configuration** and click **Save** in this page to apply all changes.

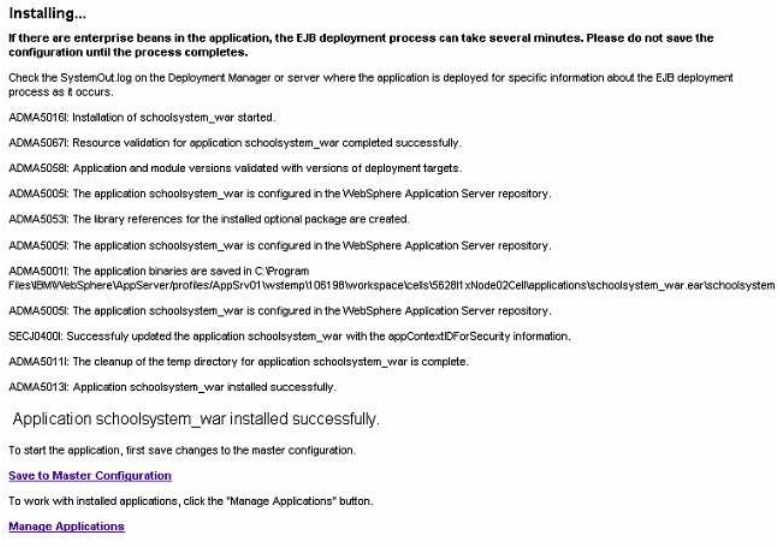


Figure 3.9 - the successful deploy message

9. Start the new deploy SchoolProject Web Application. Select **Applications > Enterprise Application.**

Figure 3.10 - Select Enterprises Application

10. Select schoolsystem_war and click the **Start** button to startup the Web Application on the server.

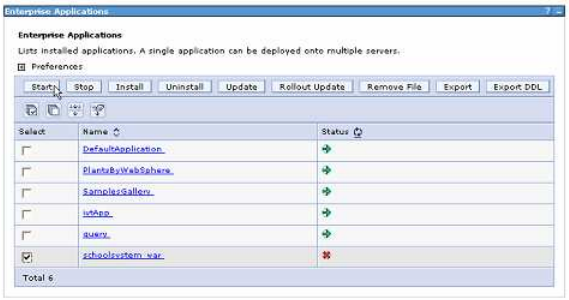


Figure 3.11 - Select the Web Application

11. Go to (http://localhost:9081/schoolsystem/index.html) to confirm the web application is running

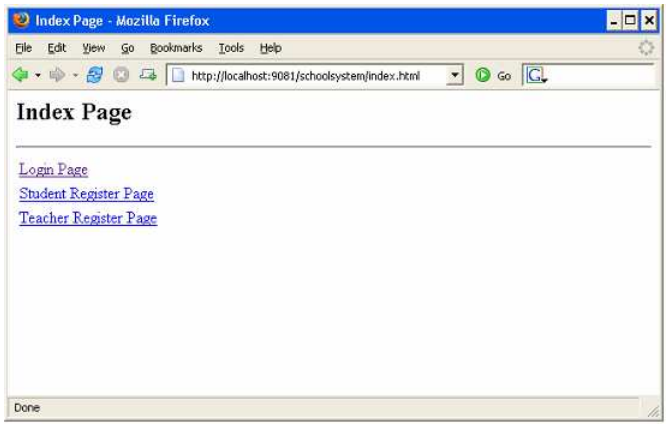


Figure 3.12 - The index page

Configuring Datasource on WebSphere

The WebSphere server can provide a datasource for application to share the JDBC connection within the server. The following steps teach you how to configure datasource on WebSphere server. We will configure the MySQL datasource on WebSphere server as an example.

1. Go to Administration Console (http://localhost:9061/ibm/console) to login.

Figure 3.13 - The Administrator Console

2. Select **Resources > JDBC Providers**

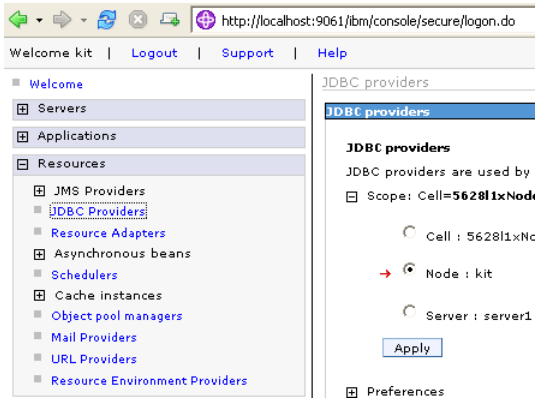


Figure 3.14 -

3. Click **New** button to create MySQL JDBC provider

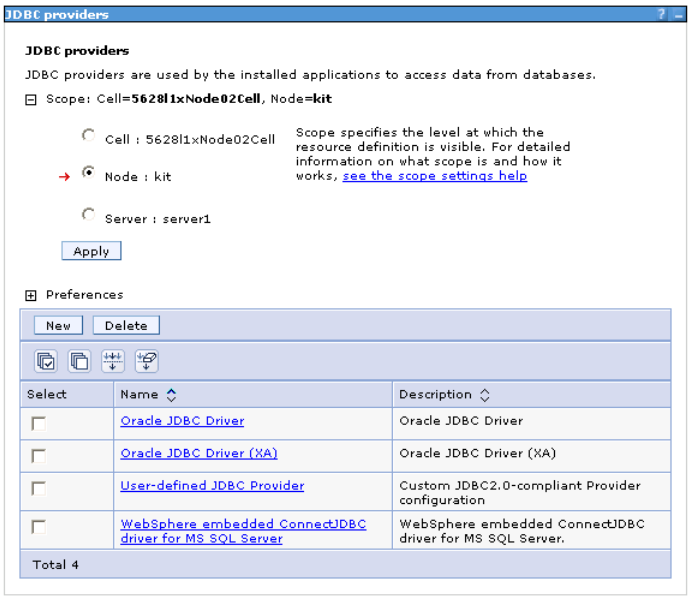


Figure 3.15 - the JDBC Provider

4. In **Choose a type of JDBC provider**, set all options to **User-defined**

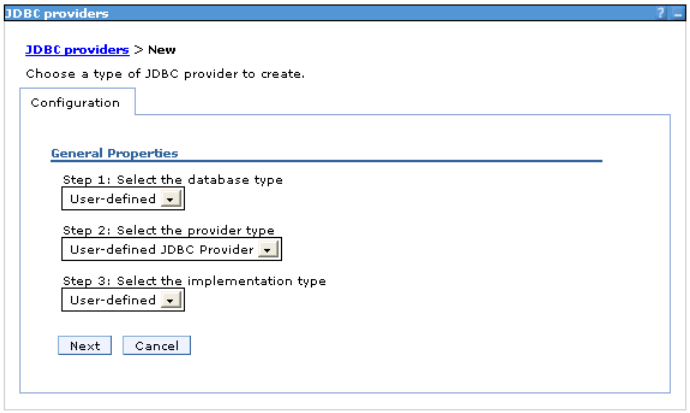


Figure 3.16 - Create a new JDBC provider

5. Fill in general properties and click **Apply**.

Name	MySQL JDBC provider
Classpath	C:\DevelopApps\jboss-4.0.3SP1\server\default\deploy\schoolsystem.war\WEB-INF\lib\mysql.jar
Implementation class name	com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource

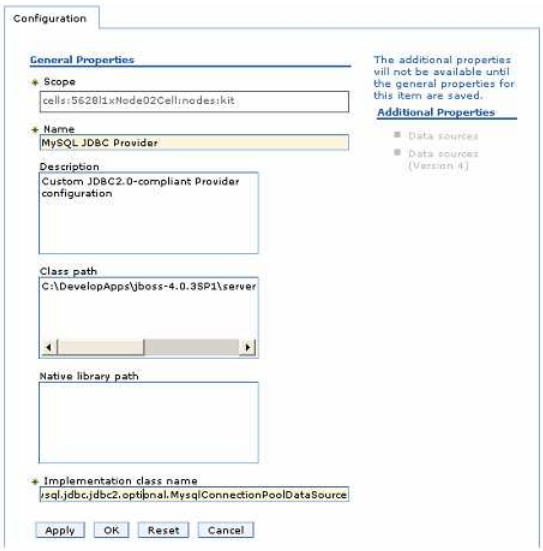


Figure 3.17 - The General Properties

6. Select the **Additional Properties > Data sources** on the right hand side.

Figure 3.18 - The Additional properties

7. Click the **New** button to create new MySQL Data source.

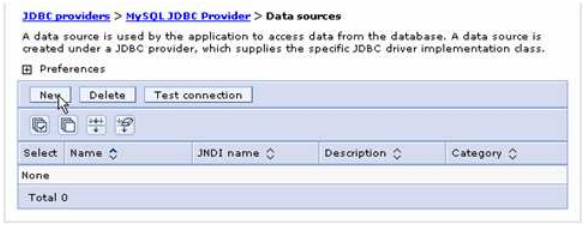


Figure 3.19 - New a Data source

8. Fill in new MySQL Data source information and click **Apply**

Name	MySQL Datasource
JNDI name	app/schoolsystem
Data store helper class name	Select a data store helper class

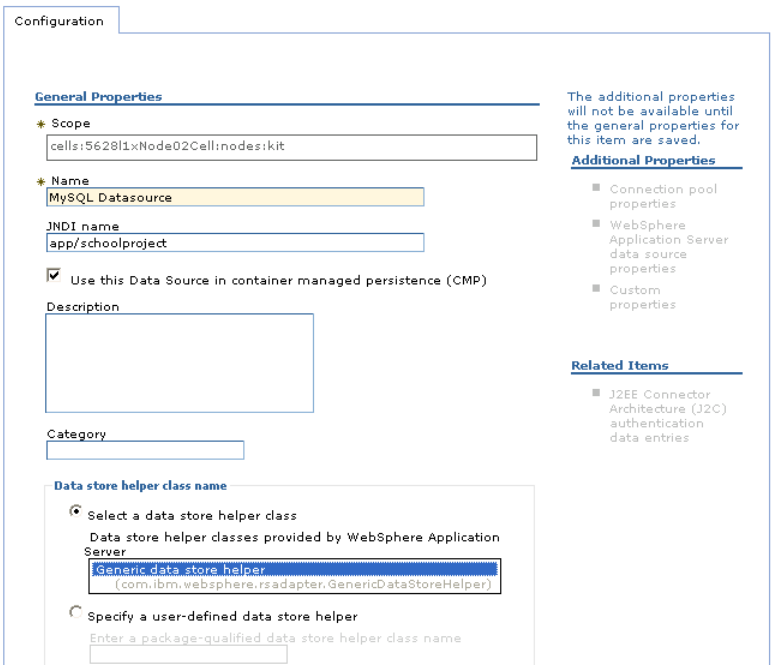


Figure 3.20 - Enter the Data source information

9. Select the **Additional Properties > Custom properties** on the right hand side.

Figure 3.21 - The additional Properties

10. Click the **New** button to create the new property

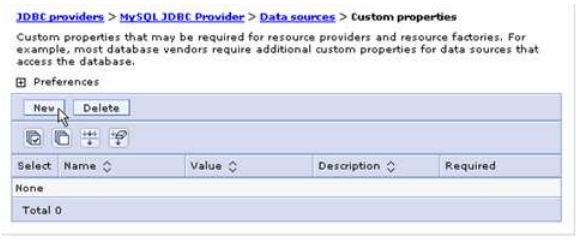


Figure 3.22 - Create a new property

11. Add url property and click **OK** to create the url property

    Name url

    Value jdbc:mysql://localhost/schoolsystem

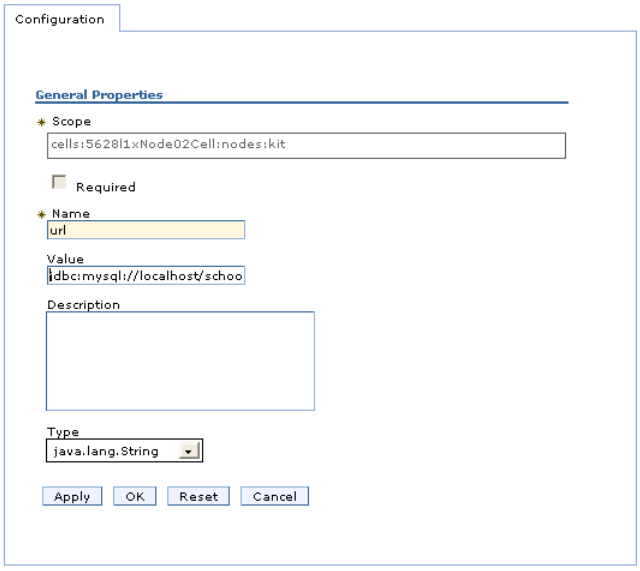


Figure 3.23 - Enter the property information

12. Click **Save** to apply all previous JDBC and datasource setting

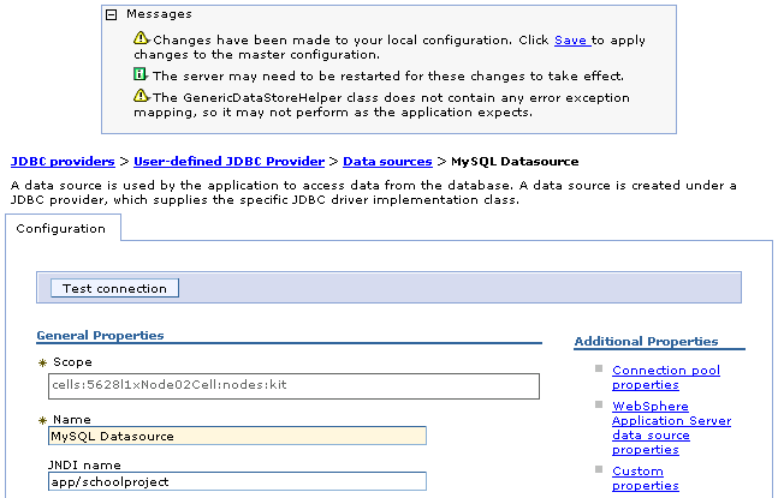


Figure 3.24 - To save the property

13. Select the **J2EE Connector Architecture (J2C) authentication data entries**. It can create the data entries of username and password to connect to database.

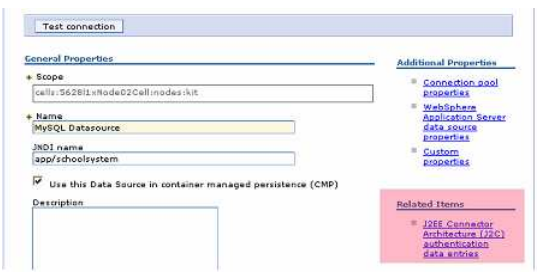


Figure 3.25 - Select J2EE Connector Architecture (J2C) authentication data entries

14. Click **New** to create the user IDs and password.

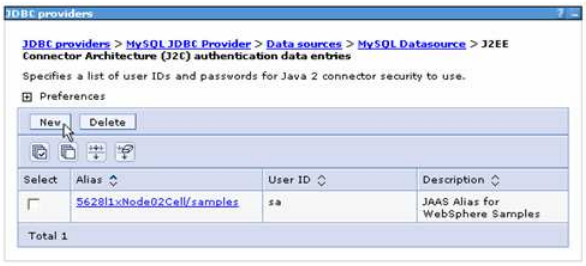


Figure 3.26 - Create a new user ID and password

15. Enter the Alias, User ID and Password and click Apply to save and return to the MySQL Datasource Page.

Figure 3.27 - Enter information for create user

16. Select the previously created username and password on Container-managed authentication and then click Apply.

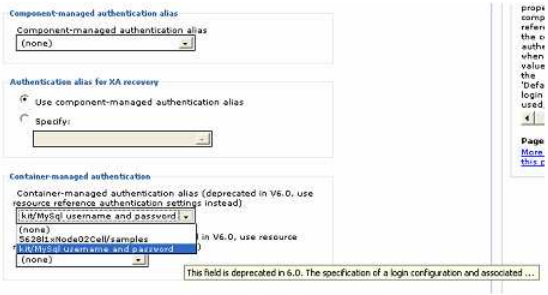


Figure 3.28 - Select the Container-managed authentication

17. Select the MySQL Datasource to **Test connection**

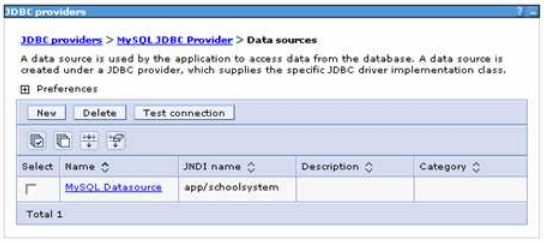


Figure 3.29 - Select Test connection

18. The test connection is successful

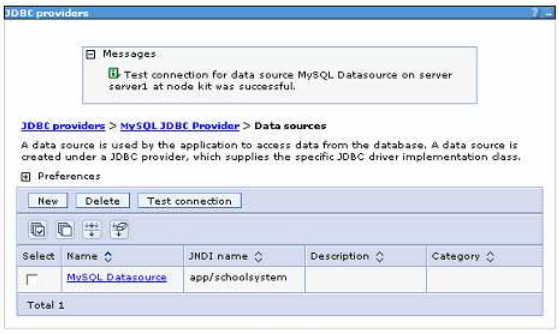


Figure 3.30 - The connection successful message

Configuring the Datasource on DB-VA

You have setup the datasource on the WebSphere and JNDI name called "app/schoolsystem" . You can configure the datasource connection to make the web application to use the datasource on the Server.

1. From the menu bar, select **Tools > Object Relational Mapping (ORM) > Generate Code ...** to open the Database Code Generation dialog box.

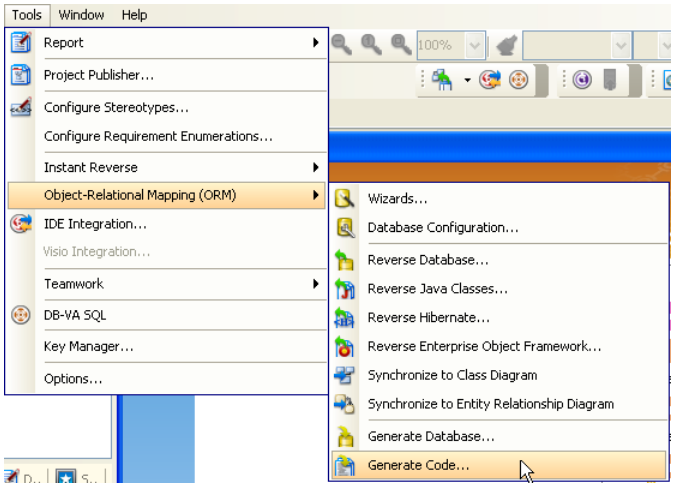


Figure 3.31 - To generate code

2. Select the **Database** tab and select the **Connection** option from **JDBC** to **Datasource**.

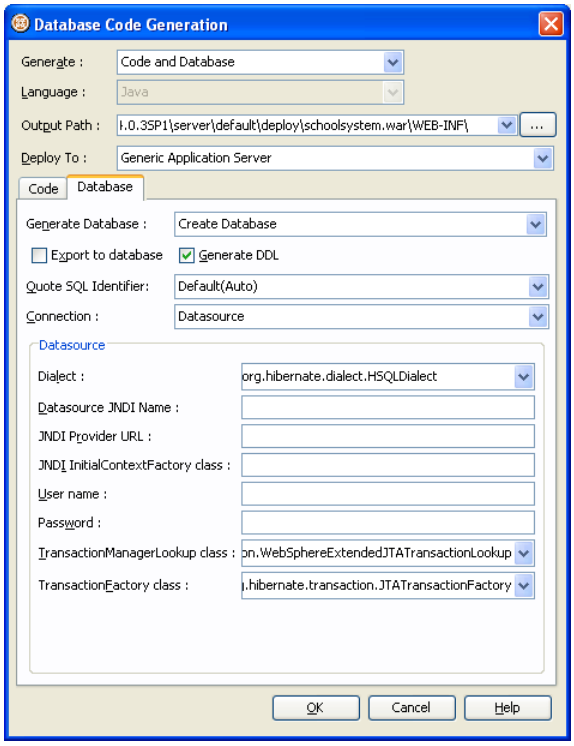
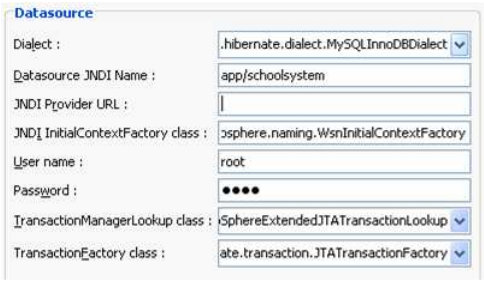


Figure 3.32 - Database configuration for using Data sources

3. Enter the following information to configure the datasource on WebSphere Server

Dialect	org.hibernate.dialect.MySQLInnoDBDialect
Datasource JNDI Name	app/schoolsystem
JNDI Provider URL	(empty)
JNDI InitialContextFactory class	com.ibm.websphere.naming.WsnInitialContextFactory
User name	(connect database user name)
Password	(connect database password)
TransactionManagerLookup class	org.hibernate.transaction.WebSphereExtendedJTATransactionLookup
TransactionFactory class	org.hibernate.transaction.JTATransactionFactory



Click **OK** to regenerate the code.

4. Copy the new configuration files from **schoolsystem.war\src\ormmapping** folder to **schoolsystem.war\classes** folder and use the jar command to create a new war file.
5. Uninstall the schoolsystem application on WebSphere Application server.
6. Repeat the Deploying to WebSphere steps to redeploy the application. The web application can then use the datasource on the server to connect to database.

4

Deploying Enterprise Java Web Application to IBM WebSphere Community Edition

Chapter 4 - Deploying Enterprise Web Application to IBM WebSphere Community Edition

DB Visual ARCHITECT (DB-VA) provides different kinds of templates for users to generate Java code. The template will optimize the configuration of generated Java Code and select jar files for different application servers or standalone Java application. In this chapter, we will deploy enterprise Java web application to WebSphere Community Edition. Since WebSphere Application Server Community Edition is a new application server, DB-VA currently does not support its datasource connection.

In this chapter:

- Introduction
- Preparing to Deploy to WebSphere Community Edition
- Deploying Web Application to WebSphere Community Edition

Introduction

This document is based on the Programmer's Guide for Java - Chapter 3 Developing Java Enterprise Web Application example to demonstrate the deployment steps on the WebSphere Community Edition. Since the Example of Programmer's Guide for Java - Chapter 3 is deployed on the JBoss Server, we need to modify some configuration before deploying on the WebSphere Community Edition.

Preparing to Deploy to WebSphere Community Edition

Suppose you have downloaded the example of the Programmer's Guide for Java - Chapter 3 Developing Java Enterprise Web Application. You need to change the template of generate code before deploying on the WebSphere Community Edition.

1. From the menu bar, select **Tools > Object Relational Mapping (ORM) > Generate Code ...** to open the Database Code Generation dialog box.

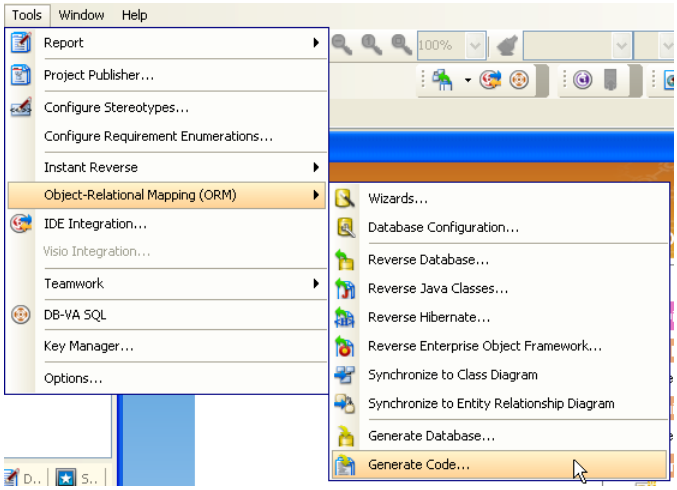


Figure 4.1 - To generate code

2. Change the Deploy To option from **JBoss Application Server** to **WebSphere Application Server Community Edition 1.0**

Figure 4.2 - Deployed To options

DB-VA helps you to select the corresponding Optional Jar files and set datasource options.

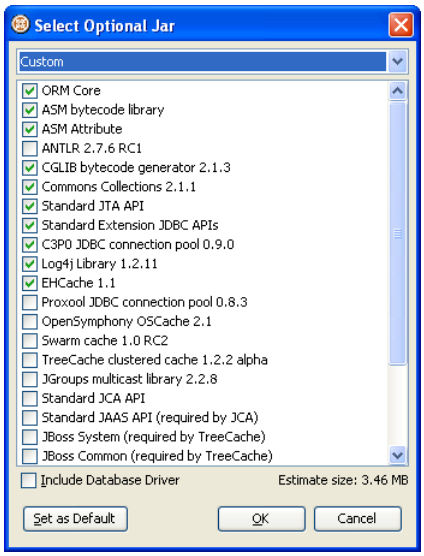


Figure 4.3 - Select Optional Jar dialog

3. Click **OK** to regenerate code.

   Copy the **schoolsystem.war\src\ormmapping** folder to **schoolsystem.war\classes** folder to make sure the configuration files are also updated.

Deploying Web Application to WebSphere Community Edition

Now the orm.jar is updated and generated in the JBoss Server deploy folder.

1. Copy the **antlr.jar** from the **DB-VA installation Directory\ormlib** folder to the **WEBSPHERE_HOME\lib\endorsed** folder.

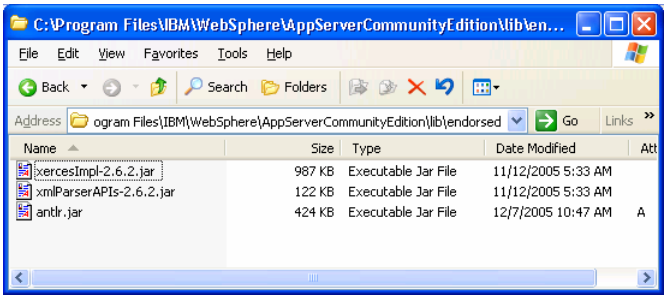


Figure 4.4 - The antlr.jar copied to WebSphere home directory

2. Open the command prompt to the schoolsystem.war folder in the JBoss deploy folder.

3. Execute jar to create a war file in command prompt.

   The command to create a war file:

   ```
   jar - cvf schoolsystem.war .
   ```

   The schoolsystem.war file is created inside the schoolsystem.war folder

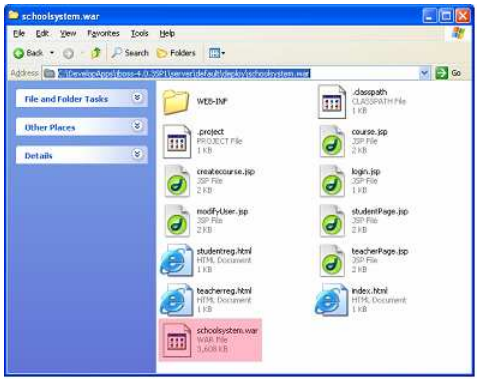


Figure 4.5 - The schoolsystem.war

4. Start the WebSphere Application Server from system menu **Start > All Programs > IBM WebSphere > Application Server Community Edition > Start the server**

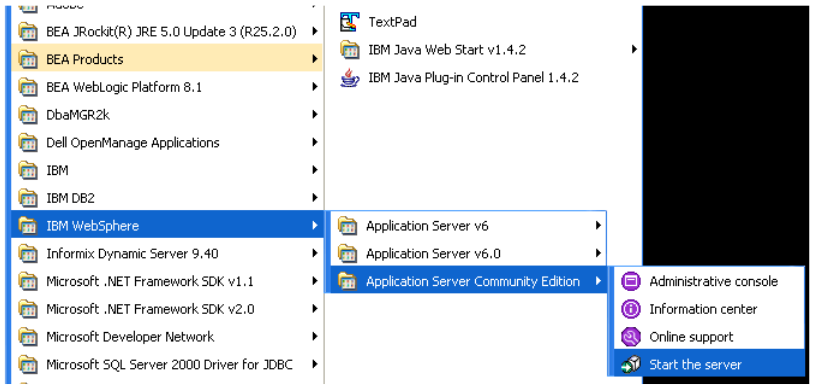


Figure 4.6 - Start WebSphere Application Server

5. Go to Administration Console (http://localhost:8080/Console) and enter user name and password

   The default user name and password:

   User name: System

   password: Manager

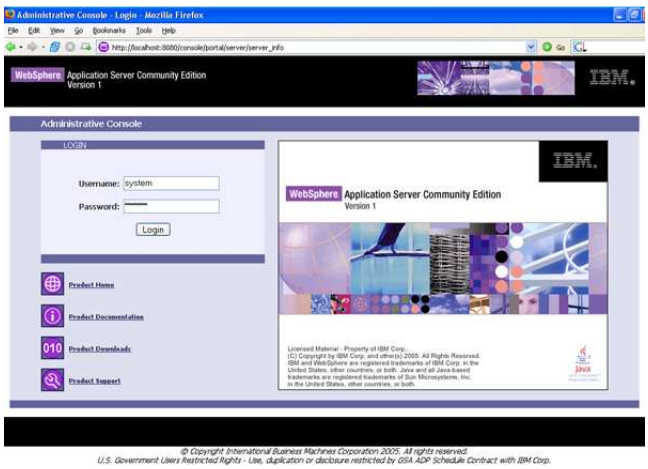


Figure 4.7 - WebSphere Application Server Login page

6. Select **Applications > All Configurations** on the left hand side menu.

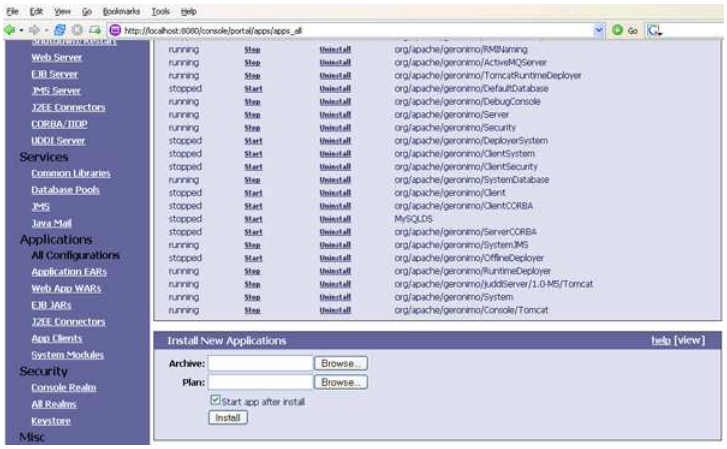


Figure 4.8 - To show all configuration

7. Select the SchoolProject war on the Archive file in **Install New Applications** block and click **Install**.

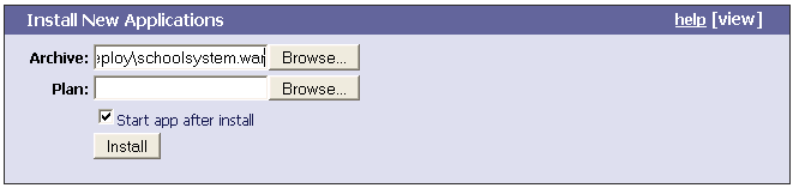


Figure 4.9 - Install new application

8. Install succeeded message is shown

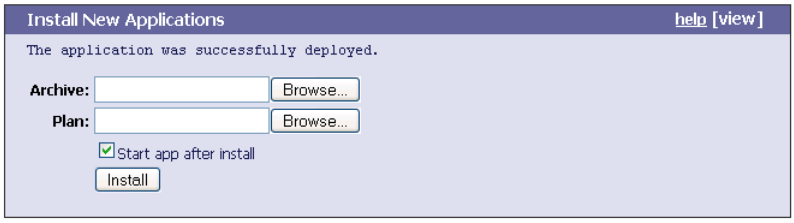
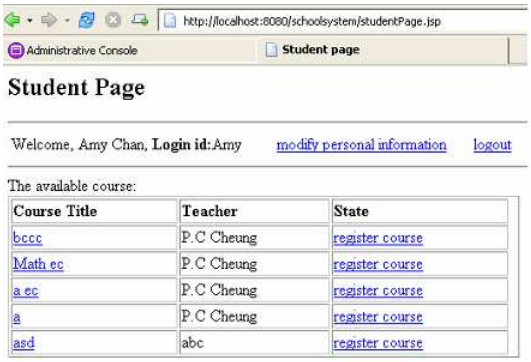


Figure 4.10 - The install successful message

9. Go to http://localhost:8080/schoolsystem/index.html. You can access the record from database by JDBC connection



5 Deploying Enterprise Java Web Application to JBoss

Chapter 5 - Deploying Enterprise Java Web Application to JBoss

DB Visual ARCHITECT (DB-VA) provides different kinds of templates for users to generate Java code. The template will optimize the configuration of generated Java Code and select jar files for different application servers or standalone Java application. DB-VA supports to generate configuration file to make the web application use datasource connection to connect to database in the application server. In this chapter, we will deploy enterprise Java web application to JBoss Application Server.

In this chapter:

- Introduction
- Preparing to Deploy to JBoss
- Deploying Web Application to JBoss
- Configuring Datasource on JBoss
- Configuring Datasource Connection on DB-VA

Introduction

This document is based on the Programmer's Guide for Java - Chapter 3 Developing Java Enterprise Web Application example to demonstrate the deployment step on the JBoss Server. The Example of Programmer's Guide for Java - Chapter 3 is deployed on JBoss, so we will point out some of its important steps. Finally, we will configure the web application to use the datasource connection provide by JBoss application server to connect to database.

Preparing to Deploy to JBoss

Suppose you have downloaded the example of the **Programmer's Guide for Java - Chapter 3 Developing Java Enterprise Web Application**.

1. From the menu bar, select **Tools > Object Relational Mapping (ORM) > Generate Code...** to open the Database Code Generation dialog box.

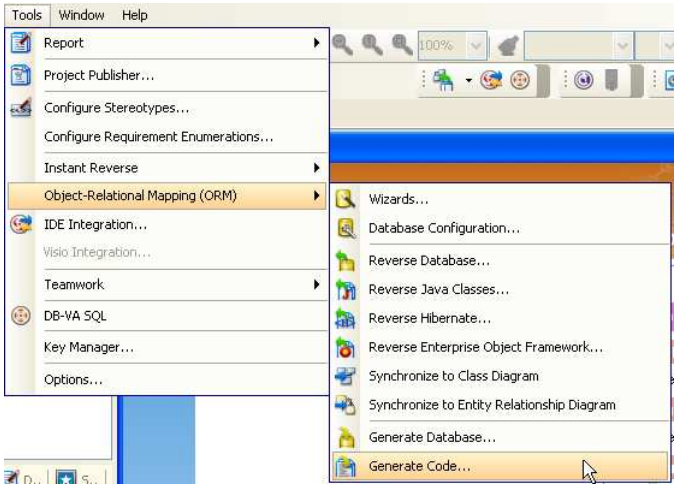


Figure 5.1 - To generate code

2. Set Deploy To option to **JBoss Application Server**.

Figure 5.2 - Set the Deploy To option

DB-VA helps you to select the corresponding Optional Jar files.

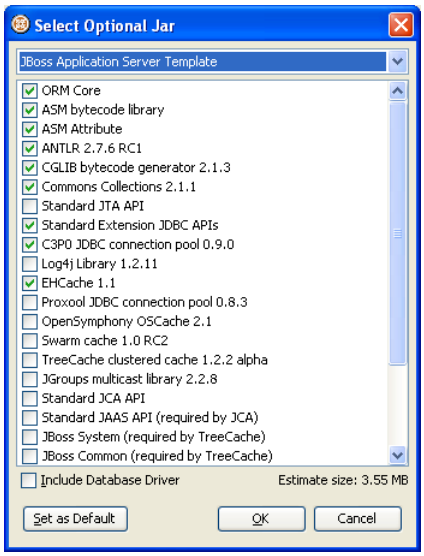


Figure 5.3 - Select Optional Jar dialog

3. Click **OK** to regenerate code.

Copy the **schoolsystem.war\src\ormmapping** folder to **schoolsystem.war\classes** folder to make sure the configure files are also updated.

Deploying Web Application to JBoss

1. After developed the web application with the generated Java code, you must copy the web application folder (named "XXX.war" , for example "schoolsystem.war") and to the JBoss deploy folder (**JBOSS_HOME\server\default\deploy**). For development of the Web Application, you can refer to the Programmer's Guide for Java - Chapter 3 Developing Java Enterprise Web Application.

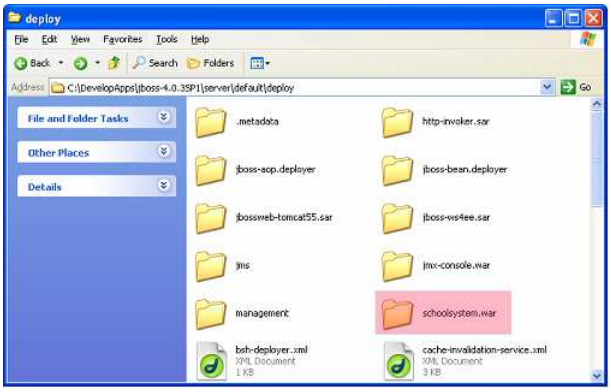


Figure 5.4 - Copy the generated code to JBoss deploy directory

2. Start the JBoss server. Execute the **JBOSS_HOME\bin\run.bat**.

3. Go to http://localhost:8080/schoolsystem/index.html. You can access the database by JDBC connection

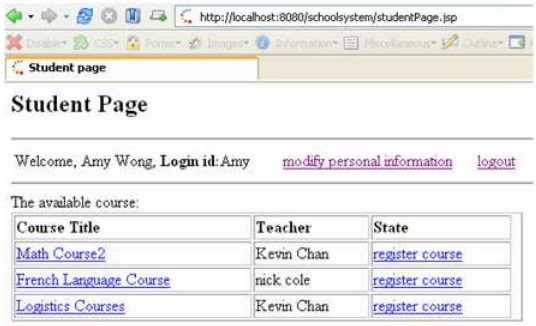


Figure 5.5 - Student Page

Configuring Datasource on JBoss

The JBoss server can provide a datasource for application to share the JDBC connection within the server. The following steps show you how to configure datasource on JBoss application server. We will configure the MySQL database on JBoss server as an example.

1. Copy the JDBC driver to the JBOSS_HOME\server\default\lib

   In this example, DB-VA generated persistent libraries include orm.jar and mysql.jar. mysql.jar is the JDBC driver for MySQL database. mysql.jar can be found at schoolsystem.war\WEB-INF\lib.

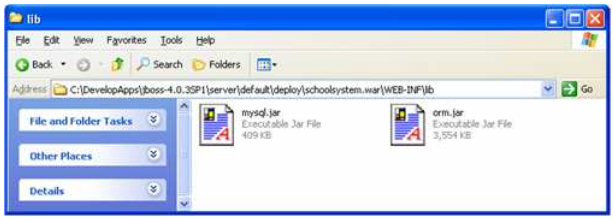


Figure 5.6 - The lib directory of the .war

2. Copy **mysql-ds.xml** from **JBOSS_HOME\docs\example\jca** to the deploy folder (**JBOSS_HOME\server\default\deploy**) and modify the content to follow the table below.

   jndi-name app/schoolsystem

   connection-url jdbc:mysql://localhost/schoolsystem

   driver-class com.mysql.jdbc.Driver

   The sample of modified mysql-ds.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- $Id: mysql-ds.xml,v 1.3.2.1 2004/12/01 11:46:00 schrouf Exp $ - ->
<!-- Datasource config for MySQL using 3.0.9 available from:
http://www.mysql.com/downloads/api-jdbc-stable.html
-->

<datasources>
      <local-tx-datasource>
            <jndi-name>app/schoolsystem</jndi-name>
            <connection-url>jdbc:mysql://localhost/schoolsystem</connection-url>
            <driver-class>com.mysql.jdbc.Driver</driver-class>
            <user-name>root</user-name>
            <password></password>
            <exception-sorter-class-name>
                  org.jboss.resource.adapter.jdbc.vendor.MySQLExceptionSorter
            </exception-sorter-class-name>
            <!-- sql to call when connection is created
```

```
                <new-connection-sql>some arbitrary sql</new-connection-sql>
                -->
                <!-- sql to call on an existing pooled connection when it is obtained from
                pool
                <check-valid-connection-sql>some arbitrary sql</check-valid-connection-
                sql>
                -->

                <!-- corresponding type-mapping in the standardjbosscmp-jdbc.xml
                (optional) -->
                <metadata>
                      <type-mapping>mySQL</type-mapping>
                </metadata>
        </local-tx-datasource>
</datasources>
```

3. The JBoss server creates a datasource and bound it to JNDI name 'java:app/schoolsystem'.

```
09:35:37,453 INFO  [ConnectionFactoryBindingService] Bound ConnectionManager 'jb
oss.jca:service=DataSourceBinding,name=app/schoolsystem' to JNDI name 'java:app/
schoolsystem'
```

Figure 5.7 - Create a datasource on JBoss Application Server

Configuring Datasource Connection in DB-VA

After configuring the datasource on the JBoss server, you can modify the generate Java code configuration to use the datasource connection to connect the database within JBoss server.

1. From the menu bar, select **Tools > Object Relational Mapping (ORM) > Generate Code ...** to open the Database Code Generation dialog box.

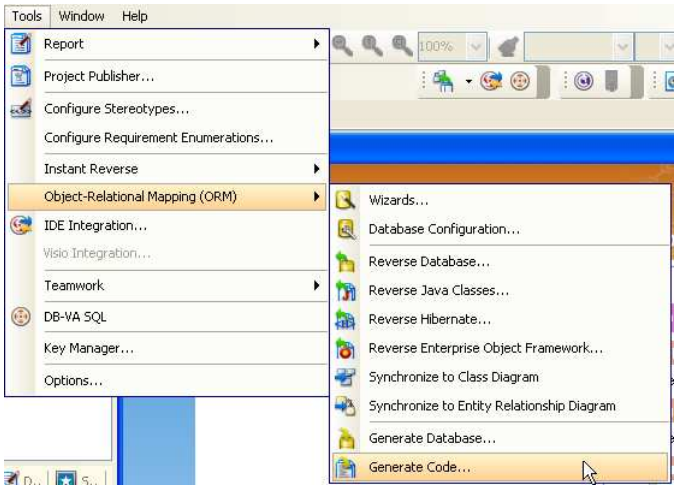


Figure 5.8 - To generate code

2. Select the **Database** tab and select the **Connection** option from **JDBC** to **Datasource**.

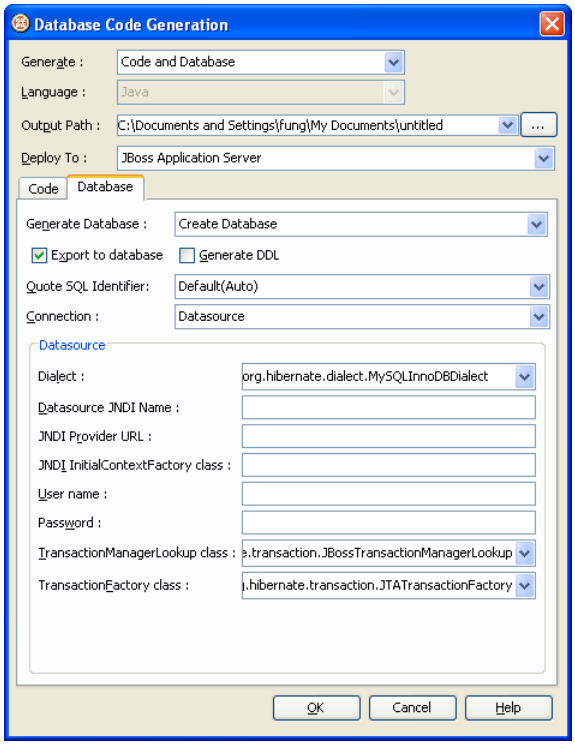
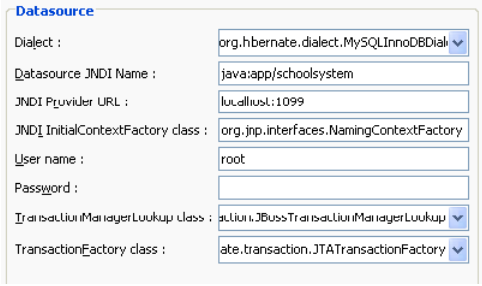


Figure 5.9 - Database Configuration

3. Enter the following information to configure the datasource on JBoss Server.

Dialect	org.hibernate.dialect.MySQLInnoDBDialect
Datasource JNDI Name	java:app/schoolsystem
JNDI Provider URL	localhost:1099
JNDI InitialContextFactory class	org.jnp.interfaces.NamingContextFactory
User name	root
Password	(empty)
TransactionManagerLookup class	org.hibernate.transaction.JBossTransactionManagerLookup
TransactionFactory class	org.hibernate.transaction.JTATransactionFactory

Table 5.1



Click **OK** to regenerate the code.

4. Copy the new configuration files from **schoolsystem.war\src\ormmapping** folder to **schoolsystem.war\classes** folder and use the jar command to create a new war file.
5. Copy the web application folder to the JBoss deploy folder again, then it will redeploy and use the datasource to connect to the database.

6 Deploying Enterprise Java Web Application to Oracle Application Server

Chapter 6 - Deploying Enterprise Java Web Application to Oracle Application Server

DB Visual ARCHITECT (DB-VA) provides different kinds of templates for user to generate Java code. The template will optimize the configuration of generated Java code and select jar files for different application servers or standalone Java application. DB-VA supports to generate configuration file to make the web application to use datasource connection to connect to database in the application server. In this chapter, we will deploy enterprise Java web application to an Oracle Application Server.

- Introduction
- Preparing to Deploy to Oracle Application Server
- Deploying Web Application to Oracle Application Server
- Configuring Datasource on Oracle Application Server
- Configuring Datasource Connection on DB-VA

Introduction

This document is based on the Programmer's Guide for Java - Chapter 3 Developing Java Enterprise Web Application example to demonstrate the deployment steps on the Oracle application server. The Example of Programmer's Guide for Java - Chapter 3 is deployed on JBoss, so we will point out some important steps for user to modify to deploy on Oracle application server. Finally, we will configure the web application to use the datasource connection provide by the Oracle application server.

Preparing to Deploy to Oracle Application Server

Suppose you have downloaded the example of the **Programmer's Guide for Java - Chapter 3 Developing Java Enterprise Web Application**.

1. From the menu bar, select **Tools > Object Relational Mapping (ORM) > Generate Code ...** to open the Database Code Generation dialog box.

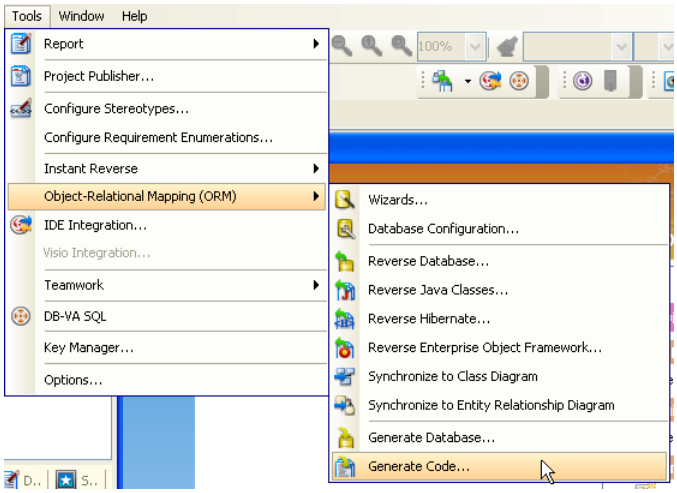


Figure 6.1 - To generate code

2. Set Deploy To option from **JBoss Application Server** to **Generic Application Server.**

Figure 6.2 - The Deploy to option

DB-VA helps you to select the corresponding Optional Jar files.

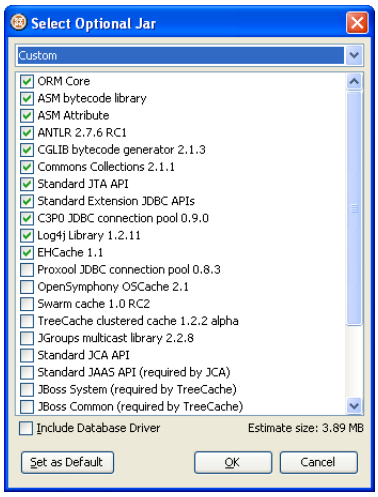


Figure 6.3 - select Optional Jar dialog

3. Click **OK** to regenerate code.

Copy the **schoolsystem.war\src\ormmapping** folder to **schoolsystem.war\classes** folder to make sure the configuration files are also updated.

Deploying Web Application to Oracle Application Server

Suppose you have installed the Oracle Application Server. In this example we will use Oracle Application Server 10g.

1. Open the command prompt and change to the schoolsystem.war folder in the JBoss deploy folder.
2. Execute jar to create a war file in command prompt.

The command to create a war file:

```
jar -cvf schoolsystem.war .
```

The schoolsystem.war file is created inside the schoolsystem.war folder

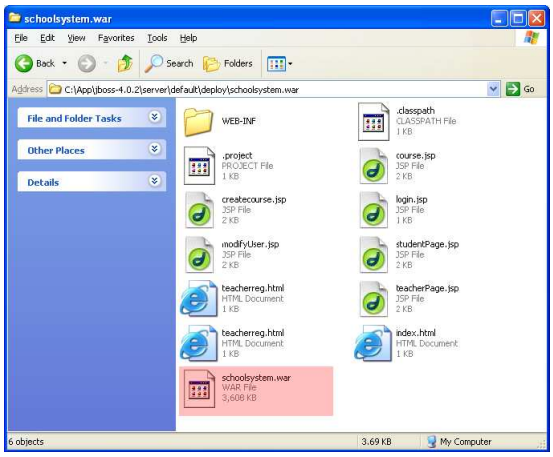


Figure 6.4 - The .war file

3. Start the Oracle Application Server from system menu **Start > Programs > Oracle Application Server 10g > Start ....**

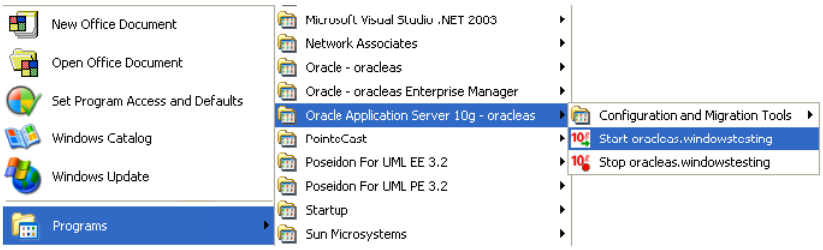


Figure 6.5 - Start Oracle Application Server

4. Go to the Application Server Control Console from system menu **Start > Programs > Oracle - oracleas Enterprise Manager > Application Server Control Console** or go to http://localhost:108100.

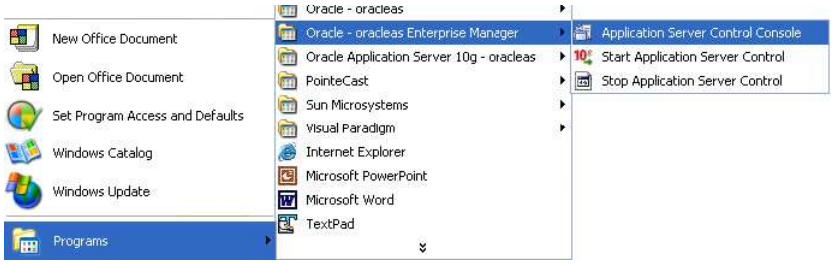


Figure 6.6 - To open the Application Server Control Console

5. Enter username and password to login. Select the **J2EE Applications** tab and then click **home** on **OC4J Instance** column to go to the OC4J Home page.

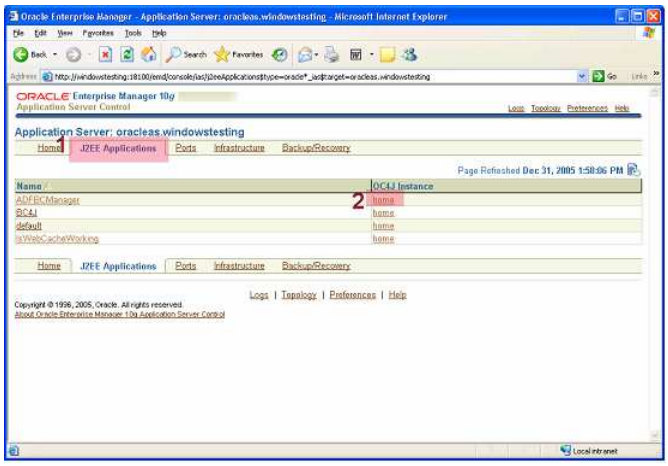


Figure 6.7 - Application Server Control Console

6. In the OC4J home page, select the **Application** tab and click **Deploy WAR file**.

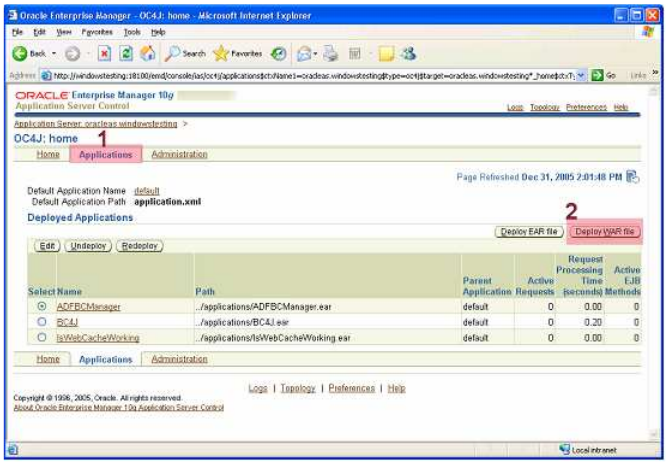


Figure 6.8 - To deploy WAR file

7. Select the schoolsystem.war file location and enter the Application Name and Map to URL information. The Map to URL must be start with /. After that, click **Deploy**.

Figure 6.9 - Enter the information of the WAR

8. When you see the successfully deployed message, the web application is running on the Oracle Application Server.

Figure 6.10 - The deploy successful message

9. Use the Map to URL information to access your web application. Go to http://localhost/schoolsystem/index.html. You may have different port number for the application server.

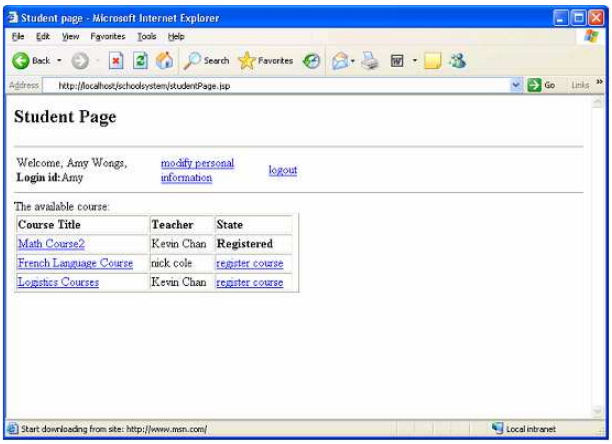


Figure 6.11 - The Student Page

Configuring Datasource on Oracle Application Server

The Oracle Application Server can provide a database for application to share the JDBC connection within server. The following steps teach you how to configure datasource on Oracle server. We will configure the MySQL datasource on Oracle server as an example.

1. Copy the mysql.jar to ORACLE_HOME/j2ee/home/applib. The mysql.jar can be found at the DB-VA generate output folder/schoolsystem.war/WEB-INF/lib.

   The j2ee/home/applib directory contains libraries that are used by all applications on your application server. By dropping the MySQL JDBC drivers here, they can be used by all applications running on this OC4J instance.

2. Edit ORACLE_HOME/j2ee/home/configure/data-sources.xml to add your datasource.

```
<data-source
class="com.evermind.sql.DriverManagerDataSource"
name="MySQLDS"
location="app/schoolsystem"
xa-location="app/xa/schoolsystem"
ejb-location="jdbc/MySQLDS"
connection-driver="com.mysql.jdbc.Driver"
username="root"
password=""
url="jdbc:mysql://localhost/schoolsystem"
inactivity-timeout="30"
/>
```

3. Go to the OC4J home page to restart OC4J, the MySQL datasource is running on the Oracle application server now.

Figure 6.12 - Restart OC4J

Configuring Datasource Connection on DB-VA

You have setup the datasource on the Oracle and JNDI name called "app/schoolsystem" . You can configure the datasource connection to make the web application to use the datasource on the Server.

1. From the menu bar, select **Tools > Object Relational Mapping (ORM) > Generate Code ...** to open the Database Code Generation dialog box.

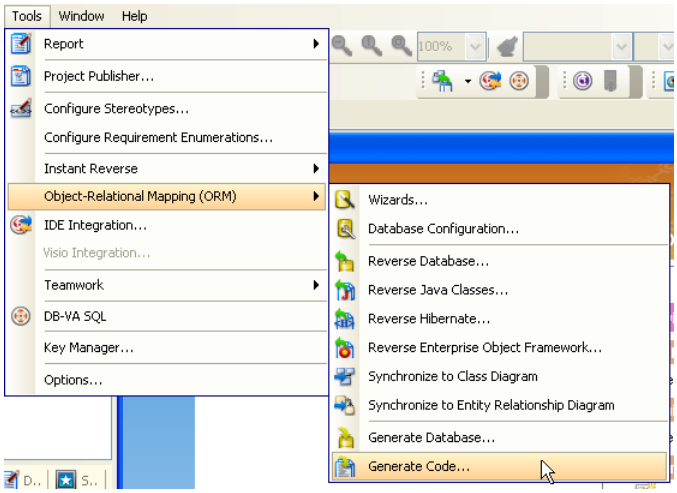


Figure 6.13 - To generate code

2. Select the **Database** tab and select the **Connection** option from **JDBC** to **Datasource**.

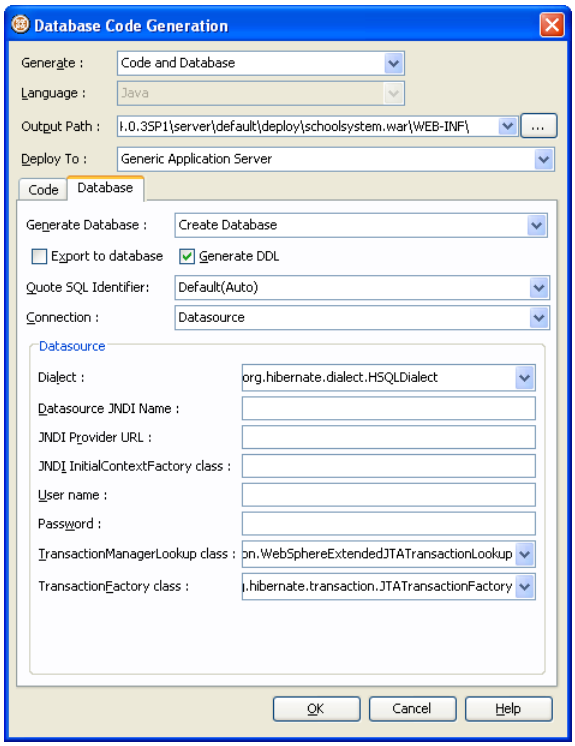


Figure 6.14 - Database Configuration

3. Enter the following information to configure the datasource on Oracle Application Server

Dialect	org.hibernate.dialect.MySQLInnoDBDialect
Datasource JNDI Name	java:comp/env/app/schoolsystem
JNDI Provider URL	(empty)
JNDI InitialContextFactory class	(empty)
User name	(connect database user name)
Password	(connect database password)
TransactionManagerLookup class	org.hibernate.transaction. OC4JTransactionManagerLookup
TransactionFactory class	org.hibernate.transaction.JTATransactionFactory

Table 6.1

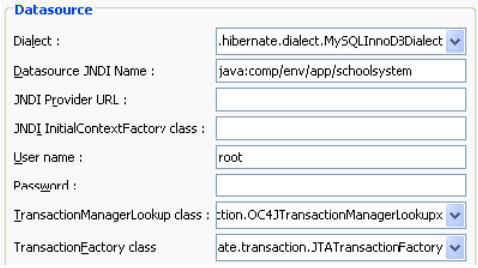


Figure 6.15 - Datasource Configuration

Click **OK** to regenerate the code.

4. Modify the **Web.xml** file in **schoolsystem.war\WEB-INF** folder to add **resource-ref** to the datasource.

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

5. Undeploy the schoolsystem web application on the Oracle application server and then repeat the step of Deploying Web Application on Oracle Application Server to deploy the web application. The web application run on the Oracle server will use the datasource connection to connect to database.