Module 2 Assignment Instructions

Part 1 - Creating Derby Database and a client to connect to it.

Step 1

First off, you need to create a database. In order to do that please create a script to start Derby's **IJ** CLI, Command Line Interface, or utility, and then to use **IJ** commands to create database. Here is an example of the script:

```
set derby_home=c:\db-derby-10.14.1.0-bin
set PATH=%derby_home%\bin;%path%
set classpath=%derby_home%\lib\derby.jar;%classpath%
ij
```

Now we are ready to create a new database called **JHU**, with one DB table called **STUDENT**, having the following columns:

Column Name	**Data Type**
FIRST_NAME	varchar(40)
LAST_NAME	varchar(40)
SSN	char(11)
EMAIL	varchar(40)
ADDRESS	varchar(40)
USERID	varchar(8)
PASSWORD	varchar(8)

While running Derby **IJ** utility, the following SQL commands can be used to create an empty **JHU** database with **STUDENT** table in it:

```
CONNECT 'jdbc:derby:c:\JHU;create=true';
CREATE TABLE STUDENT ( FIRST_NAME varchar(40), LAST_NAME varchar(40) , SSN
char(11) , EMAIL varchar(40), ADDRESS varchar(40), USERID varchar(8) ,
PASSWORD varchar(8) );
quit;
```

You can simply create a file, say called "JHU_create.txt", and provide its name as an input parameter to **IJ** utility:

```
set derby_home=c:\db-derby-10.14.1.0-bin
set PATH=%derby_home%\bin;%path%
set classpath=%derby_home%\lib\derby.jar;%classpath%
ij JHU_create.txt
```

After executing this script, **JHU** database is created with **STUDENT** table.

Step 2

Create and populate a local disk file in ASCII format with up to 10 rows of data to be used as input data in the next step of this assignment. The data have to be formatted in a convenient way so that it will be easy to read and parse it in the next step (say, you can use StringTokenizer class for parsing). You can use a space as a delimiter to separate different columns, for example:

```
John_1 Doe_1 111-22-3333 johndoe1@company.com johndoe1_address johndoe1_id johndoe1_pw
John_2 Doe_2 222-33-4444 johndoe2@company.com johndoe2_address johndoe2_id johndoe2_pw
John_3 Doe_3 333-44-5555 johndoe3@company.com johndoe3_address johndoe3_id johndoe3_pw
John_4 Doe_4 444-55-6666 johndoe4@company.com johndoe4_address johndoe4_id johndoe4_pw
```

Step 3

In this step of the assignment, you **will not be using** WebLogic Server database connection pool capabilities. Instead, you will design and implement a two-tiers connection scenario, that is **client - database through the JDBC driver**.

Write a Java program that connects to the JHU database and inserts all the rows of data from the input file (that you created in Step 2) into STUDENT table.

After inserting data, print out the full contents of the STUDENT table using ResultSet class, first off by "walking forward" and then by "walking backward".

Note: Walking backward requires certain preparations and special considerations. Pay attention to program creation of a Statement object and how to handle the autoCommit feature if you need to work with scrollable cursor (see "Result Sets" Sun's JDBC Online Tutorial for details).

If you have difficulties getting it to work use our forum for discussions "Session 3 Homework discussion" topic to ask questions.

Take a screen shot of your program output and save it on your local drive for submitting.

Step 4

In this step of the assignment you have to create JDBC Data Source resource for managing a pool of database connections in the WebLogic Server domain that you are running. You will be working with this Data Source resource in the Part 3 of this assignment and in several upcoming assignments during the course.

Before you begin working on it please get introduced with BEA WebLogic "Configuring JDBC DataSources" document:

http://docs.oracle.com/middleware/1221/wls/JDBCA/jdbc_datasources.htm#JDBCA137

You will create JDBC Data Source resource using the Admin Console web browser based tool of running instance of the WLS.

But before that, make sure that derby.jar file is in CLASSPATH. We need it there because we want the WLS running instance to load Derby JDBC driver when a client will start working with the DataSource object that is defined using that JDBC driver. The easiest way to get derby.jar file defined in CLASSPATH is to modify setDomainEnv.cmd script located in %WLS_HOME%\user_projects\domains\your-WLS-domain-directory\bin. Get there, open setDomainEnv.cmd file with Notepad, and stick the following command into it:

set PRE_CLASSPATH=C:\db-derby-10.14.1.0-bin\lib\derby.jar;

This setDomainEnv script is called by startweblogic script when WLS instance starts.

So, now start WLS and connect to it through the web browser based Admin console:

http://localhost:7001/console

On the left hand side, click "Services", then click on "Data Sources".

On the right panel, “Summary of JDBC Data Sources", click on "New" button to open drop-down menu, then select “Generic Data Source” and click on it.

Now, after you've got "Create a New JDBC Data Source" screen, type **jhuDS** in the "Name" box, **jhuDataSource** in "JNDI" box, select "Derby" as “Database Type” and wait for a screen to refresh.

Make sure to select "Other" as Database Driver and click on "Next".

On the next screen, "Create a New JDBC Data Source", leave default parameters unchanged, and click "Next".

Next screen, "Connection Properties", is for defining DB connection properties. So, here they are:

```
Database Name: JHU
Host Name: C:\JHU
Port: any-number
DB User Name: anything-you-want
Password: anything-you-want
```

After you finish defining DB connection pool properties click on "Next", and get into "Test DataBase Connection" screen. Once there make sure:

- to enter in the "Driver Class Name" textbox a name of Derby JDBC driver, which is: **org.apache.derby.jdbc.EmbeddedDriver**
- the URL textbox includes the following parameter: **jdbc:derby:c:\your-db-location\JHU;create=true**

  (notice a "semicolon" NOT a "colon" before "create" parameter !!!)

Next, type in "STUDENT" as database table name in "Test Table Name" box and click on the "Test Configuration" button. You should get a screen refreshed with "Connection test succeeded" message at the top. If not go back and review all parameters you provided (perhaps you mistyped something somewhere).

Now, finally, you can click on the "Next" button to finish creating this pool and getting your domain server to actually know about it. So on the "Select Targets" page check "AdminServer" box and click a "Finish" button.

Take a screen shot of the Admin console showing the DB connection pool you created. You will be required to submit it.

Part 2 - JNDI

This part of the homework assignment is to familiarize students with the most frequently used aspects of the JNDI.

Step 1

In order to reinforce learning about the JNDI please:

- Execute the following JNDI examples programs (adapted from WebLogic Server 8.1 package), InitialContextExample and the WebLogicContextExample. Please download the other attachments for this assignment to get source code for each of these programs:

  InitialContextExample.java
  and
  WebLogicContextExample.java

  Save each code on your hard drive. Then compile it with proper directory structure for Java packaging. Then run each code as a client against running WLS instance.

  Note: before running those example programs from DOS shell, make sure to set up CLASSPATH for proper inclusion of JEE and WebLogic jar files. To do that, simply switch to **C:\Oracle\Middleware\user_projects\domains\yourDomain\bin** directory and run **setdomainenv** script which will set CLASSPATH up.

- Study both InitialContextExample and WebLogicContextExample Java source code. Make sure to understand every Java statement related to JNDI processing.

While running both programs, capture screen shots of your DOS prompt demonstrating successful execution.

Step 2

Before you start developing the main program/client, you have to develop Java class called **StudentInfo** that encapsulates the following private properties (Java variables):

first_name
last_name
address
phone
email

All of these properties are of type String.

This **StudentInfo** class should have accessor methods **getXXX(...)** to return private properties, one method per each variable, so that this class is a Java bean.

At this point you are ready to develop a Java program/client that will bind three StudentInfo objects into the WebLogic Server JNDI naming service. Each of them should be an instance of the StudentInfo class that you just developed. A client can use **rebind( )** method instead of bind( ) to be able to rerun it multiple times.

A few tips I would like to give you. As a Java bean, the StudentInfo class has to be of type Serializable. Also, since you want WebLogic Server to bind an object to JNDI tree, WebLogic Server has to be able to load StudentInfo class. So, you have to include a location (path) of StudentInfo class into CLASSPATH before starting WebLogic Server.

Run this Java program with WebLogic Server. Capture screen shots of your DOS prompt demonstrating successful execution.

Lastly, develop another Java client to lookup all of these bounded objects using the names they have been bounded under. To verify that all the objects have the same state as when they were bound by the first program simply print out the information provided by the getXXXX(...) methods of the StudentInfo class.

Run it and capture screen shots of your DOS prompt demonstrating successful execution.

Note 1: for the purposes of Part 2 of the homework assignment, we are binding three "StudentInfo" objects to the WebLogic Server's JNDI tree and then are looking up for them, and we are not going to use those StudentInfo objects anywhere within upcoming assignments.

Note 2: I would like to provide here a short instruction on how to run a stand-alone Java application that has to connect to the WebLogic Server instance and to bind an object or objects to JNDI tree and/or to look up those objects on the JNDI tree. In addition, this instruction is supposed to help students to understand how to run Java applications that need to work with the WLS in the future assignments.

So here is what needs to be done.

- Open DOS shell
- Go to you Weblogic Server domain directory (i.e. c:\oracle\middleware\user_projects\domains\lfDomain) and run "startweblogic" script
- Wait for the WLS instance to start, then open web browser and go to Admin console (http://localhost:7001/console)
- On the left panel, under Domain Structure, open up Environment-Servers, then on the right panel click on AdminServer link
- It will open next window, where you need to click on View JNDI tree on the right panel, which will display JNDI tree

At this point please observe the JNDI tree state with the objects bound to it.

Now you are ready to do the next steps related to working with stand-alone Java application.

- Open another DOS shell
- Before anything else, run "setdomainenv" script is located in c:\oracle\middleware\user_projects\domains\lfDomain\bin. This will set up CLASSPATH to include WLS jars and JEE jars that are necessary to compile and to run Java applications that are using WLS
- Compile Java code examples provided in this HW
- Run each Java application example
- Refresh JNDI tree on Admin console and observe "example.one" entry in the JNDI tree that was created by WebLogicContextExample code

Part 3 - Using DataSource object to provide database connections to a client

Ok, the last step in this assignment.

Take the program you have created in the JDBC part of this assignment (Part 1, Step #3) and modify it to use the DataSource object defined in WLS to obtain a connection to the JHU database and to print all records from table STUDENT.

Run your program and capture the screen shots with the output produced.