Database Access: Glassfish/MySQL

This exercise shows how to use the MySQL database system from an application deployed on a Glassfish server using the Java Database Connectivity (JDBC) API and then with Java Persistence API (JPA).

For this exercise we will use the MySQL server on the Linux server. The URL of the server, your login details, and the name of your database will be provided by the tutor during the lab session.

Before starting the practice take into account the following:

- It is advisable to use your own computer; otherwise, after creating your programs you have to deploy the generated .war file to Glassfish in ats
- When deploying the application to ats, remember to create the connection to the databases with the same name of your connection

MySQL

There are many ways of connecting to a MySQL server to create and administer databases. One way is using the command line. You can do so by connecting to the Linux server and entering the command.

```
mysql -u username -p
```

To change your password you can use the following SQL command:

```
mysql> set password = password('newPassword')
```

On the MySQL command line you can directly enter SQL commands (terminated with a semicolon). Eg:

```
mysql> use yourDatabaseName;

mysql> CREATE TABLE Customer (
       Name VARCHAR(80) NOT NULL,
       Street VARCHAR(100) NOT NULL,
       Postcode INTEGER NOT NULL,
       PRIMARY KEY (Name))
       ;

mysql> INSERT INTO Customer VALUES ('ATI', 'J Liivi 2', 50409);

mysql> SELECT * FROM Customer;

mysql> DROP TABLE Customer;
```

MySQL using NetBeans

Next we will show how to use the NetBeans IDE to connect to the MySQL server. For security reasons, the MySQL database server only accepts connections from within the university network, so you might not be able to complete this part from outside the university network (you can use SSH tunnelling if needed)

1. Some versions of NetBeans do not come with the MySQL driver. You can check this by opening NetBeans, opening to the tab “Services”, selection the menu item “Drivers”, and checking if there is a MySQL driver. If there is a MySQL driver, then you can go straight to Step 3. Otherwise, you'll need to download a JDBC driver for MySQL from the MySQL web page. For convenience, a version of this driver is available here. Unzip the file to get a “jar” file.
2. Under NetBean's "Runtime" tab, open the `Databases` folder, Right click on `Drivers` and select "`New Driver`". Browse to the `mysql-connector-java-….jar` file that you downloaded. If you follow this procedure successfully, you will see that a MySQL driver has been added to the list of available drivers.
3. In the tab “Services” on NetBeans, right-click on `Database` and "`New Connection …`".

Name: `MySQL (Connector/J driver)`
Database URL: `jdbc:mysql://serverName:3306/yourDatabaseName`
Plus enter your MySQL login details.

- View the Tables, etc under your new connection. Right click on table to "`View Data …`".

JDBC

Next we'll create a simple Java application that connects to our database.

- In NetBeans, select "File/New Project …", and create a new Web Application for the Glassfish server. When giving a name to your Web Application, please give name like yourLogin_XXX, so that there is no name clash with other students when deploying the application on Glassfish.

- Right click on "Libraries" and select "Add Jar/Folder" to add the `mysql-connector-java` jar file that you downloaded.

- There are many ways to build a JSP or a Java class that interacts with a MySQL database. Here we will use a simple method – coding the database connection and data manipulation as plain Java code in the JSP. This is not necessarily the most elegant method – using JSTL tags is often more concise and readable, but it’s a good starting point so that you can get started.

- In the JSP add the following code:

First we need to register the MySQL JDBC driver:

```
Class.forName("com.mysql.jdbc.Driver");
```

Next we'll create a JDBC connection to the itb717u server:

```
Connection con =
DriverManager.getConnection("jdbc:mysql://serverName:3306/Database
Name", "UserName", "Password");
```

You can either fully qualify these names or simply `import java.sql.*;`

Next we use JDBC to create an SQL statement:

```
Statement stmt = con.createStatement();
```

Then we use that statement to execute an SQL query that returns a JDBC result set:

```
ResultSet rset = stmt.executeQuery("SELECT * FROM Customer");
```

We can iterate through the rows of the result set by calling its `next` method until it returns false.

```
while (rset.next())
    ...
```

For each row, we will print out the 3 columns:

```
    System.out.println(rset.getString(1));
    System.out.println(rset.getString(2));
    System.out.println(rset.getInt(3));
```

Finally we need to close the result set to free resources:

```
rset.close();
```

If you try to compile the above you will find that some of the methods throw various exceptions that must be either caught or declared to be thrown, so put the above in a try block that catches and displays the exceptions in question.

➢ Finally, build the project and deploy the generated web archive file (war) to the Glassfish server. As usual, make sure you name the package using your login as a prefix to avoid name conflicts... Once deployed, you can test the application.

Deploying on ats:

- For security reasons, the URL to the admin console, the login and the password are not made available in this tutorial handout. They will be given to you by the lab assistant.
    - Once you have logged into the admin console, go to Applications and click Deploy…
    - Find the war file in your local machine that contains your application. Usually, this is located in the `dist` directory of the project.
      Note: when the project is compiled, NetBeans will display the location of this directory.
    - In the field "Application name", please use the following naming convention: *yourLogin-nameOfTheApplication*, where yourLogin is your UT user name. Note that the Glassfish server is being used by all ESI students. So it is important that you follow this naming convention when deploying Web services to this shared Glassfish server. Otherwise there will probably be name clashes with other students.
    - Once you have uploaded the war file, browse to the "Web Services" link in the Glassfish admin console and click on Launch next to the name of your application.

Note: to deploy your application on your machine, it is necessary you are using eduroam wireless connection.

Java Persistence API (JPA)

JPA is a framework for managing relational data in Java applications in a quasi-transparent way. The central concept of JPA is that of an "Entity Class", which is a lightweight Java class whose state is stored in a database. The programmer specifies the mapping of object instances to relational tables by means of annotations and is not longer required to deal with JDBC code (as it is generated from the annotations).

As part of this lab session, you are asked to follow the tutorial available at:
http://www.1001-computer.de/persistence-tutorial-jpa-javase-netbeans/

To deploy the application to the server, it is necessary to create a JDBC resource in application server Glassfish to access the database (it shall has the same JDBC resource name of the connection that you used when developed the application):

- First create a Connection Pool, in the administration console select Resources -> JDBC -> Connection Pools -> New
    - Resource Type: javax.sql.DataSource
    - Database Vendor: Mysql
    - The required information is:
        - Url: jdbc:mysql://localhost:3306/*databaseName*
        - User
        - Password

- Finally, create a JDBC resource. Click on Resources -> JDBC -> JDBC Resources -> New

You now know how to use JPA from a console application. However, JPA can also be used from a Web application. Create a new Web application and add an Entity class to handle the information about a Product (i.e. name, description, and price). Add a frontend to your application.

Hint: Netbeans provides a wizard to generate a basic frontend: right-click on the project name and select "New -> JSF Pages from Entity Classes…" and follow the corresponding instructions.