Software

To finish the lab, you may need the following software:

1. NetBeans IDE version 8.2 or later (or, NetBeans)
2. JDK version 1.8.0 (jdk1.8.0_202, or later)
3. GlassFish Server Open Source Edition version 4.1.1 or later (or, GlassFish)

Aim

Programming the EJB Stateless Session Bean “MyuserBean” and its client application.

Figure 1 shows a rough design of the “MyuserAppClient” application and its related classes, including their roles. The “MyuserAppClient.java” acting as a client will handle the input from users to perform the CRUD operations of the MYUSER database table via the Myuser class (ORM).

As mentioned in Lab 03, we need a JPA controller for a better design. In Lab 03, we did this in a normal Java class. However, as this normal Java class is not managed by any server, we could not achieve scalability. In this lab, we will be programming a StateLess Session Bean (SLSB) to take on this role. As the SLSB is managed by the GlassFish server, scalability is achieved. You need to read about the “theory” stuff (in lecture notes) on how GlassFish server (or other Java EE compliant server) does all these to achieve “scalability”. When doing this lab using the “auto-code generation features” provided by NetBeans, it has its own naming convention. NetBeans called this SLSB, “MyuserFacade”, (“…Façade”) because it follows the famous Façade design pattern. Also, the naming conventions for the remote interface of the Façade is “…FacadeRemote” whereas the local interface is “…FacadeLocal”.

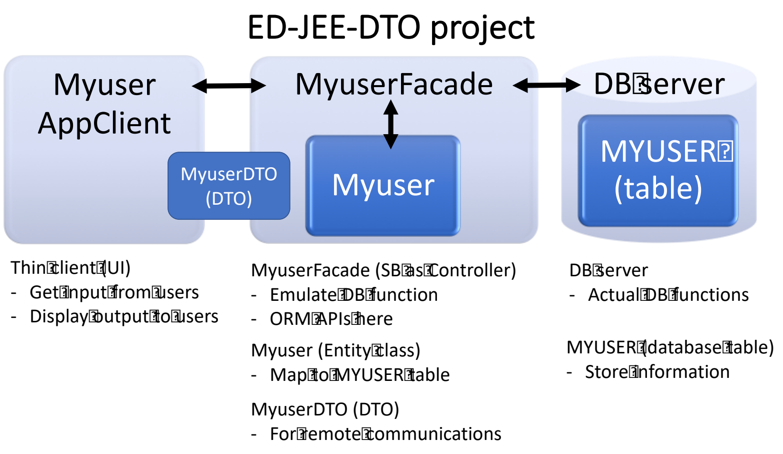

Figure 1 A rough design of MyuserAppClient, a ED-JEE-DTO project

Overview of Lab Tasks

In this Lab, you will learn to perform the following tasks in programming a stateless session bean:

LT1. Create a Java Class Library for the Remote Interface of the stateless session bean
LT2. Program a DTO class and put it in the Remote Interface project
LT3. Create an Enterprise Application project
LT4. Add an Entity class to the Enterprise Application’s EJB project
LT5. Create the Stateless Session Bean
LT6. Program the Remote business method in the Stateless Session Bean
LT7. Program the Remote Interface class of an EJB Stateless Session Bean
LT8. Program a client application that calls an EJB Stateless Session Bean
LT9. Deploy the Enterprise Application to the GlassFish server
LT10. Run the Enterprise Application Client

Lab Tasks

This Lab should be run on MS Windows Platform

LT1. Create a new Java Class Library Project in NetBeans called "`ED-JEE-DTO-RI`" to hold the remote interface of the stateless session bean

LT1.1 Select "File" > "New Project"
LT1.2 Choose "Java" > "Java Class Library", then click "Next"
LT1.3 Enter "`ED-JEE-DTO-RI`" in the "Project Name" field
LT1.4 Click "Finish"
*Note: The project has been created. We will just leave it as is. NetBeans will automatically put the required information in this project when necessary as we perform the learning task in this lab.

LT2. Program the Data Transfer Object (DTO) class for "Myuser.java"

LT2.1 Create a new Java class "MyuserDTO" in the package "entity" in the "ED-JEE-DTO-RI" project. (Right-click "ED-JEE-DTO-RI", choose "New"->"Java Class…" – refer to Lab01b LT4.2)
LT2.2 Make sure that "MyuserDTO" implements the "java.io.Serializable" interface
Hint: **Add** "**implements Serializable**" in between "public class MyuserDTO" and "{". Remember to fix imports.
LT2.3 Copy and paste the following code segments into "MyuserDTO" class

```
private final String userid;
private final String name;
private final String password;
private final String email;
private final String phone;
private final String address;
private final String secQn;
private final String secAns;
```

LT2.4 Generate a constructor
a. Right click within the "MyuserDTO" class, select "Insert Code…" > "Constructor…"
b. In the "Generate Constructor" window, click the "Select All" button > "Generate"
Note: NetBeans generates a constructor that sets all the instance variables from its parameters.
LT2.5 Generate the getters of these instance variables (Only getters are required. No setters.)
Note: NetBeans generates all the getters.

LT3. Create a new Enterprise Application Project in NetBeans called "`ED-JEE-DTO`"

LT3.1 Select "File" > "New Project"
LT3.2 Choose "Java EE" > "Enterprise Application", then click "Next"
LT3.3 Enter "ED-JEE-DTO" in the Project Name field, then click "Next"
LT3.4 Use the default server "GlassFish" and Java EE version "Java EE 7"
LT3.5 Make sure the following check boxes are checked
a. "Create EJB Module" – "ED-JEE-DTO-ejb"
b. "Create Web Application Module" – "ED-JEE-DTO-war"
LT3.6 Click "Finish"
*Note: There are three new "projects" in the "Projects" Windows. They are
a. ED-JEE-DTO (Enterprise Application Project)
b. ED-JEE-DTO-ejb (Enterprise Application's EJB Project) where our SLSB will sit
c. ED-JEE-DTO-war (Enterprise Application's Web Application Project) where our Web App will sit (see Lab 5)

LT4. Add an Entity class to the EJB Project

LT4.1 Select the "ED-JEE-DTO-ejb" project
LT4.2 Right click the mouse and select "New" > "Entity Class from Database…"
LT4.3 In the "New Entity Classes from Database" window, select "jdbc/__default" in the Data Source field and put the "MYUSER" table in the "Selected Tables" text area (Make sure the "Include Related Tables" check box is checked)
Note: If this is the first time you do this, NetBeans will try to get the connections to the database. It will display the "New Connection Wizard" window, you then need to check the following settings:

```
Driver Name:   Java DB (Network)
```

```
Host:         localhost
Port:         1527
User Name:    APP
Password:     ●●● (should be APP anyway)
JDBC URL:     jdbc:derby://localhost:1527/sun-appserv-samples;create=true
```

After checking these, try click on "Test Connection". Result should be "Connection Succeeded" – this is to make sure things can be done properly. If there is any connection problem, probably you need to sort it out first (e.g. database server has not been started or database connection strings while setting up the database in Lab_1a is not correct). There are so many reasons to go wrong, please do let me know and I will try my best to help; and then enrich this "Note".

In case, the "Test Connection" is successful. Then, Click "Next". You should select schema "APP" (already been pre-selected for you by NetBeans). NetBeans then displayed the connection name as "…".

Note: If you cannot find the "MYUSER" table in the "Available Tables:" field, make sure your database connection string (See JDBC URL above) is right and you have set up the database properly with the proper username "APP" and password "APP". You may want to remove the database and recreate one with the right settings. See Lab_01a_Setup_JavaEE.

Note: When asked about the database username and password, enter "APP" for both (or, whatever you put during the setting up of your own databases).

LT4.4 Click "Next"

Note: A new dialog box appears with "MYUSER" in the "Class Names" field and "ED-JEE-DTO-ejb" in the Project field

LT4.5 Enter "entity" in the "Package" field

LT4.6 Make sure that the following check boxes are checked

a. "Generate Named Query Annotations for Persistent Fields"

Note: By checking the box, NetBeans will create some standard named queries like

"Myuser.findAll"	to find all myusers in the database table
"Myuser.findByUserid"	to find a particular myuser using userid (userid is the primary key of the table)

… (others omitted) …

b. "Generate JAXB Annotations"

Note: By checking the box, NetBeans will create some standard annotations that allow the entity class to be used via JAXB web services. We do not need this actually.

c. "Create Persistence Unit"

LT4.7 Click "Finish"

Note: NetBeans will automatically generate the entity POJO "Myuser.java" for you, including some named queries.

Note: If you do not have any methods to update the data, this entity class is now complete.

Note: This entity class is a Data Access Object (DAO) that is responsible for holding the required information for the records in the MYUSER database table.

Note: In the source folder, you can see a "package" called "META-INF". The file "persistence.xml" is the persistence unit of Myuser, in xml format and act as a config file.

LT4.8 This task will fix some "unknown" bug in NetBeans

a. Expand on "Configuration Files…"

b. Double click on "persistence.xml"

c. In the "persistence.xml" pane, uncheck "Include All Entity …"

d. Click on "Add Classes…"

e. In the "Add Entity Class" window, select "entity.Myuser" and click "OK"

f. Click "Save"

LT5. **Create the Stateless Session Bean class, called "MyuserFacade"**

LT5.1 Select the "ED-JEE-DTO-ejb" project

LT5.2 Right click the mouse and select "New" > "Session Beans…"[1]

Note: If you cannot find "Session Beans…", select "Other…". Then, in the "New File" window, choose "Enterprise JavaBeans" in the "Categories:" selection box, then choose "Session Bean" in the "File Types:" selection box. Then, click "Next".

LT5.3 In the "New Session Bean" window,

a. Enter "MyuserFacade" in "EJB Name" text field

b. Enter "session" in the "Package" text field

c. Select "Stateless" for "Session Type"

d. Check the "Remote" checkbox in "Create Interface"

e. Select "ED-JEE-DTO-RI" in the "Remote in project" selection box

Note: When "Remote" is checked, NetBeans asks for a java class library project (in this case, select "ED-JEE-DTO-RI", the one we just created in LT1)

f. Click "Finish"

Note: NetBeans now generates the "MyuserFacade" class and put "MyuserFacadeRemote" class in the "ED-JEE-DTO-RI" project under "session" package.

LT5.4 Copy and Paste the following code segment into the MyuserFacade class

```
@PersistenceContext(unitName = "ED-JEE-DTO-ejbPU")
private EntityManager em;

protected EntityManager getEntityManager() {
    return em;
}

private void create(Myuser myuser) {
    em.persist(myuser);
}

private void edit(Myuser myuser) {
    em.merge(myuser);
}

private void remove(Myuser myuser) {
    em.remove(em.merge(myuser));
}

private Myuser find(Object id) {
    return em.find(Myuser.class, id);
}
```

Remember to fix imports and format the code.

Note: The line with "@PersistenceContext(unitName = "…") …" tells the GlassFish server to create the required entity manager when creating the SLSB object, most likely using the "EntityManagerFactory" API. As a developer, you are not required to program this part.

Note: We have all these methods declared as "private" because we do not want the actual DAO (Myuser) to be passed to the actual client. The DTO done early is responsible for such purposes.

Now, we are going to program the business methods for "MyuserFacade"

LT6. **Program the remote business method, "public boolean createRecord(MyuserDTO)"**, in the "MyuserFacade" class. It accepts a MyuserDTO object and tries to store the relevant information into the database. If a record with the same userid (the primary key) exists in the database, it does nothing and returns false. Otherwise, it creates the record in the database and return true.

LT6.1 Put your mouse inside the "MyuserFacade" class (that is, mouse cursor within the braces "{" and "}") in the editor pane

LT6.2 Right click the mouse and Select "Insert Code…" > "Add Business Method…"

Note: If you put the mouse in the wrong place in LT6.1, you will get some other options but lacking the "Add Business Method…"

[1] We do not use "Session Beans from Entity Classes…" option due to some unknown NetBeans issues.

LT6.3 In the “Add Business Method…” window,

- a. Enter “createRecord” in the “Name” field (This specifies the name of the method)
- b. Enter “boolean” in the “Return Type” field
- c. Select the “Parameters” tag and click “Add”
  1. Enter “myuserDTO” under the “Name” column
  2. Enter “MyuserDTO” under the “Type” column
- d. Click “OK”

Note: NetBeans creates the method in the “MyuserFacade” class and a declaration in the “MyuserFacadeRemote” interface class. Remember to fix the imports of “MyuserDTO” in these two classes.

LT6.4 Copy and paste the following code segment into the “createRecord” method

```
if (find(myuserDTO.getUserid()) != null) {
    // user whose userid can be found
    return false;
}

// user whose userid could not be found
try {
    Myuser myuser = this.myDTO2DAO(myuserDTO);
    this.create(myuser);    // add to database
    return true;
} catch (Exception ex) {
    return false; // something is wrong, should not be here though
}
```

LT6.5 Copy and paste the following code segment into “MyuserFacade”

```
private Myuser myDTO2DAO(MyuserDTO myuserDTO) {
    Myuser myuser = new Myuser();
    myuser.setUserid(myuserDTO.getUserid());
    myuser.setName(myuserDTO.getName());
    myuser.setPassword(myuserDTO.getPassword());
    myuser.setEmail(myuserDTO.getEmail());
    myuser.setPhone(myuserDTO.getPhone());
    myuser.setAddress(myuserDTO.getAddress());
    myuser.setSecqn(myuserDTO.getSecQn());
    myuser.setSecans(myuserDTO.getSecAns());
    return myuser;
}
```

LT6.6 Remember to save both files

Note that: In this Lab task, you have learned how to write the Remote interface class
If you used “Insert Code…” > “Add Business Method…” in LT6.2, all business methods’ signatures will be added to the corresponding Remote Interface class. Hence, no need to do anything. Otherwise, you have to manually type in the relevant method signatures in the Remote interface class.

Note: Even though the description is for Remote interface class, the same is also true for “writing the Local interface class”.

LT7. In this Lab task, you will learn how to program the application client “MyuserAppClient” to access the Stateless Session Bean “MyuserFacade”

LT7.1 Create the "ED-JEE-DTO-appclient" Enterprise Application Client project

- a. Select “File” > “New Project”
- b. Choose “Java EE” > “Enterprise Application Client”, then click “Next”
- c. Enter “ED-JEE-DTO-appclient” in the Project Name field, then click “Next”
- d. Use the default server “GlassFish” and Java EE version “Java EE 7”

  Note: There is no need to add the project to an enterprise application.
- e. Change the Main class to “edjee.MyuserAppClient”
- f. Click “Finish”

Note: NetBeans now creates the ED-JEE-DTO-appclient (Enterprise Application Client Project) project.

LT7.2 Now, we need to add a reference to the Stateless session bean "MyuserFacade" in the EJB project in the application client "MyuserAppClient.java" class

a. In the Projects window, select "ED-JEE-DTO-appclient" project
b. Expand "ED-JEE-DTO-appclient" project > "Source Packages" > "edjee" > "MyuserAppClient.java"
c. Open "MyuserAppClient.java" in the editor window
d. Locate the mouse within the "MyuserAppClient" class
e. Right click the mouse and select "Insert Code..." > "Call Enterprise Bean…"
   Note: If your mouse is in the wrong position, you won't be able to see the right option.
f. In the "Call Enterprise Bean" window, expand on "ED-JEE-DTO-ejb" node and select "MyuserFacade" (remember this is an EJB)
g. Select "Remote" in the "Referenced Interface" (or, it has been selected for you)
h. Click "OK"

Note 1: NetBeans will create the following annotation for you "@EJB private static MyuserFacadeRemote myuserFacade;" (This actually creates a remote reference to the actual EJB SLSB.)

Note 2: NetBeans also adds the "ED-JEE-DTO-RI.jar" class library file as a project library.

LT7.3 Copy and paste the following segment of code to "MyuserAppClient.java"

```
public MyuserAppClient() {
}

public static void main(String[] args) {
    MyuserAppClient client = new MyuserAppClient();

    // assuming inputs from keyboard or any GUI
    MyuserDTO myuserDTO = new MyuserDTO("000001", "Peter Smith", "123456",
                "psmith@swin.edu.au", "9876543210", "Swinburne EN510f",
                "What is my name?", "Peter");
    boolean result = client.createRecord(myuserDTO);
    client.showCreateResult(result, myuserDTO);

    // assuming inputs from keyboard or any GUI
    MyuserDTO myuserDTO2 = new MyuserDTO("000007", "David Lee", "654321",
                "dlee@swin.edu.au", "0123456789", "Swinburne EN510g",
                "What is my name?", "David");
    result = client.createRecord(myuserDTO2);
    client.showCreateResult(result, myuserDTO2);
}

public void showCreateResult(boolean result, MyuserDTO myuserDTO) {
    if (result) {
        System.out.println("Record with primary key " + myuserDTO.getUserid()
            + " has been created in the database table.");
    } else {
        System.out.println("Record with primary key " + myuserDTO.getUserid()
            + " could not be created in the database table!");
    }
}

public Boolean createRecord(MyuserDTO myuserDTO) {
    return myuserFacade.createRecord(myuserDTO);
}
```

Remember to "Fix Imports", format the code, and finally, save the file

LT8. In this lab task, you will learn how to deploy the enterprise application "ED-JEE-DTO" project

LT8.1 Select "ED-JEE-DTO" enterprise application project in the "Projects" window

LT8.2 Right click your mouse and select “Deploy” (You may want to select “Clean and Build” first to clean up any mess that has been done.)

Note: In case you encounter problems in deploying and you are very sure that everything is right (that is you follow the lab instructions to the letter), you may want to try "Clean and Build" on the Enterprise Application EJB project first and then "Deploy" on the Enterprise Application project. [This is how I overcome the deployment problem while preparing this lab sheet. :-) Reason is unknown to me. Once I did that, I have no further deployment issues. Good Luck!]

LT9. Run the enterprise application client “ED-JEE-DTO-appclient” project

LT9.1 Select “ED-JEE-DTO-appclient” enterprise application client project in the “Projects” window

LT9.2 Right click the mouse and select “Run”

Now, observe the outputs of “MyuserAppClient” in the “ED-JEE-DTO-appclient (run)” tab of the “Output” pane. It should show something like the following (assuming you have not changed the content in the database after you completed Labs 02 and 03)

```
“Record with primary key 000001 could not be created …”
“Record with primary key 000007 has been created …”
```