Assignment 2 Instructions

Nash Dean

Travis Dietz

1. Unzip the projects into the assignments directory
2. Start IntelliJ and open the projects
3. Make sure SQL Server is running
    i. Go to “Services” on your Windows computer and restart SQL Server (MSSQLSERVER) if it is off
    ii. Go to SQL Server Configuration Manager > SQL Server Network Configuration > Protocols for MSSQLSERVER and enable TCP/IP.
4. Check to make sure you have created a database with tables filled with data

The following steps will allow you to build the project from scratch. As long as the client class, bean class, and remote interface are built with the following dependencies attached to the appropriate modules, as well as the artifact, the project will not need to be rebuilt from scratch:

Asg2 > src

- **RPClient Class**
    - **RemoteLib**
    - **Appserv-rt.jar**

RegistrationBean > src > main > registrationbean

- **RPBean Class**
    - **The default dependencies included in the EJB Enterprise Java Build**
    - **dblib.jar**
    - **RemoteLib**

RemoteLib > src > remotelib

- **IRemoteRP Interface**
    - **Maven: javax.ejb:javax.ejb-api:3.2.2**

Asg2 > out > artifacts

- **RegistrationBean**

    <output root>

    - **Extracted ‘dblib.jar/’**
    - **‘RegistrationBean’ compile output**
    - **‘RemoteLib’ compile output**

5. Build the Project
    a. Verify that you have three separate modules (One for the client (should be the default module - asg2), one for the bean, and one for the remote library).
        i. Project Structure > Modules > New Module > Java
            1. Name one of the Modules “RegistrationBean”
                a. Choose “Java Enterprise” (Make sure the template is *Web application* and Application Server is *GlassFish 4.1.2*)
                b. Change the specifications from *Servlet* being ticked to *Enterprise Java Beans (EJB)*
                c. Name the module and set the group to the company domain
            2. Name the second module “RemoteLib”
    b. Edit the module dependencies for “RemoteLib”
        i. Project Structure > Modules > RemoteLib > Library
            1. Select Mavin: javax.ejb:javax.ejb-api:3.2.2
            2. Check the Export ticker
    c. Create the RemoteLib Interface under the RemoteLib folder’s src folder
        i. Right click → New **>** Java Class **>** Interface
        ii. Name it “remotelib.IRemoteRP”
        iii. Declare the database method headers in the interface
        iv. Import the javax.ejb.Remote
        v. Annotate the interface “@Remote”
    d. Add the bean’s dependencies under Project Structure
        i. Project Structure > Modules > RegistrationBean > Module Dependency
            1. Select “RemoteLib”
        ii. Project Structure > Modules > RegistrationBean > JARs or Directories
            1. Locate the dblib JAR from assignment 1
                a. Will be in the asg1 Out > Artifacts folder as “dblib.jar”
            2. Select the “dblib.jar”
            3. Right click “dblib.jar” → Choose “Move to Project Libraries”
                a. Name “dblib” and check “Move library files to:” so that the library is under the *lib* folder.
    e. Create the bean class under the RegistrationBean module main > java > registration
        i. Annotate class “@Stateless”
        ii. Set the name to “RPEJB”
            1. This is the name the naming service at port 3700 looks for to align the bean with the client
        iii. Import the remote library “remotelib.IRemoteRP”
        iv. Create a constructor that initializes the Registration class from the imported dblib.jar
            1. Should connect to the database by initializing the login and password parameters

v. Override the methods from the remote interface

f. Create Client Class in main src folder module
   i. Name it “RPClient”
   ii. Make sure there is a Remote object initially set to null
   iii. Try to initialize a connection to the bean using the remote interface with an InitialContext object from the Initial Context class
      1. Use this classes *lookup* method passing the name of the remote “remotelib.IRemoteRP” as a parameter
         a. This reaches out to the naming manager in the server
      2. Cast this as a remote object (a proxy is returned)

g. Add the RemoteLib and *appserv-rt.jar* as dependencies for the Client Class
   i. Project Structure > Modules > RegistrationBean > Module Dependency
      1. Select “RemoteLib”
   ii. Project Structure > Modules > RegistrationBean > JARs or Directories
      1. Locate the *appserv-rt.jar* from the GlassFish4 subdirectory *lib*
      2. Select *appserv-rt.jar*

h. Create a bean artifact
   i. Project Structure > Artifacts > EJB Application: Exploded
   ii. Name the EJB Application “RegistrationBean”
   iii. Move the *Extracted ‘dblib.jar/’, ‘RegistrationBean’ compile output, and ‘RemoteLib’ compile output* to the <output root>
   iv. Check/Tick *Include in project Build*

6. Add SQLServerDriver Dependency if not already done
   a. Copy the JAR “mssql-jdbc-9.2.1.jre8.jar” file and move it to the GlassFish4 subdirectory *lib*

If the following configurations do not exist or have mistakes, edit the configurations as the following:

7. Run the “GlassFish 4.1.2 Configuration”
   a. Edit Configurations (top right of IntelliJ) > Add New Configuration > GlassFish Server
      i. Name it “GlassFish 4.1.2”
      ii. Click on “Configure…” to the right of “Application Server:”
      iii. Select the “+” Button on the top left
      iv. Select the folder icon that is to the right of the “GlassFish Home” textbox
      v. Locate where the “glassfish4” folder is on your computer
         1. Click “Ok”
      vi. Click “Apply” and “Ok”
      vii. Select “Default” for the JRE (Default for this project should be 1.8 - project SDK)
      viii. Select “domain1” from the “Server Domain” drop-down box
         1. The Username should appear as “admin”
      ix. Select the “Deployment” tab

1. Ensure that “RegistrationBean” appears in the “Deploy at the server startup” box.
    a. If not, select the “+” button
    b. Find glassfish4 in your file directory, open it
    c. Find glassfish, open it
    d. Find lib, open it
    e. Select “appserv-rt.jar,” select “ok”
    f. Click “Apply” and “Ok”

b. Press Hammer Symbol (“Build Project”) CTRL+F9
c. Run the Configuration (“Run ‘ClientApp’) SHIFT+F10
d. Explore the Application

8. Deploy the project/Run the Client
    i. Edit Configurations (top right of IntelliJ) > Add New Configuration > Application
        1. Name it “Client”
        2. Set the Main Class to “Client” and specify the module as “asg2”
        3. Click ”Apply” and click “Ok”
    ii. Press Hammer Symbol (“Build Project”) CTRL+F9
    iii. Run the Configuration (“Run ‘ClientApp’) SHIFT+F10

Client

Follow the messages/instructions provided by the application client

1. You will be prompted to enter a ‘Year’ and ‘Semester’ to display a list of available, non-full classes

    a. Valid ‘Semesters’ are Fall and Spring

3. Type “L” and hit enter to List a student’s classes. You must then enter their student number and the year and semester you are interested in seeing.

4. Type “R” and hit enter to register a student for a class. You must enter their student number, a course number, and section number.

    a. The business logic makes sure the student is valid, not already registered for the course, and that the course is not at capacity.

5. Type “Q” and hit enter to quit the client application.

Registration

List (String year, String semester)

1. The first List method takes two String parameters, year and semester. It formats a SQL query as a String that is executed by a method from the Statement class and stores the values that are returned as a ResultSet. The query selects the Sections of classes that are not at capacity (the available classes) during a given year and semester. These values are then retrieved and stored as strings in a while loop, and formatted into a table. The values are *courseNo, sectionNo, room, days, and time*. If no SQL Exceptions occur, then this is what will be returned as a formatted String.

List (int StuNo, String year, String semester)

2. The second List method takes an int parameter, StuNo, and two String parameters, year and semester. It formats a SQL query as a String that is executed by a method from the Statement class and stores the values that are returned as a ResultSet. The query returns a students course schedule for a given semester and year. These values are then retrieved and stored as strings in a while loop, and formatted into a table. The values are *courseNo, sectionNo, room, days, and time*. If no SQL Exceptions occur, then this is what will be returned as a formatted String.

Register (int StuNo, String CourseNo, String SecNo)

3. The Register method takes an int parameter, StuNo, and two String parameters, CourseNo and SecNo. It creates three prepared SQL Statements. The first gets the TotalEnrolled and Capacity for the course number and section number that was passed through the method. This allows the method to check to make sure the course the student wishes to be added to isn't already at capacity. The second prepared statement query gets the student numbers for the students registered for the course number and section number provided. This allows the method to compare these students with the student number passed through the method so that there is no double registration. The third prepared statement simply gets all students enrolled in the school. This lets the method check to make sure the student number passed is a valid student. After making these checks, the method updates the Section table by adding to the TotalEnrolled column and inserts a new row into the Enrollment table. The method returns the number of rows modified or created.

IRemoteRP

4. The interface IRemoteRP is a remote interface on the client application that stores 3 method calls (from the bean, but does not store any logic). The class is denoted as remote via the ”@Remote” Session Bean designation above the Java Class header. IRemote contains the business logic Register method and the two List methods (which have different parameters). When the remote methods are called, they send an interface request to the App Server and then receive a proxy stub in return. This proxy stub enables the client to remotely use methods that are stored in the bean.

RegistrationBean

5. The RegistrationBean class “RPBean” implements the IRemoteRP interface via @override. RPBean is a “Stateless” type of EJB bean, meaning that no state is tracked during the lifecycle of this bean. This class instantiates the Registration class attrbute mdb, which stores the login information to MSSQL Server. If mdb is null, the system prints “Connection null.”