MONASH University

FIT5042 Studio 4

Advance Applications of Java Persistence

Last updated: 29 Feb 2020

OBJECTIVES

- The purpose of this tutorial is to introduce data persistence in Java, using JPA to communicate with a relational database, and more advanced ORM
- Understand Criteria API
- Understand the Container Managed Entity Manager
- Understand the difference between the Application Managed and Container Managed Entity Manager
- How to use the criteria API to generate Java Persistence Query Language
- Understand the difference between JPQL and SQL
- Understand the benefit of using a connection pool instead of not using one
- Understand more about the Java EE architecture: web client, named beans, EJB, dependent injection and related annotation, etc
- Understand more about data validations in JSF

INSTRUCTIONS

- You are encouraged to use your own laptop or device to do the tasks
- If you do not bring your own device, feel free to use the lab machine.

EFOLIO TASK 4 (PASS LEVEL)

DESCRIPTION:

- Become familiar with Java Persistence API and some of its advanced features

WHAT TO SUBMIT (INDIVIDUAL):

- Modified ***PropertyRepository.java*** file highlighting changes with comments
- Modified ***JPAPropertyRepositoryImpl.java*** file highlighting changes with comments
- Modified ***Address.java*** file highlighting changes with comments
- Modified ***ContactPerson.java*** file highlighting changes with comments
- Modified ***Property.java*** file highlighting changes with comments
- Submit a screenshot of the ***Edit screen***
- Submit a screenshot of the ***Search Property by Id*** functionality
- Submit a screenshot of the ***Search Property by Budget*** functionality
- Make sure the screenshot shows what your application is doing. There is no need to tidy the screenshot, some background, such as your name on the eFolio will add authenticity to the image. Take care to not submit any confidential information in the background

MONASH University

EVALUATION CRITERIA:

- Website UI and data retrieving work properly
- Search Property by ID and Budget works properly
- CRUD functions work as expected
- Tidy and structured code
- The result is shown as expected
- Additional tasks

STUDIO ACTIVITIES

Task 4 Exercise Advanced Application of Java Persistence

You will update the Real Estate Agency System making use of advanced features of the Java Persistence API for Database Connectivity.

Update the Real Estate application to provide functionality to Store property and Address data in a single table, add functionality to Search for a property by property Id, and add functionality to Search for a property by budget (price).

Please keep in mind that all the exercises are due for feedback at the beginning of the week, Monday in the following week. These will be assessed at the end of semester, along with your learning summary. You should ask for help BEFORE the class if you have any difficulties.

Please download the starter file (a zip file) called ***Week 4 Tutorial Exercise Students Version*** from Moodle.

Remember you need to choose the development workspace in Eclipse for your projects.

You may create each project as shown in Table 1 below using Eclipse, then copy the corresponding files from the starter file to the project structure as shown in Figure 1 below.

Before we start, you will need to understand how each project will depend on each other. If you do not, you will have trouble troubleshooting. The structure of your solution is similar to what we have done in the previous weeks, so you may need to refer to the tasks in the previous weeks.

Table 1 Provided project files for Tutorial 4

Project folder Name	Description
W4ExeStudent-war	The WAR tier of the JEE project
W4ExeStudent-ejb	The EJB of the JEE project
W4ExeStudent-common	The common library project which is shared among different tiers

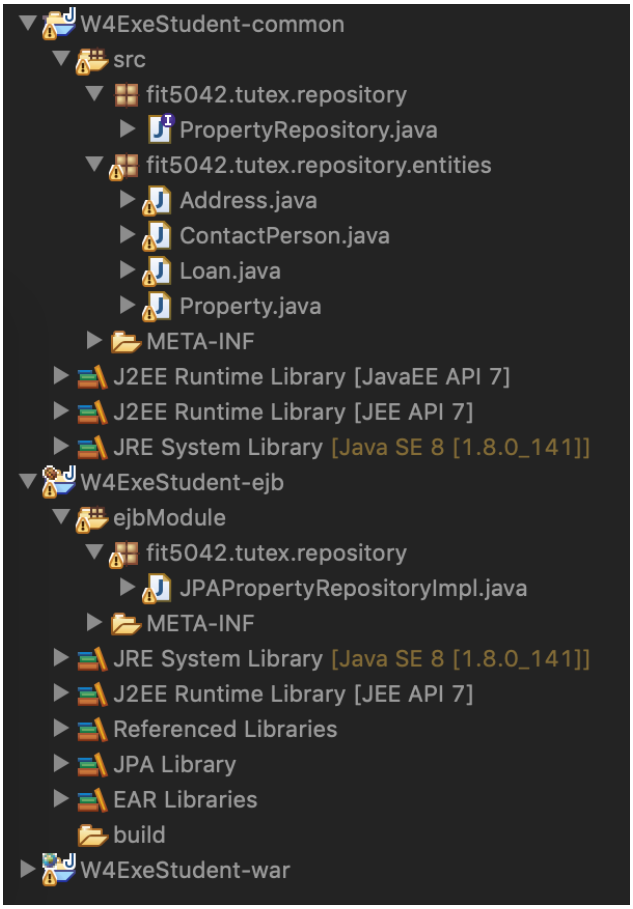


Figure 1 Overview of the project structure

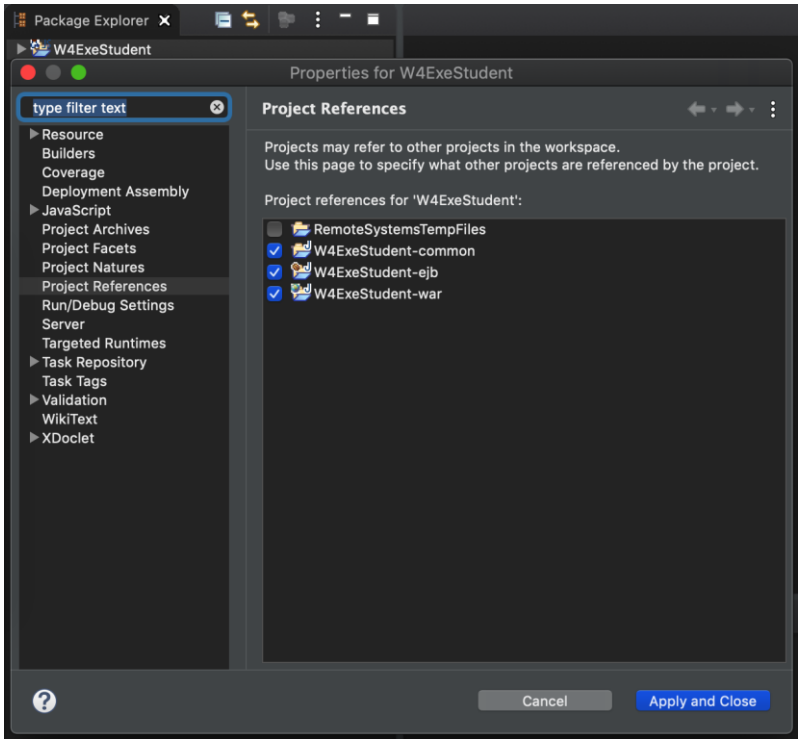


Figure 2 Enterprise application project references

Your solution for this week will be assembled and deployed as an Enterprise Application. The Enterprise Application is NOT provided. You will need to create your own Enterprise Application project. Your Enterprise Application project should be named W4ExeStudent and include all other three projects.

So, our Enterprise Application for this week will consist of

1) The EJB Module
2) The Web Client (WAR client) Module
3) The common shared project

Both EJB and Web Client will include the Common project.

Task Description

In this tutorial, you are required to do the following

1) In ***W4ExeStudent-common***, the code in file ***PropertyRepository.java*** is incomplete**.** Please complete the code and make it correct.
2) In ***W4ExeStudent-common***, apply necessary annotations in ***Property*** and ***Address*** entities so that the information of an address will be stored as part of the property record the address refers to in the same table. The address of a property must NOT be stored in a separate table and you must not change the data type of the attribute address in Property class. *The comments about where you need to apply annotations in Property and Address entities are given in the provided code*.
3) In ***W4ExeStudent-common***, annotate the attribute tags in ***Property*** entity so that the tags of a property will be stored in a table called ***TAG***. The tags of a property should be eagerly fetched, and the value of each tag must be stored in a column ***VALUE*** in the ***TAG*** table.
4) In ***W4ExeStudent-common***, enforce the relationship between a property and its contact person using annotation(s). Each property has one and only one contact person. Each contact person might be responsible for zero to many properties.
5) In project ***W4ExeStudent-ejb***
    i. Implement ***searchPropertyByBudget(double budget)***, which is a new method that will return the properties whose price is less than or equal to the budget that is passed as parameter. The method must be completed using Criteria API.
    ii. Complete the rest of the methods defined in ***JPAPropertyRepositoryImpl*** class. You must use container managed entity manager to complete those methods.

It is required for you to add a persistence unit at project ***W4ExeStudent-ejb*** as well as establish the database. Do not create a persistent unit elsewhere. Please follow the step by step guide as follows.

Since we are now using the container managed entity manager approach, a different method of creating the resources will be needed.

Remember that you are supposed to run the project from the Enterprise Application Project (***W4ExeStudent***) and not anywhere else.

If you have any concerns regarding implementing these classes, please refer to the Guidance part provided at the end of this document.

Please do not proceed further if you have not completed the modification of the above-mentioned files.

Creating the derby database

Step 1

Before creating the database, we need to first download the derby database libraries. Please click here to download the dependencies (or you can find it provided on Moodle, named JEE_Library.zip). After downloading, unzip the zip file and put it in your development workspace. For example, if your workspace Eclipse is “FIT5042”, then you will put the unzipped folder into this workspace so that it can be accessed by your application.

Please note that the path of your development workspace shouldn’t contain any space.

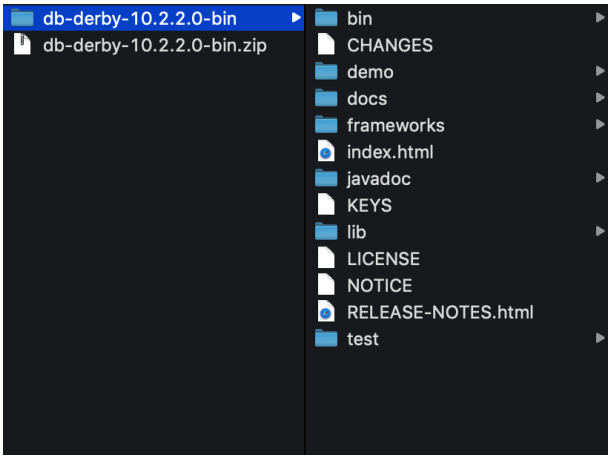


Figure 3 Derby Library

Go back to your Eclipse, make sure you have Data Source Explorer opened in the bottom panel. If not, please go to **Window** → **Show View** → **Other...** → Search for "**data**" → Choose "**Data Source Explorer**" → Click **Open**.

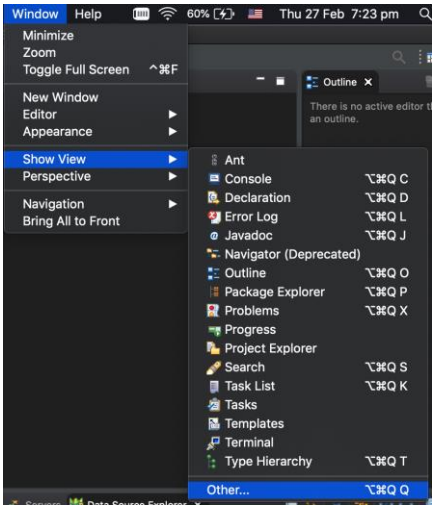


Figure 4 Show Data Source View

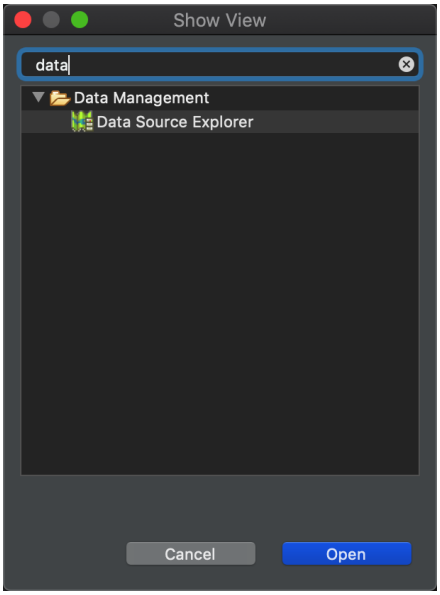


Figure 5 Find Data Source Explorer

Now, you should be able to see the Data Source Explorer window in the bottom panel.

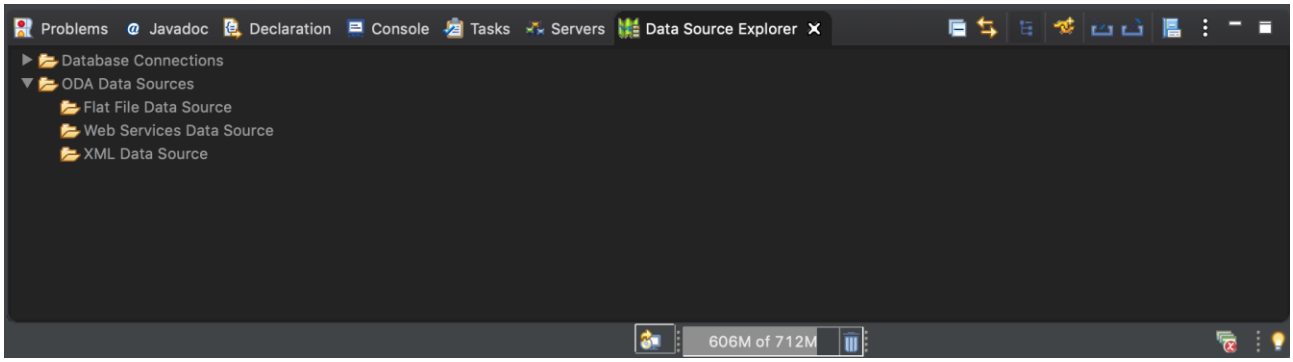
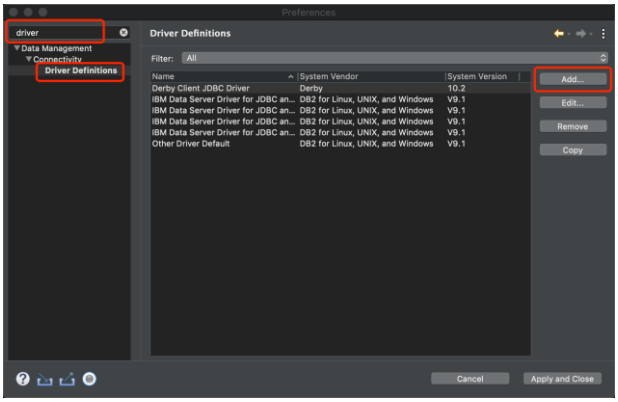


Figure 6 Data Source Explorer window

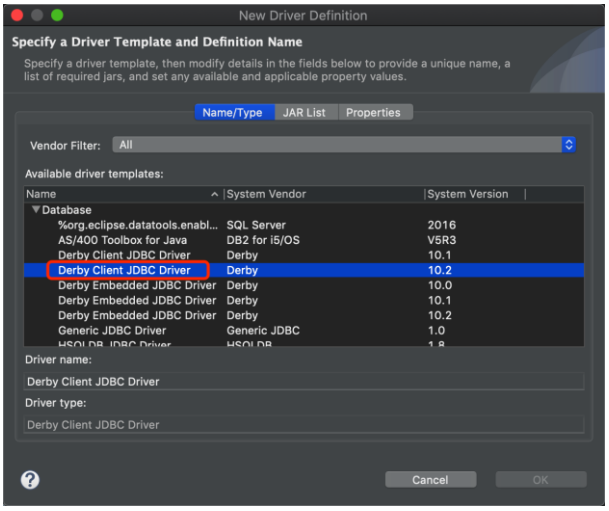


Step 2

Go to the **Window** → ***Preference***, search for the "***driver***" menu, click "***Driver Definitions***".

Here if your Eclipse is fresh installed, you need to define the ***Derby Client JDBC Driver***.

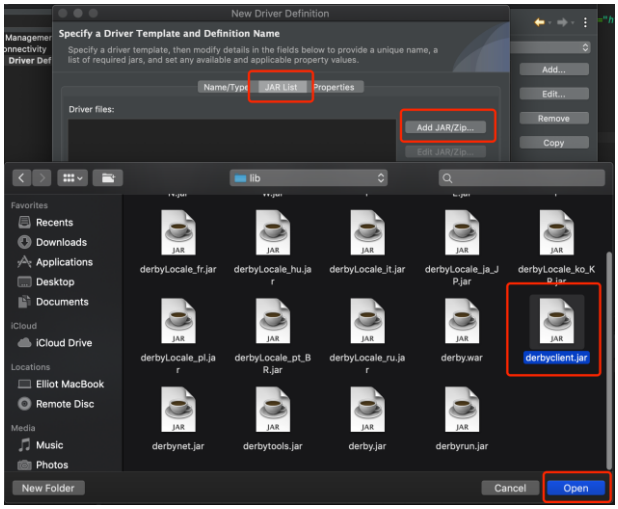
Click ***Add...***



Step 3

In the ***Name/Type*** panel, choose "***Derby Client JDBC Driver***" with the version of ***10.2***, leave the driver name and driver type as default.

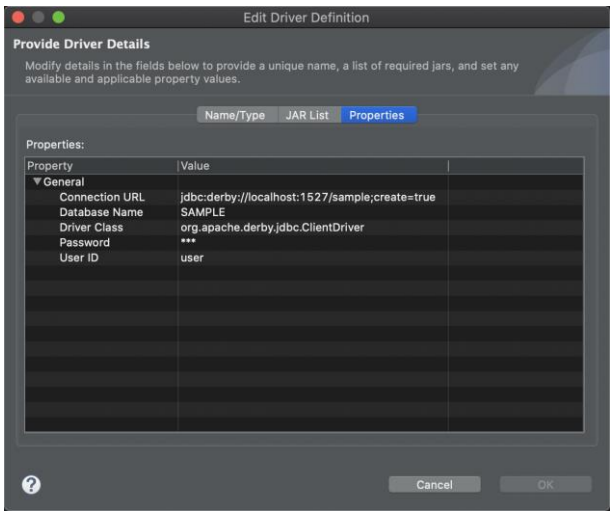
Click on the next Panel which is "***JAR List***".



Step 4

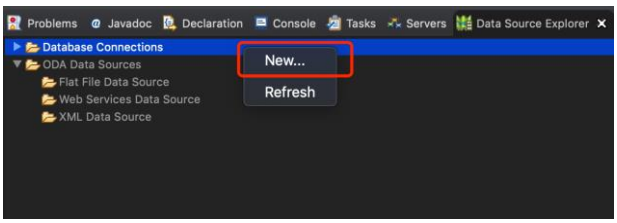
Click "Add JAR/Zip...", go to the derby database library folder, for example, ***[Your workspace path]/db-derby-10.2.2.0-bin/lib***

And then, choose the "derbyclient.jar" as our resource file.



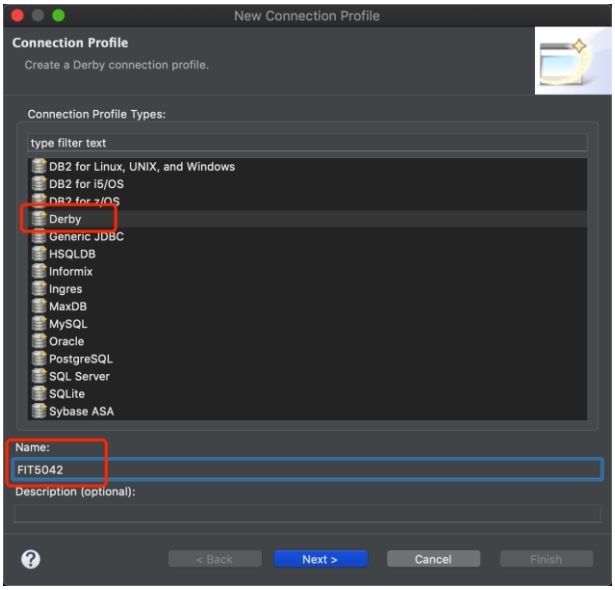
Step 5

Next, go to "***Properties***" panel. You can keep the default settings for these properties as this is just a template. It is not necessarily to modify these attributes values. After reviewing all the settings, click "***OK***"



Step 6

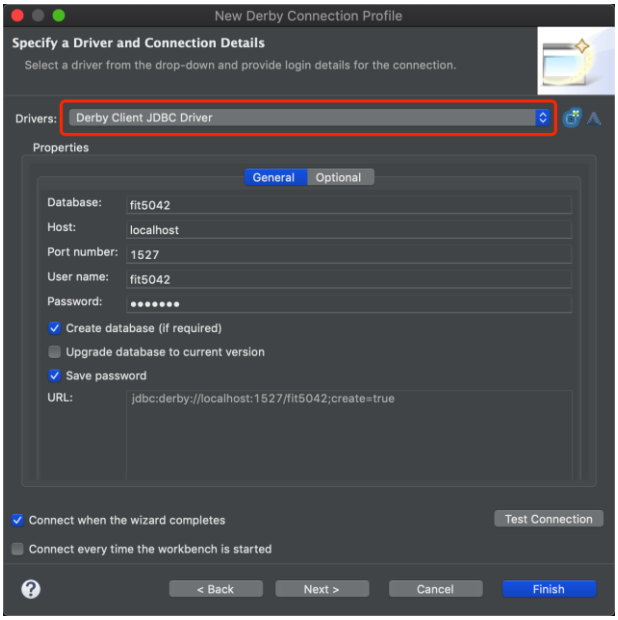
Go back to "Data Source Explorer", right click on the ***Database Connection*** folder shown in the Data Source Explorer window and select ***New...***



Step 7

Choose ***Derby*** as our database type and put **fit*5042*** as the name of our database.

Click ***Next >***



Step 8

Choose the driver that we've just created which is ***Derby*** **Client JDBC Driver**.

Make sure all properties are as the same as they are in the snapshot.

Database: *fit5042*

Host: *localhost*

Port number: *1527*

User name: *fit5042*

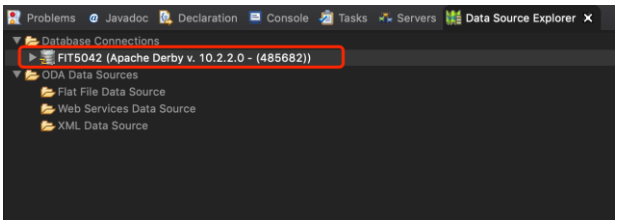
Password: *fit5042*

Create database (if required): *checked*

Save password: *checked*

URL: *jdbc:derby://localhost:1527/fit5042;create=true*

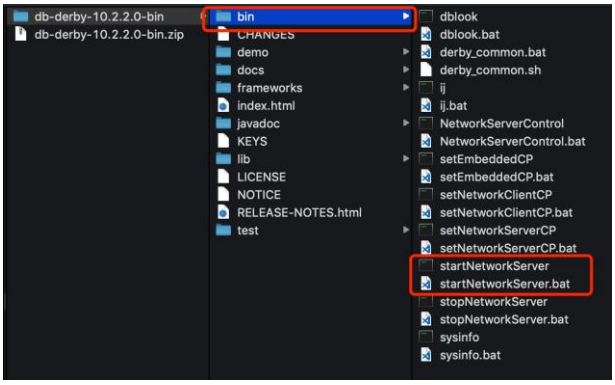
Click "***Finish***"



Step 9

Now, you should be able to find the database that we've just created when you open the ***Database Connections*** folder.

Currently, if you try to connect to the database, it will show an error message. That is because we haven't started our database.



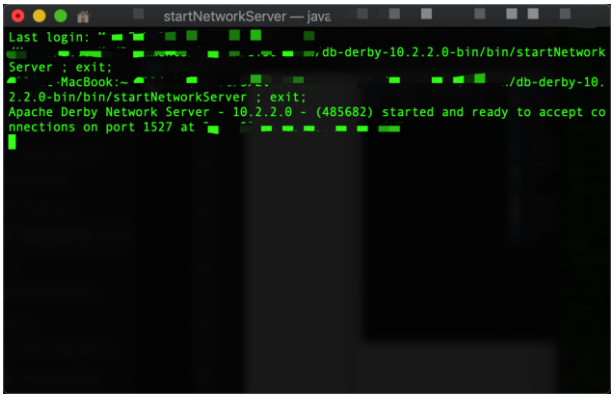
Step 10

In order to start the database, we need to go to the derby database library folder, under /bin folder:

- Mac user: run "***startNetworkServer***"
- Windows user: run "***startNetworkServer.bat***"

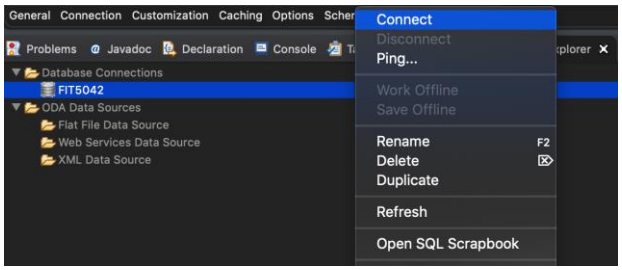
You might find some system issues here. This is mainly because you haven't setup the system variable and path for the derby database and Java.

Please ask your tutor for additional help if you encounter running database issues here. Alternatively, you can get some help by looking at this link: click here.



Step 11

After you successfully started the database, you should be able to see this message in your terminal.



Step 12

Go back to your Eclipse, right click on the database and choose "Connect".

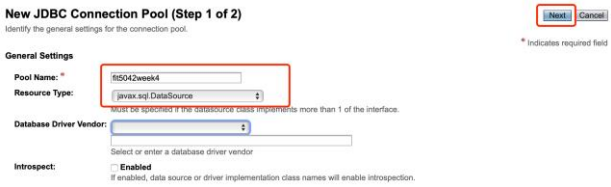
Congratulations, you have successfully connected to the derby database.

Creating the jdbc connection pool and jdbc resource for glassfish server

Step 1

Go to the Admin console of your glassfish server.

Expand the ***JDBC*** → Choose ***JDBC Connection Pools*** → Click ***New***

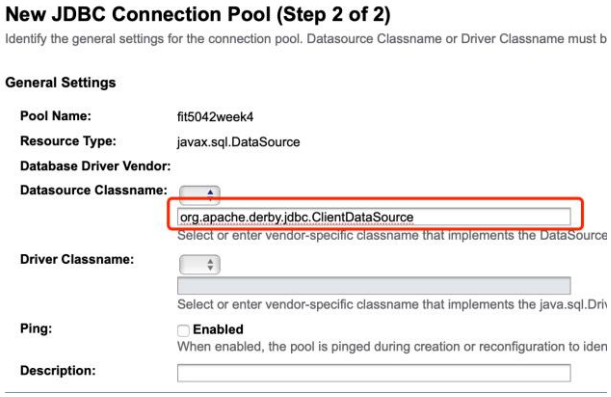


Step 2

Pool Name: *fit5042week4*

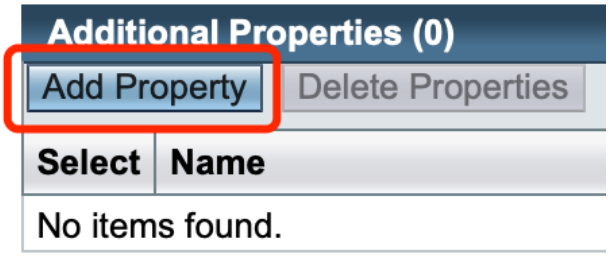
Resource Type: *javax.sql.DataSource*

Click ***Next***



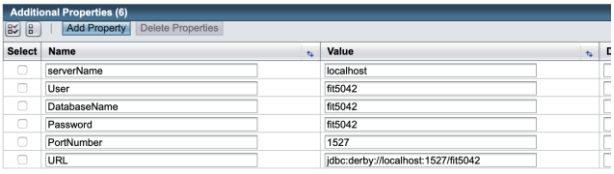
Step 3

Datasource Classname: *org.apache.derby.jdbc.ClientDataSource*



Step 4

Scroll down till the bottom, click ***Add Property***



Step 5

Put these attributes into the table, the order does not matter.

URL: *jdbc:derby://localhost:1527/fit5042*

PortNumber: *1527*

Password: *fit5042*

User: *fit5042*

DatabaseName: *fit5042*

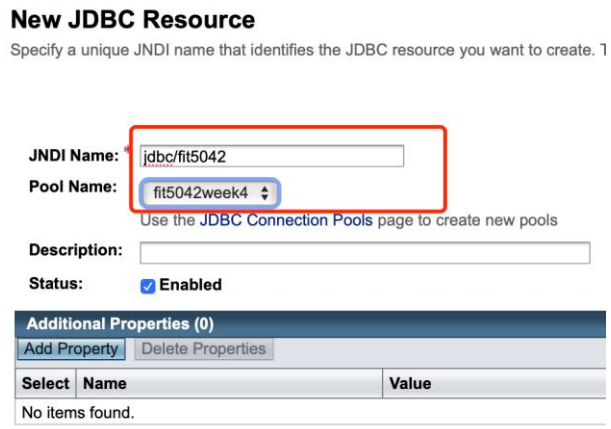
serverName: *localhost*

Click ***Finish***

Note: you may need to restart GlassFish server or Eclipse if the new connection pool '*fit5042week4*' is not appearing.

Step 6

Go to ***JDBC Resources***, click ***New***

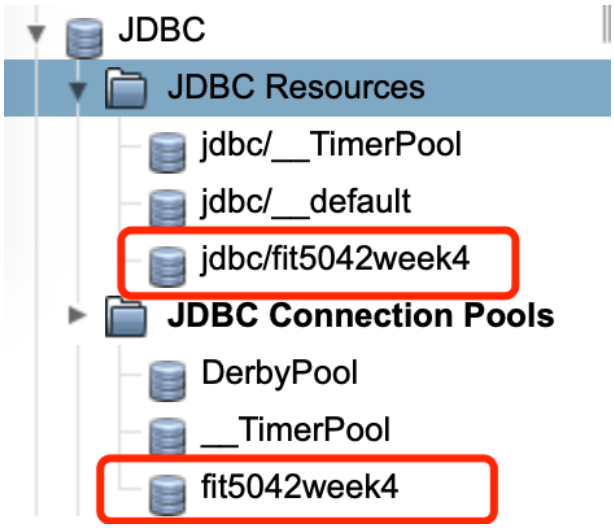


Step 7

JNDI Name: jdbc/*fit5042week4*

Pool Name: *fit5042week4*

Click ***OK***



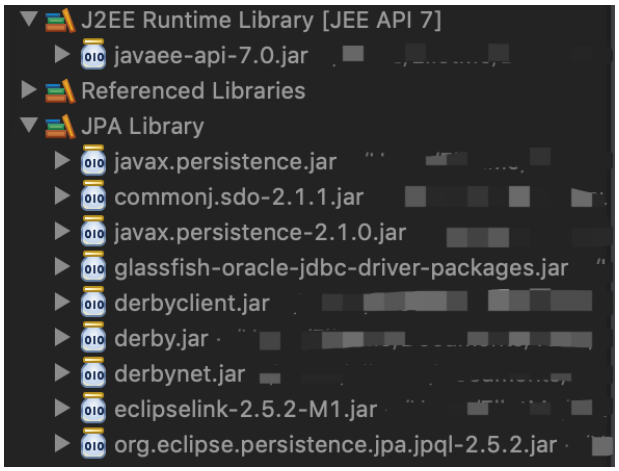
Step 8

Now, if you expand both ***JDBC Resources*** and ***JDBC Connection Pools***, you should be able to see a similar structure as showing in the snapshot.

Note: you may need to restart GlassFish server or Eclipse if the new JDBC Resource '*jdbc/fit5042week4*' is not appearing.

Till this step, you've successfully created the jdbc resources link to the corresponding jdbc connection pool.

Creating the persistence unit to connect our JDBC resource



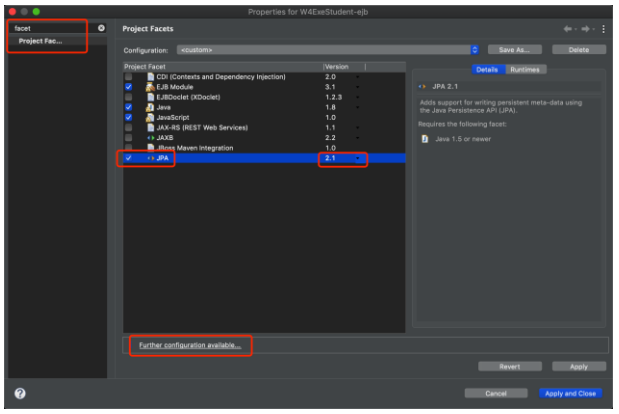
Step 1

Please make sure you have imported all these dependencies as expected in the EJB project. Otherwise, you will encounter some errors when you deploy your application.

All the dependencies are provided as a zip file (named JEE_Library.zip) which can be downloaded on the Moodle.

The following instructions will give you some guidance about how to set up the JPA Library.

If you have any import issues with dependencies, please ask your tutor for help.



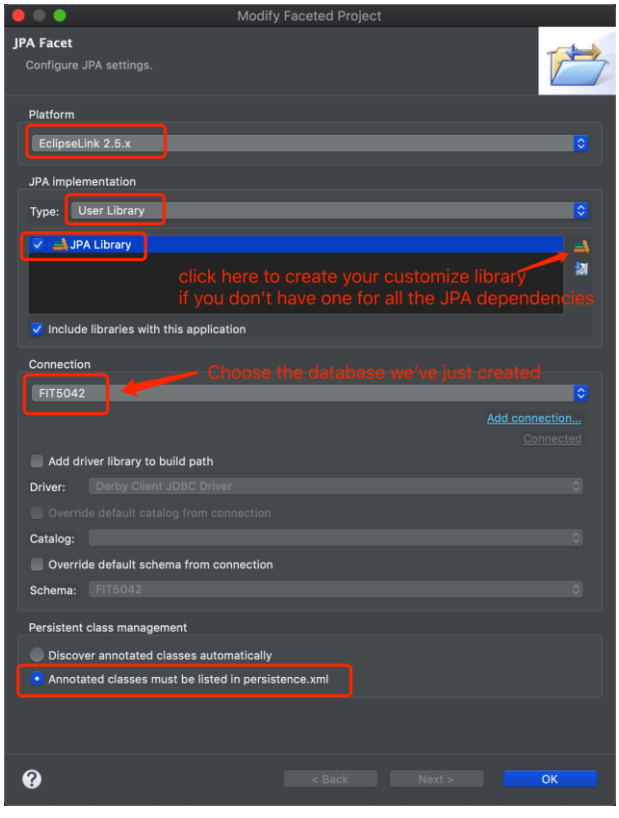
Step 2

Right click your EJB project, open the ***properties*** window.

Search for the ***facet*** Tab, and choose the ***Project Facet***.

Make sure you check the ***JPA*** with ***version 2.1***.

Click "***Further configuration available...***" at the bottom. Note: if no such link exists, the next step explains how to do the further configuration.



Step 3

Choose ***EclipseLink 2.5.x*** as the Platform.

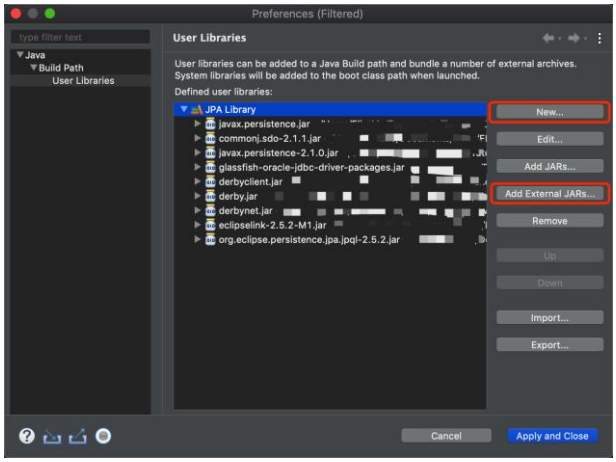
Choose ***User Library*** for the JPA implementation. *(If you don't have JPA Library, click the library button on the right-hand side shown in the snapshot and follow the next step to create the JPA library. You must add the dependencies into the JPA library by using Add External JARs ....)*

Now, in the Data Source Explorer, connect to the database that we've created at the previous steps.

Check the ***Annotated classes must be listed in persistence.xml*** under Persistent class management. The ***persistence.xml*** file should be under ejbModule/META-INF.

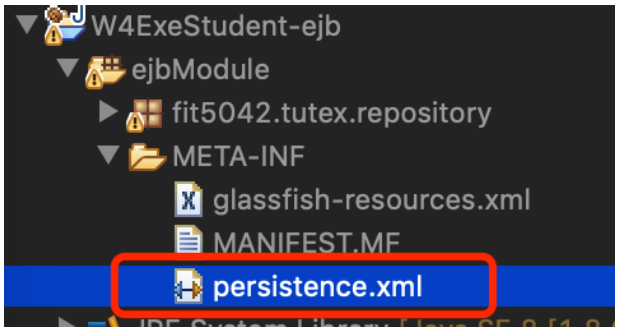
And hit "***OK***".

If there is no "***Further configuration available...***" link, you can do the above configurations in ***Properties/JPA,*** but you will have to include all the dependencies (by using ***Add External JARs ...***) into ***JPA Library*** before you hit the ***Apply*** button for the configuration.



Step 4

Again, if you don't have the dependency library for the JPA, you can create a new one and ***Add External JARs*** from the workplace where you save all the dependence jars.



Step 5

Once you created the persistence unit, you should be able to find this XML file under the META-INF folder.

Open the XML file.

Step 6

To make things simple here, you'll be just showed in a source format of this file. Note: the persistence-unit "W4ExeSolution-ejbPU" should appear as the persistence unit name in the EJB ***JPAPropertyRepositoryImpl.java*** class.

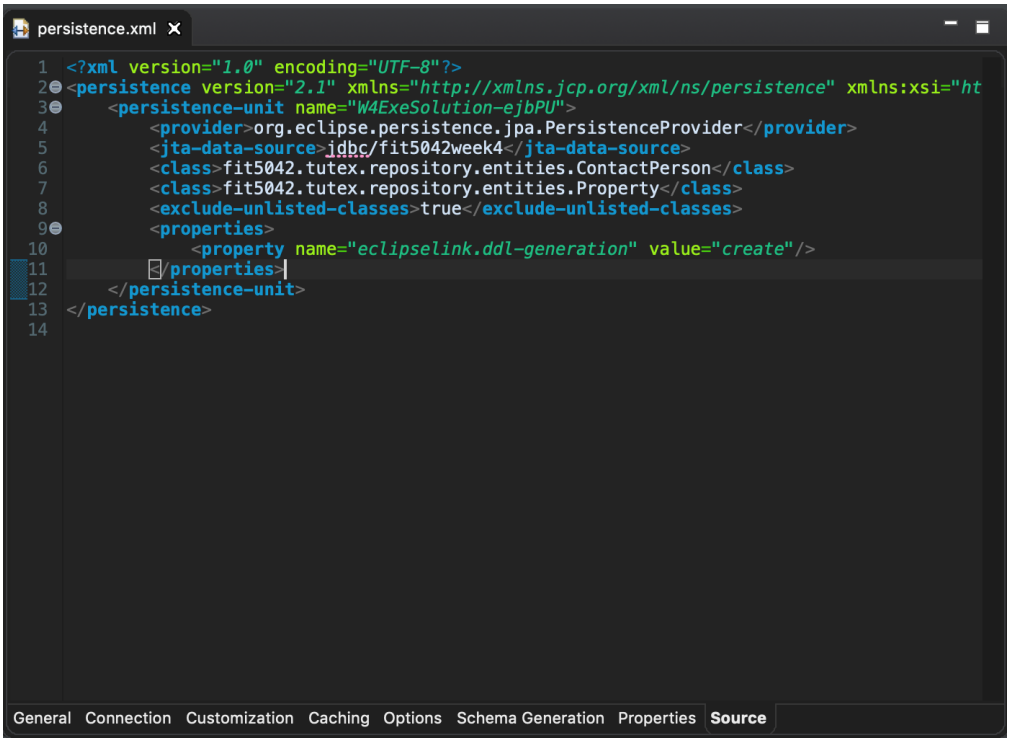


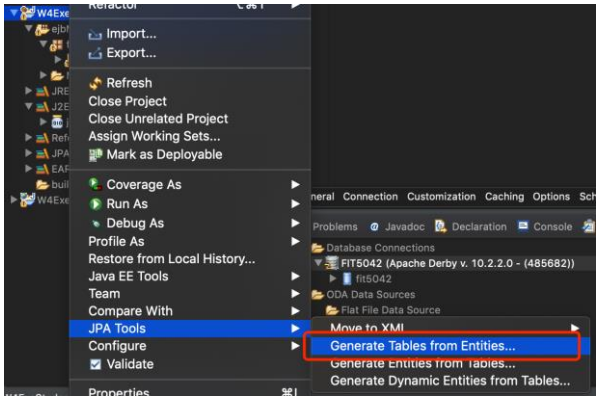
Figure 7 Persistent unit file

Copy and paste the code below:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
        <persistence-unit name="W4ExeSolution-ejbPU">
                <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
                <jta-data-source>jdbc/fit5042week4</jta-data-source>
                <class>fit5042.tutex.repository.entities.ContactPerson</class>
                <class>fit5042.tutex.repository.entities.Property</class>
                <exclude-unlisted-classes>true</exclude-unlisted-classes>
                <properties>
                        <property name="eclipselink.ddl-generation" value="create"/>
                </properties>
        </persistence-unit>
</persistence>
```

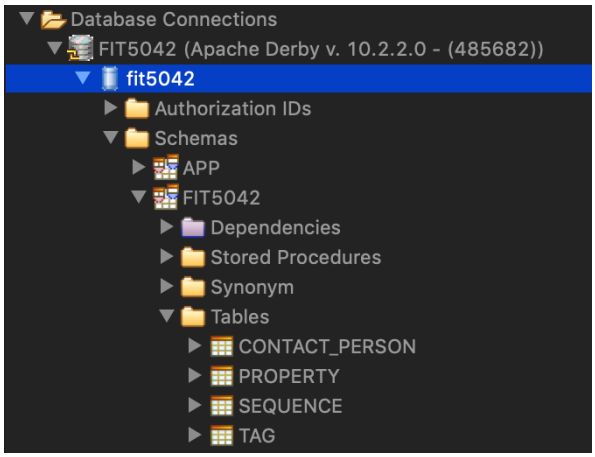
Alternative, you can use the GUI to do the configuration. Please dig deep inside by yourself for other tabs to see how it goes regarding the configuration. Feel free to ask your tutor for any questions.

Deploying the application and creating the entities using JPA



Step 1

Now, since we've successfully created the persistent unit. We can use our entity classes to generate tables in the database, but firstly please make sure the database is connected. Right click the ***EJB*** project, in the ***JPA Tools*** menu, choose ***Generate Tables from Entities...***



Step 2

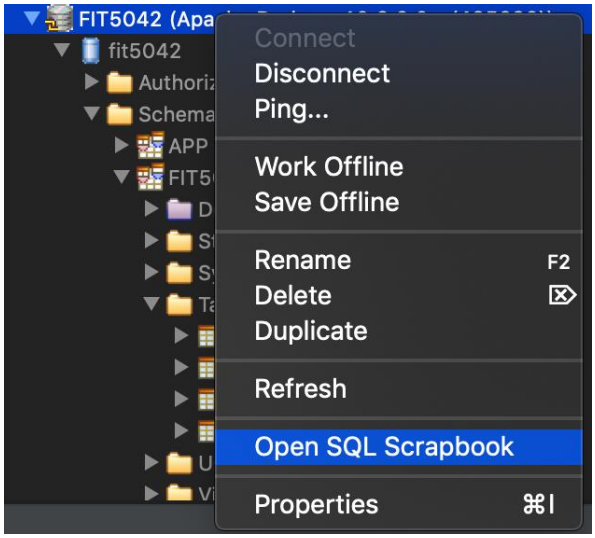
If you followed the above steps correctly, you should now find these tables in your database including:

CONTACT_PERSON

PROPERTY

SEQUENCE

TAG



Step 3

In order to insert data into our tables, right click our database, choose Open SQL Scrapbook

Step 4

Copy and paste the query below to the SQL Scrapbook

```
INSERT INTO FIT5042.CONTACT_PERSON (CONTACT_PERSON_ID, "NAME", PHONE_NUMBER) VALUES (1, 'Eddie Leung', '0411234567');
INSERT INTO FIT5042.CONTACT_PERSON (CONTACT_PERSON_ID, "NAME", PHONE_NUMBER) VALUES (2, 'Sunil Panda', '0422334335');
INSERT INTO FIT5042.CONTACT_PERSON (CONTACT_PERSON_ID, "NAME", PHONE_NUMBER) VALUES (3, 'Jian Liew', '0409233432');
INSERT INTO FIT5042.PROPERTY (PROPERTY_ID, NUMBER_OF_BEDROOMS, PRICE, "SIZE", POSTCODE, "STATE", STREET_ADDRESS, STREET_NUMBER, SUBURB, CONTACTPERSON_CONTACT_PERSON_ID) VALUES (1, 2, 250000, 150, '3145', 'Victoria', 'Boston Avenue', '24', 'Malvern East', 1);
INSERT INTO FIT5042.PROPERTY (PROPERTY_ID, NUMBER_OF_BEDROOMS, PRICE, "SIZE", POSTCODE, "STATE", STREET_ADDRESS, STREET_NUMBER, SUBURB, CONTACTPERSON_CONTACT_PERSON_ID) VALUES (2, 4, 425000, 360, '3168', 'Victoria', 'Bettina Street', '11', 'Clayton', 1);
INSERT INTO FIT5042.PROPERTY (PROPERTY_ID, NUMBER_OF_BEDROOMS, PRICE, "SIZE", POSTCODE, "STATE", STREET_ADDRESS, STREET_NUMBER, SUBURB, CONTACTPERSON_CONTACT_PERSON_ID) VALUES (3, 3, 600000, 150, '3163', 'Victoria', 'Wattle Avenue', '3', 'Glen Huntly', 2);
INSERT INTO FIT5042.PROPERTY (PROPERTY_ID, NUMBER_OF_BEDROOMS, PRICE, "SIZE", POSTCODE, "STATE", STREET_ADDRESS, STREET_NUMBER, SUBURB, CONTACTPERSON_CONTACT_PERSON_ID) VALUES (4, 5, 553000, 150, '3204', 'Victoria', 'Hamilton Street', '5', 'Bentleigh', 3);
```

Step 5

Make sure the database is selected as ***fit5042***, then run the script by right click at the blank space and choose ***Execute All***.

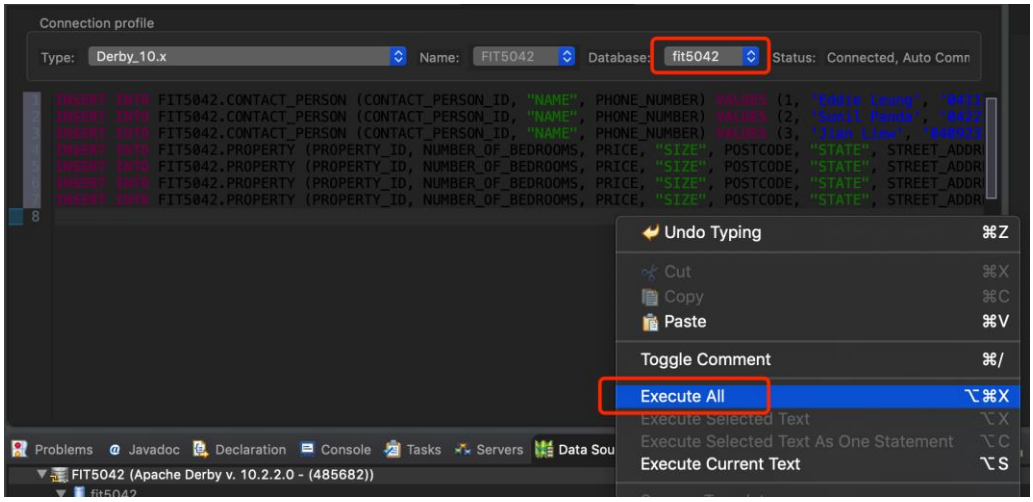
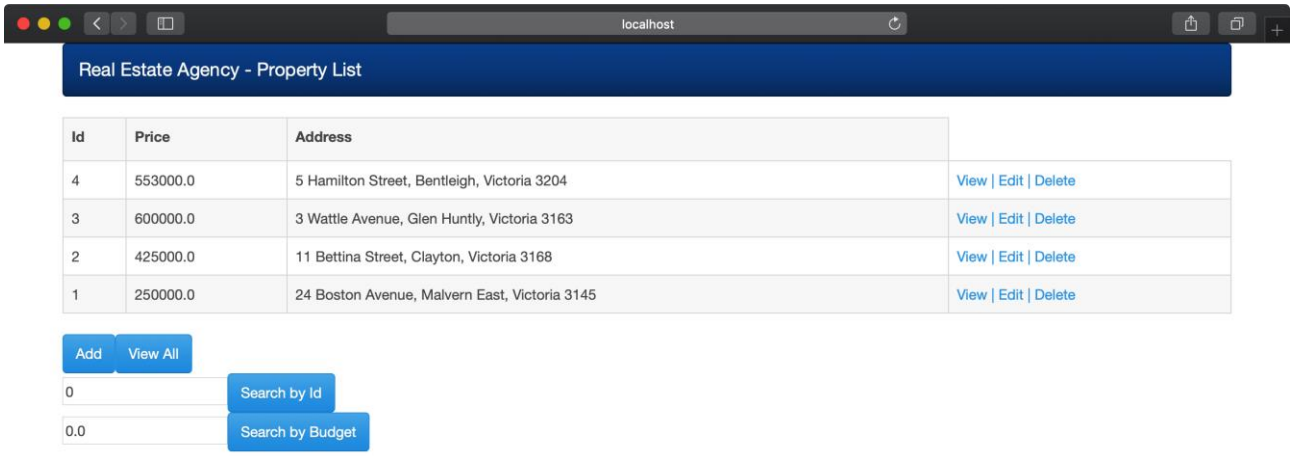


Figure 8 Execute query script

Step 6

Create the Enterprise application and include all the other three projects. Run the Enterprise application on the server.



The above screenshot shows the property list. You can view the property details by clicking on the View link next to the property item in the row, which will show another window with the property details. You can also Add, Edit or Delete a property by clicking the corresponding link, as shown in the above figure.

Search function only covers search by property id and search by budget, as shown above. Please note: it is not a combination search. Please have a play with it and get yourself familiar with the application in accordance with the Java EE architecture.

GUIDENCE

1. In project ***W4ExeStudent-common***, annotates the attribute tags in ***Property*** class so that the tags of a property will be stored in a table called ***TAG***. The tags of a property should be eagerly fetched and the value of each tag must be stored in a column called ***VALUE*** in the ***TAG*** table.
    a. So this needs the attribute tags/getTags method to be an ***@ElementCollection***, also the table name, ***@CollectionTable***, should have a parameter ***name = "TAG"*** and a column name, ***@Column*** should have an attribute ***name = "VALUE"***
2. In project ***W4ExeStudent-common***, enforce the relationship between a property and its contact person using annotation(s). Each property has one and only one contact person. Each contact person might be responsible for zero to many properties. The relationship is bidirectional that means:
    a. the attribute ***contactPerson/getContactPerson*** method in the class ***Property*** should be ***@ManyToOne***. Also the attribute ***properties/getProperties*** method in the class ***ContactPerson*** should be ***@OneToMany*** and it should be mapped back to contactPerson (in Property) so needs an attribute ***mappedBy = "contactPerson"***
3. In project ***W4ExeStudent-ejb***, implement ***searchPropertyByBudget(double budget)***, which is a new method that will return the properties whose price is less than or equal to the budget that is passed as parameter. The method must be completed using ***Criteria API***.
    a. So this needs the ***searchPropertyByBudget(double budget)*** in the ***JPAPropertyRepositoryImpl*** class to be completed:

```
// Create a Criteria Builder
CriteriaBuilder builder = entityManager.getCriteriaBuilder();
// Create a query
CriteriaQuery query = builder.createQuery(Property.class);
//Create a root property
Root<Property> p = query.from(Property.class);
// Create and execute criteria query
query.select(p).where(builder.lessThanOrEqualTo(p.get("price").as(Double.class), budget));
// return the results
```

4. Complete the rest of the methods defined in ***JPAPropertyRepositoryImpl*** class. You must use container managed entity manager to complete those methods. The comments where you need to insert your code are provided in the given code.
    a. So please open the ***JPAPropertyRepositoryImpl*** class file and find out all the methods with some comments asking you to complete that method.

ADDITIONAL TASKS

1) You may have noticed that the Contact Person is not available yet in the Add and Edit pages. If you have finished all the above functions, you should go ahead to add the Contact Person to the Add and Edit windows. The Contact Person should appear as a dropdown list on the screen, showing a list of names of the contact person. Upon you select a contact person from the dropdown list, you should be able to save it into the database in association with this Property.
2) Not all validations are provided for the Add and Edit functions. Please apply some validations to the Add and Edit XHTML files. You should look on the lecture slides (Week 2) to find out the suitable types of validations and apply them in your applications (note: you can assume the data range for the component that you want to apply the validations to).

CONCLUSION

Upon the completion of this lab, you now know

1) The difference between the container managed and application managed entity manager.
2) How to use JPQL and how this differs from SQL.
3) How to use the criteria API to generate Java Persistence Query Language
4) The benefit of using a connection pool.
5) Understand Criteria API
6) Data validations in JSF.