University of Buea

Faculty of Engineering and Technology

Department of Computer Engineering

CEF 502 Java 2 Enterprise Edition Project

EJB Tutorial Assignment

By
Ngong Ivoline-Clarisse Kieleh FE12A131

Lecturer: Mr Brinard Elingese

Tuesday,7 th June 2016.

Definition of Folder Structure and Files

The folder is divided into 3 subfolders and a file. Most chapters of the tutorial have exercises and it is presented as follows:

- The EjbModule folder consist of different packages which are named according to what the chapter requires. We have the following packages in the folder.
  - com.ivoline.stateless
  - com.ivoline.stateful
  - com.ivoline.persistence
  - com.ivoline.entities
  - com.ivoline.callback
  - com.ivoline.interceptor
  - com.ivoline.message
  - com.ivoline.model
  - com.ivoline.query
  - com.ivoline.rawdatabase
  - com.ivoline.timer
  - com.ivoline.webservice

  These packages contains the necessary EJB's, datasource and persistent units.

- The EjbModuleTester contains the tester files for all the packages in the EJBcomponent. The EjbTester provides the interface.

- EjbWebService provides an option to expose session EJB as a web-service.

Requirements to run the program

The requirements on for the running of the program goes as follows.

Tool	**Description**
IDE	NetBeans 8.1
Server	Jboss 5.1.0
Database	Postgres
OS	Windows, Unix, Linux
Extra Jars	Jpa-api-2.0-cr-1.jar, hibernate-core.jar

Steps to Run Program

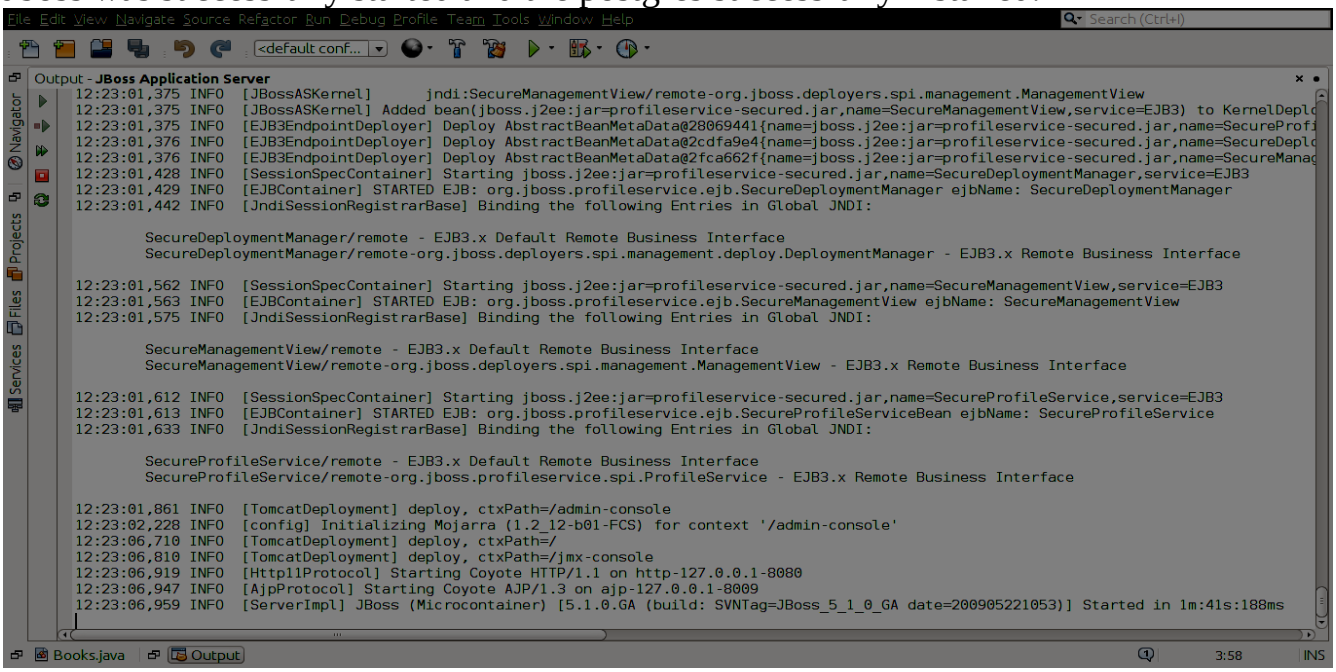
To run the program, import the different folders into the IDE specified above. Start your jboss server under servers tab and select jboss. The EjbComponent folder has to be deployed on Jboss before it runs, so right click on the EjbComponent and click on deploy . This deploys the module on the server. Under the jboss server tab, take note of the line containing jndi, it has the necessary details you need for the lookup in your client application.

On the EjbTester application, right click and choose properties select library and choose add project select the ejbmodule you just deployed on your jboss server select addjar/folder and choose all the libraries in the client folder of your jboss server save and run. In addition, add the extra jars, shown above to your project. You're done.

ScreenShots of Outputs and Challenges Faced

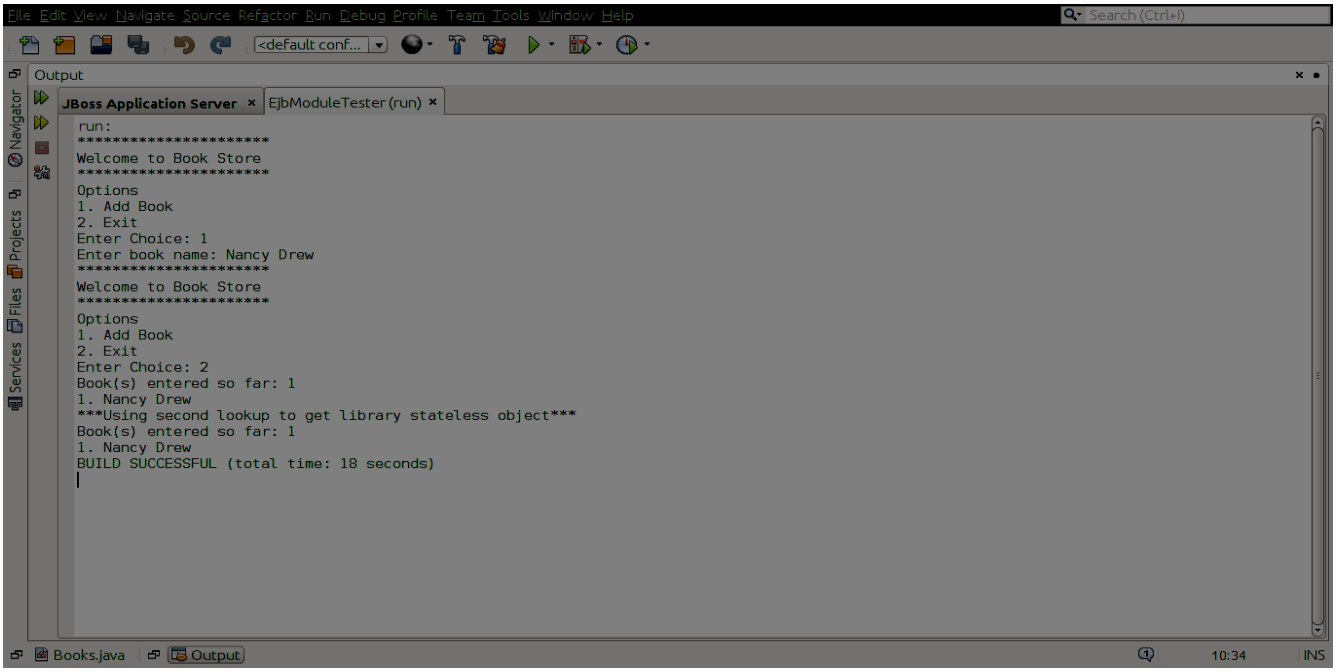
Chapter 2

Problems began during environment set-up, netbeans 7 wasnt detecting jboss, so I had to install Netbeans 8.1 which finally detected jboss5 and started the server successfully. Jboss was successfully started and the postgres successfully installed.



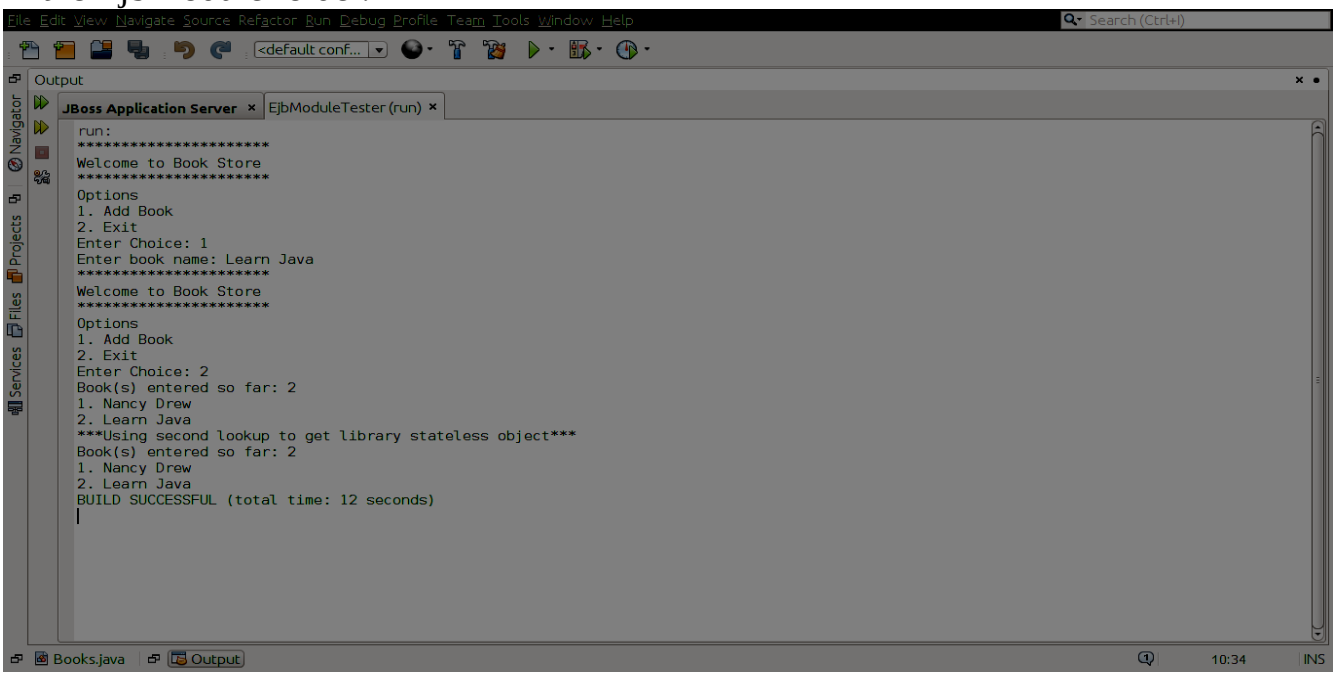
Chapter 3: EJB – CREATE APPLICATION

The aim of this project is to show the reader how to create a simple EJB application.In EjbModuleTester folder, run the EjbTester class. When this class is run, the output in figure 2 is obtained.



Chapter 4: EJB – STATELESS BEAN

The aim of this project is to show the reader how to create a stateless bean.A stateless session bean is a type of enterprise bean, which is normally used to perform independent operations.The **@Stateless** annotation to signify it a stateless bean. In EjbModuleTester folder, run the EjbTester.java class(Output in figure 3). This class makes use of the LibrarySessionBean/remote, which was automatically created by Jboss, after the EjbModule is deployed. It also makes use of the package **com.ivoline.stateless** found in the EjbModule folder.

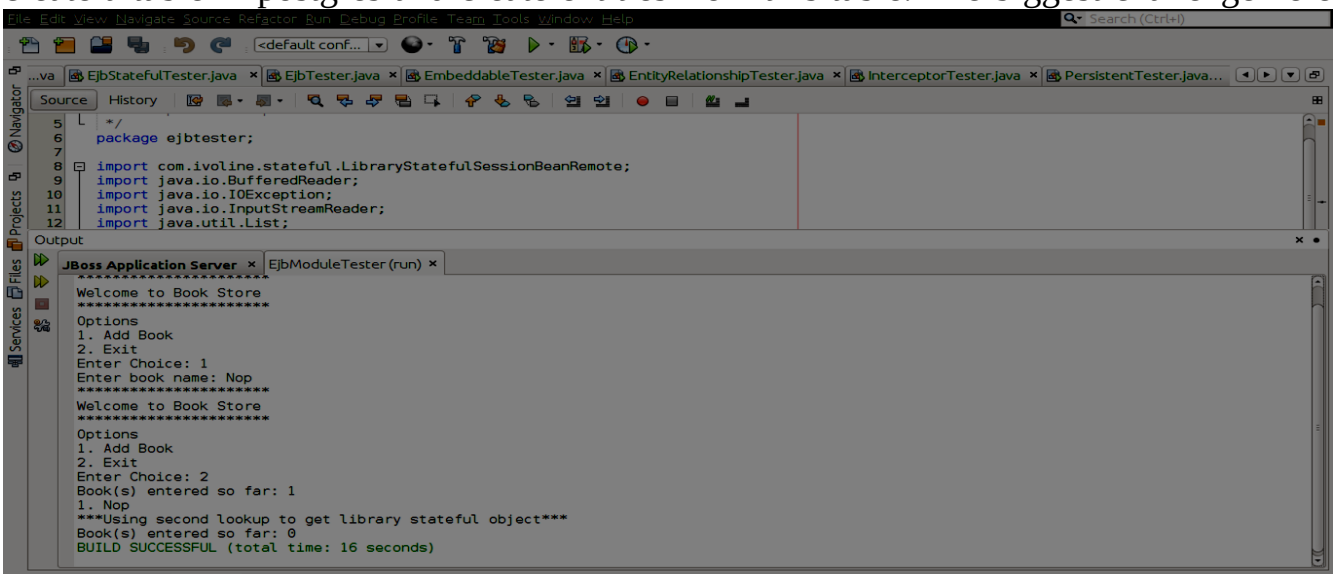


Chapter 5: EJB – STATEFUL BEAN

The aim of this project is to show the reader how to create a statelful bean.AA stateful session bean is a type of enterprise bean, which preserve the conversational state with client. The **@Stateful** annotation to signify it a stateful bean. In EjbModuleTester folder, run the EjbStatefulTester class(Output in figure 4). This class makes use of the LibraryStatefulSessionBean/remote, which was automatically created by Jboss, after the EjbModule is deployed. It also makes use of the package **com.ivoline.stateful** found in the EjbModule folder**.**

Chapter 6: EJB – PERSISTENCE

Here we got to use a persistence unit, datasource, entity and entity manager. We had to create a table in postgres and create entities from this table. The biggest challenge here



was connecting the table to Netbeans which was not described in the tutorial. Also, we had to make use of the jpa-api2.0.1 jar library which was not also specified in the tutorial. It should be noted that this was the most troublesome chapter as there were a lot of errors in the code, queries had to be changed and a lot of modifications had to made for the code to work. For example: the query

entityManager.createQuery("From Book").getResultList();

had to be changed to

entityManager.createQuery("SELECT b FROM Book b");

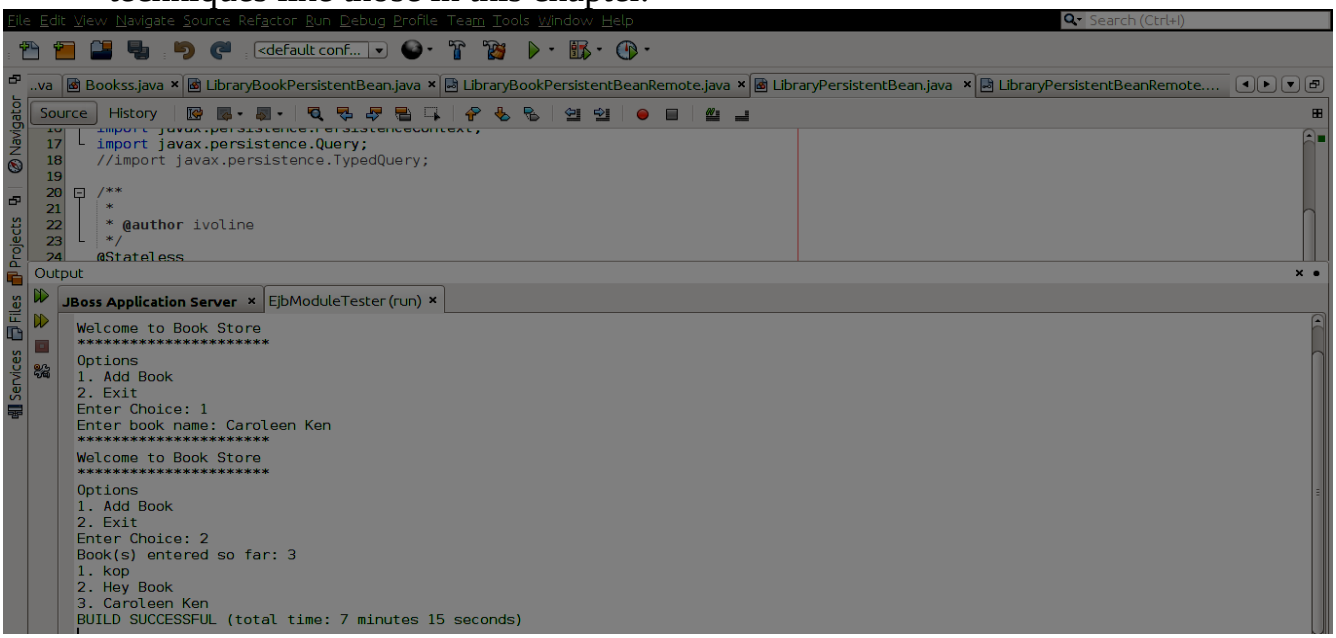
According to the tutorial, the persistence unit and datasource had to be created manually. When this is done manually,I got a lot of errors. After doing a lot of research and debugging, I discovered that there was a way for netbeans to do this automatically, which was going to prevent all the errors.

With all the errors rectified, the class PersistentTester was run, and the output in figure 5 was gotten. It makes use of makes use of the LibraryPersistentBean/remote , which was automatically created by Jboss, after the EjbModule is deployed. It also makes use of the

package com.ivoline.persistence and com.ivoline.entities found in the EjbModule folder.

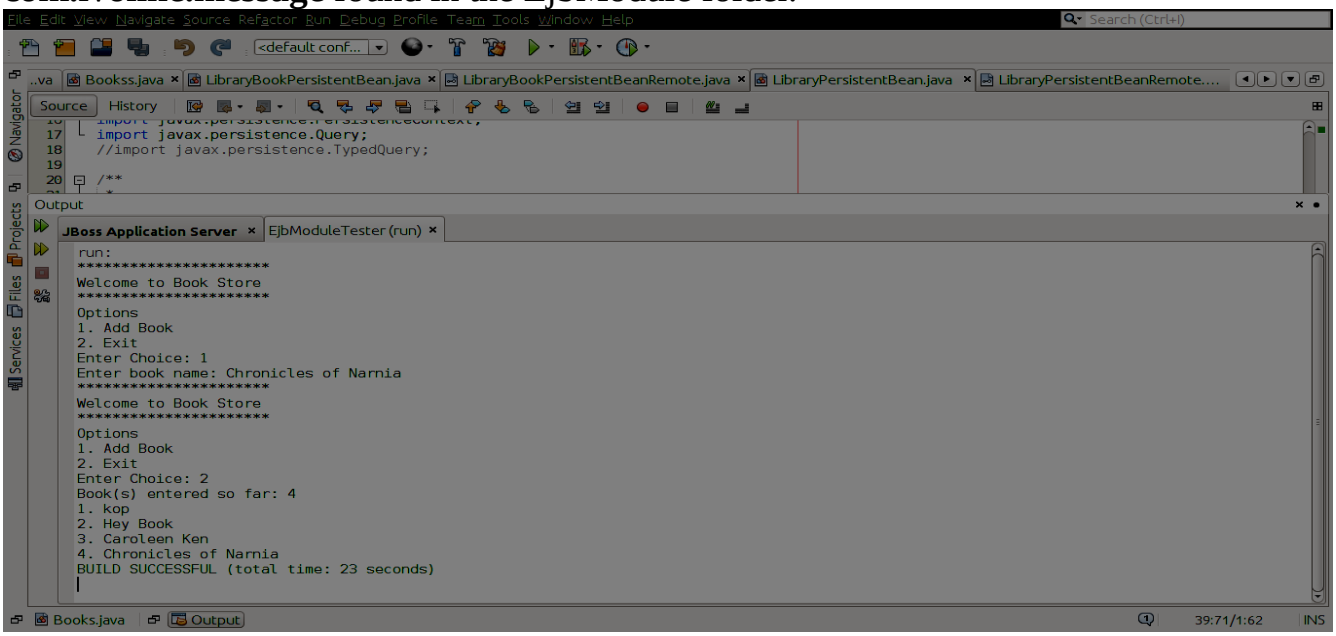
Note

1. **com.ivoline.entities** contains all the entities for the tables in the database
2. The remaining chapters did not have a lot of errors as they made used the same techniques like those in this chapter.



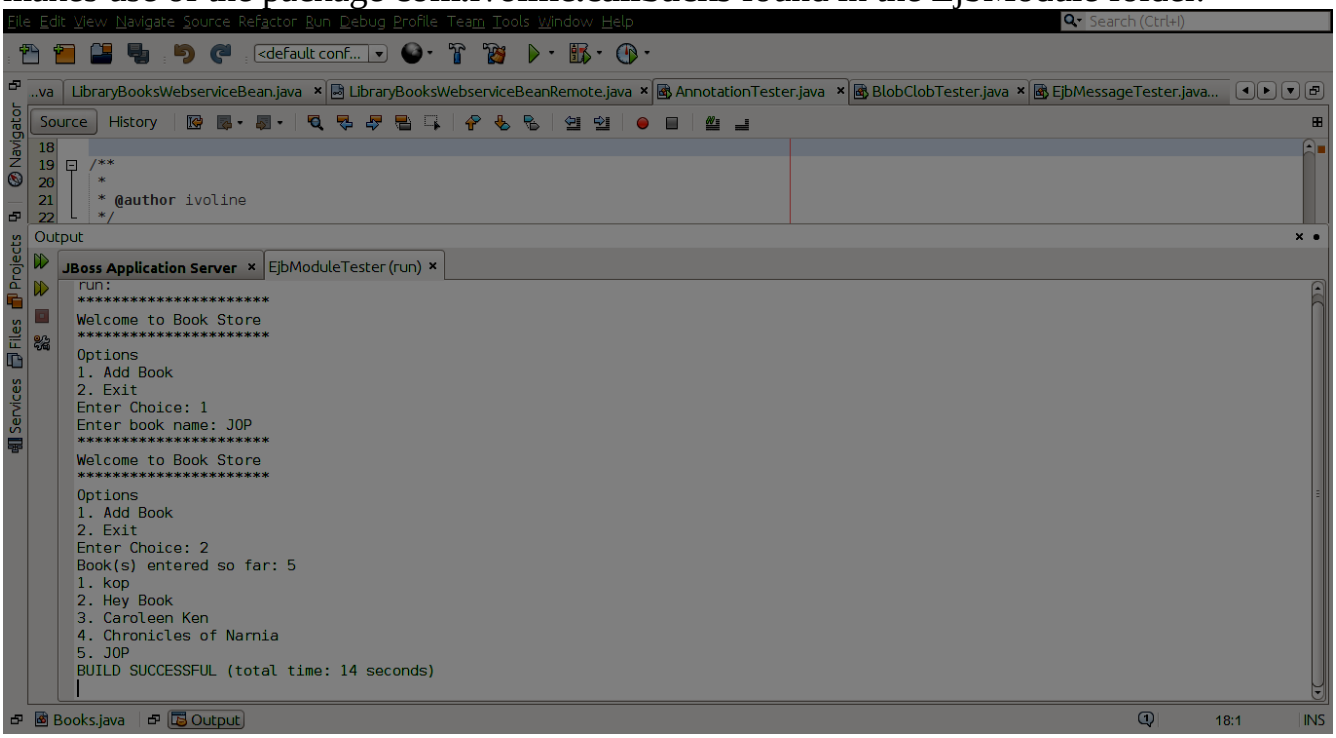
Chapter 7: EJB – MESSAGE DRIVEN BEANS

The aim of this project is to show the reader how to create a message driven bean.A message driven bean is a type of enterprise bean, which is invoked by EJB container when it receives a message from queue or topic. In EjbModuleTester folder, run the EjbMessageTester.java class(Output in figure 6).It makes use of the package **com.ivoline.message** found in the EjbModule folder.



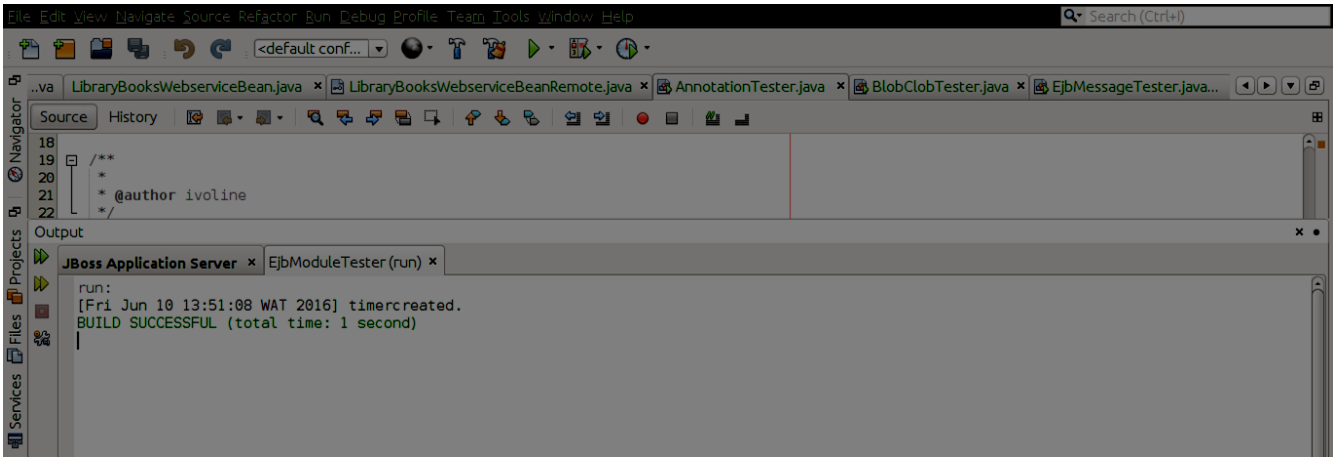
Chapter 9: EJB – CALLBACKS

Callback is a mechanism by which the life cycle of an enterprise bean can be intercepted. In EjbModuleTester folder, run the AnnotationTester.java class(Output in figure 7).It makes use of the package **com.ivoline.callbacks** found in the EjbModule folder.



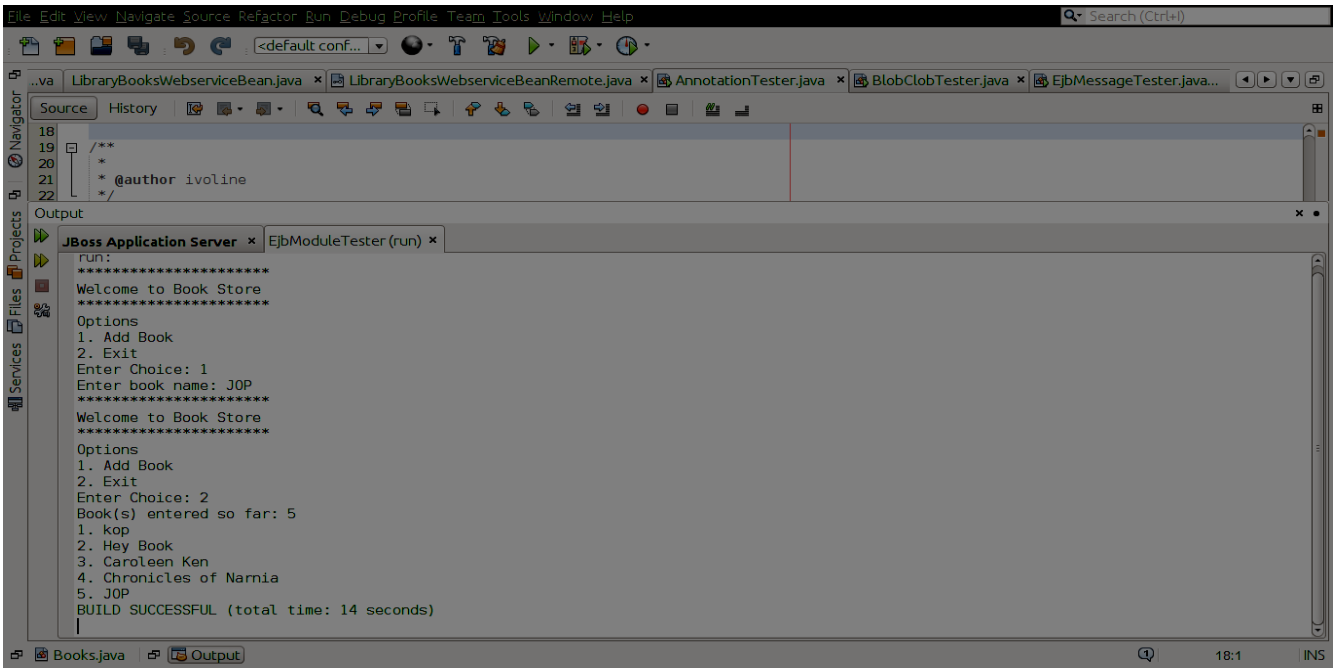
Chapter 10: EJB – TIMER SERVICE

Timer Service is a mechanism by which scheduled application can be build. In EjbModuleTester folder, run the TimerTester.java class(Output in figure 8).It makes use of the package **com.ivoline.timer** found in the EjbModule folder.



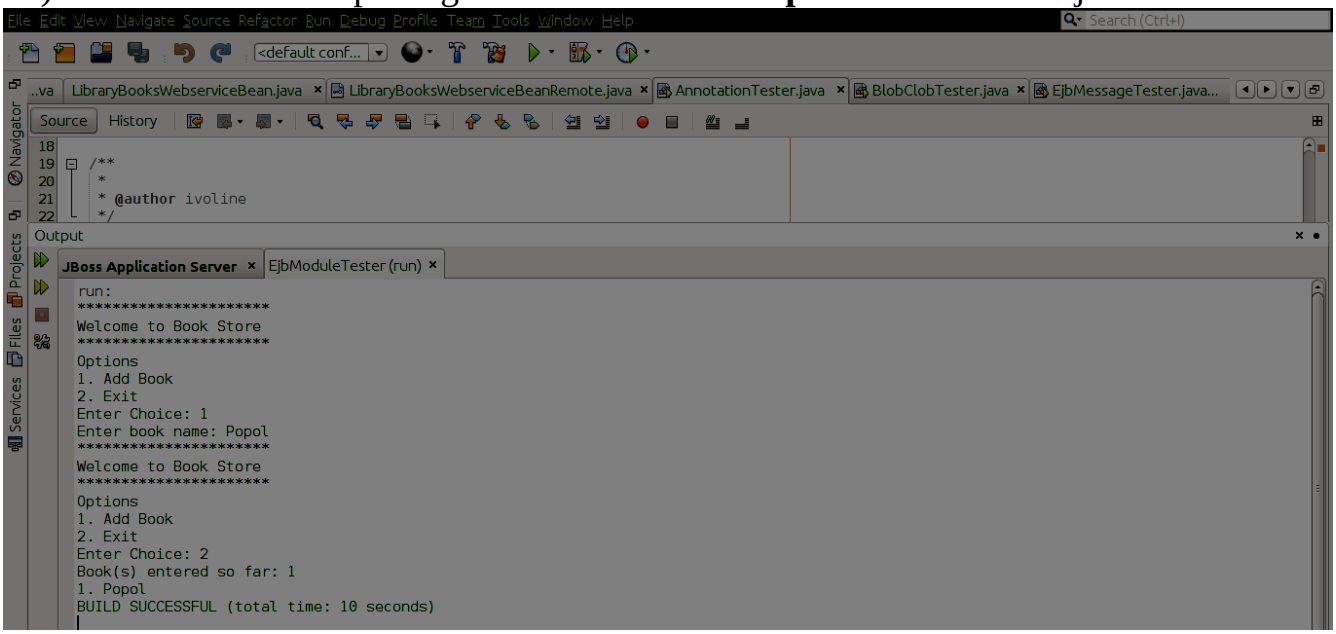
Chapter 11: EJB – DEPENDENCY INJECTION

EJB 3.0 specification provides annotations, which can be applied on fields or setter methods to inject dependencies. EJB Container uses the global JNDI registry to locate the dependency. In EjbModuleTester folder, run the AnnotationTester.java class(Output in figure 9).It makes use of the package **com.ivoline.callback** found in the EjbModule folder.



Chapter 12: EJB – INTERCEPTORS

An interceptor method is called by EjbModule before business method call it is intercepting. In EjbModuleTester folder, run the InterceptorTester class(Output in figure 10).It makes use of the package **com.ivoline.interceptor** found in the EjbModule folder.



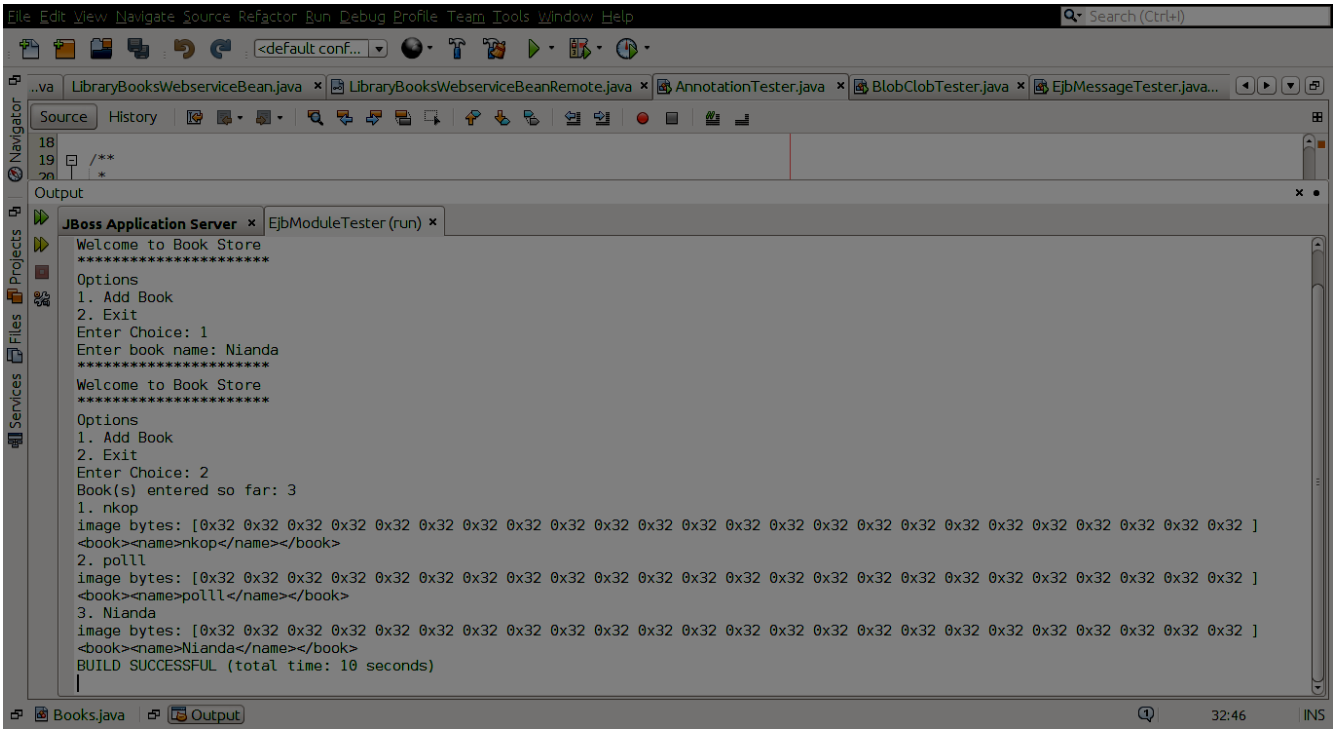
Chapter 13: EJB – EMBEDDABLE OBJECTS

EJB 3.0 provides option to embed JAVA POJO (Plain Old Java Object) into an entity bean and allows to map column names with the methods of the embedded POJO class. A java POJO to be embedded must be annotated as @Embeddable..

Here, I had to create new columns in the Books table, create a Pubisher class in entities etc. In EjbModuleTester folder, run the EmbeddableTester class(Output in figure 11).It makes use of the package **com.ivoline.stateless and com.ivoline.entities** found in the EjbModule folder.

Chapter 14: EJB – BLOBS/CLOBS

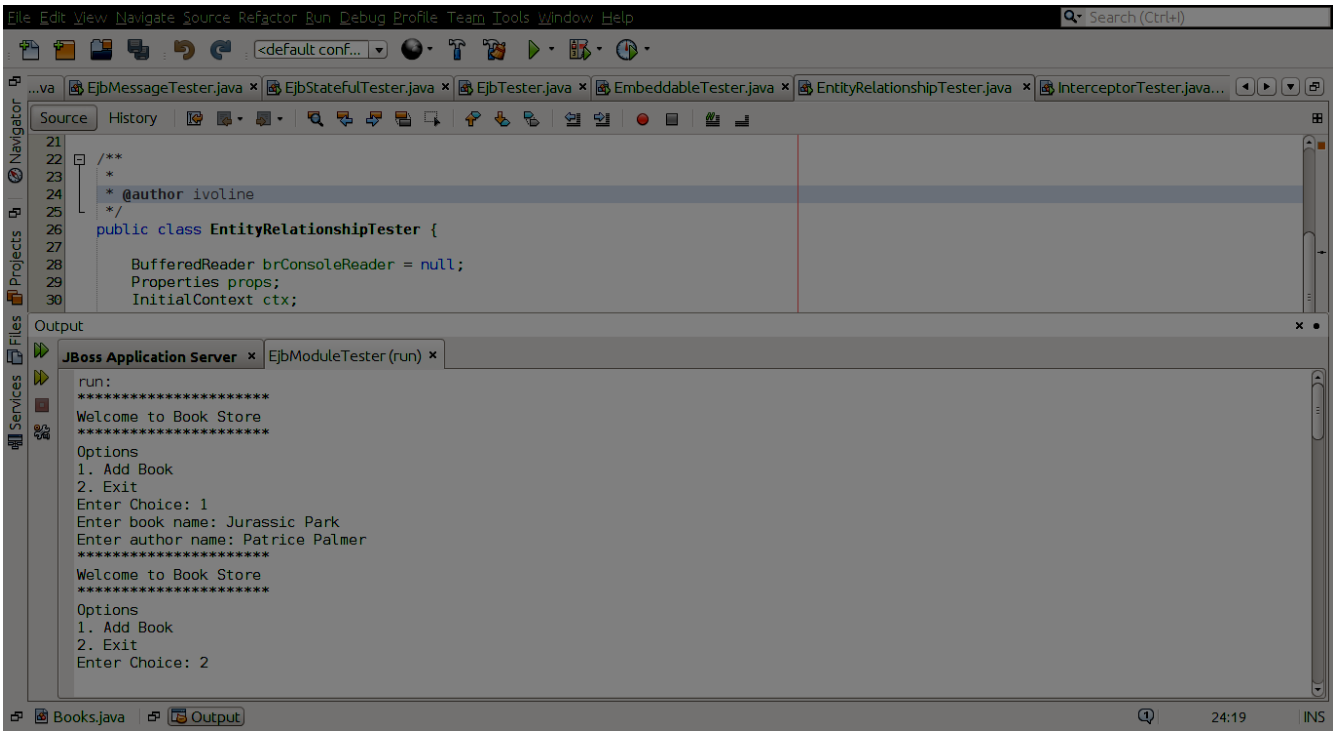
EJB 3.0 provides option to make use of @Lob annotation. Which we learn in this chapter. Here, I had to create new columns in the Books table. In EjbModuleTester folder, run the BlobClobTester class(Output in figure 12).It makes use of the package **com.ivoline.stateless and com.ivoline.entities** found in the EjbModule folder.



Chapter 18: EJB – ENTITY RELATIONSHIPS

EJB 3.0 provides option to define database entity relationships/mappings like one-to-one, one-to-many, many-to-one, and many-to-many relationships.

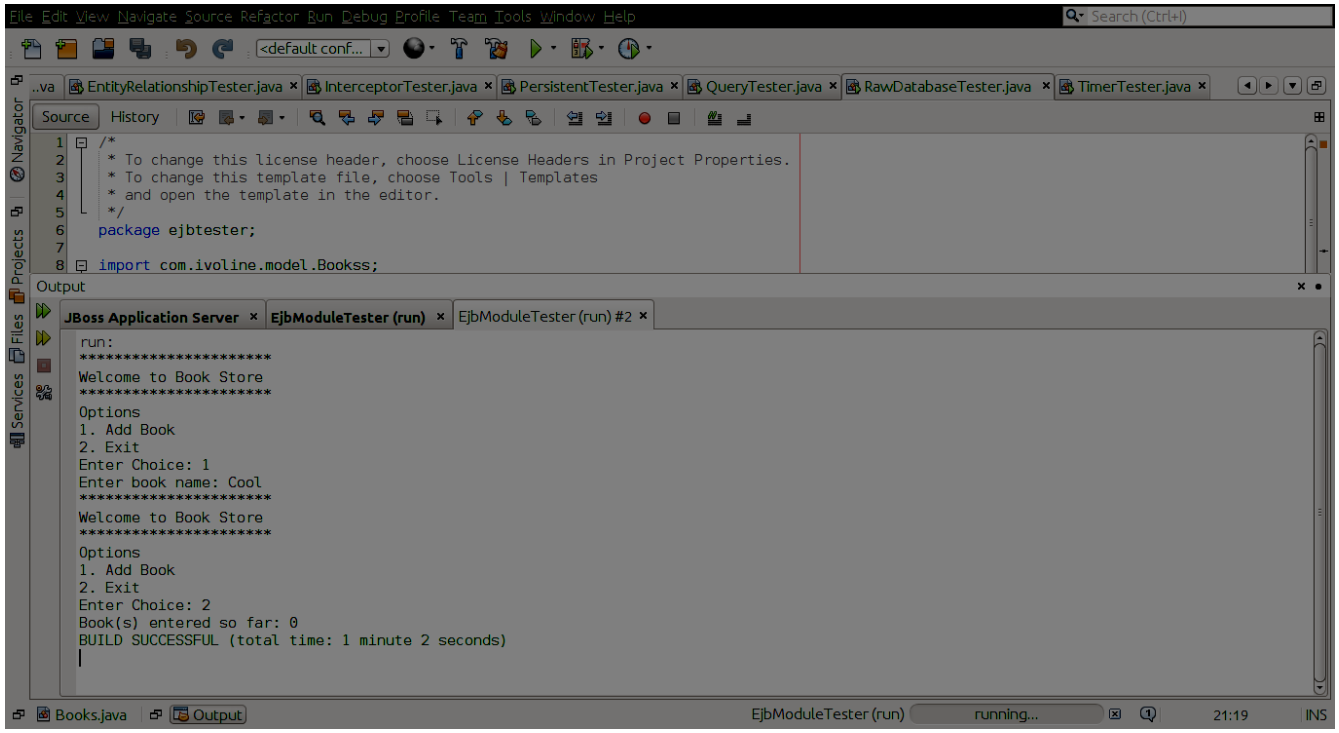
Here, I had to create new tables author, book_author. In EjbModuleTester folder, run the EntityRelationshipTester class(Output in figure 13). It makes use of the package **com.ivoline.stateless** and **com.ivoline.entities** found in the EjbModule folder.



Chapter 19: EJB – ACCESS DATABASE

In EJB 3.0, persistence mechanism is used to access the database in which the container manages the database related operations

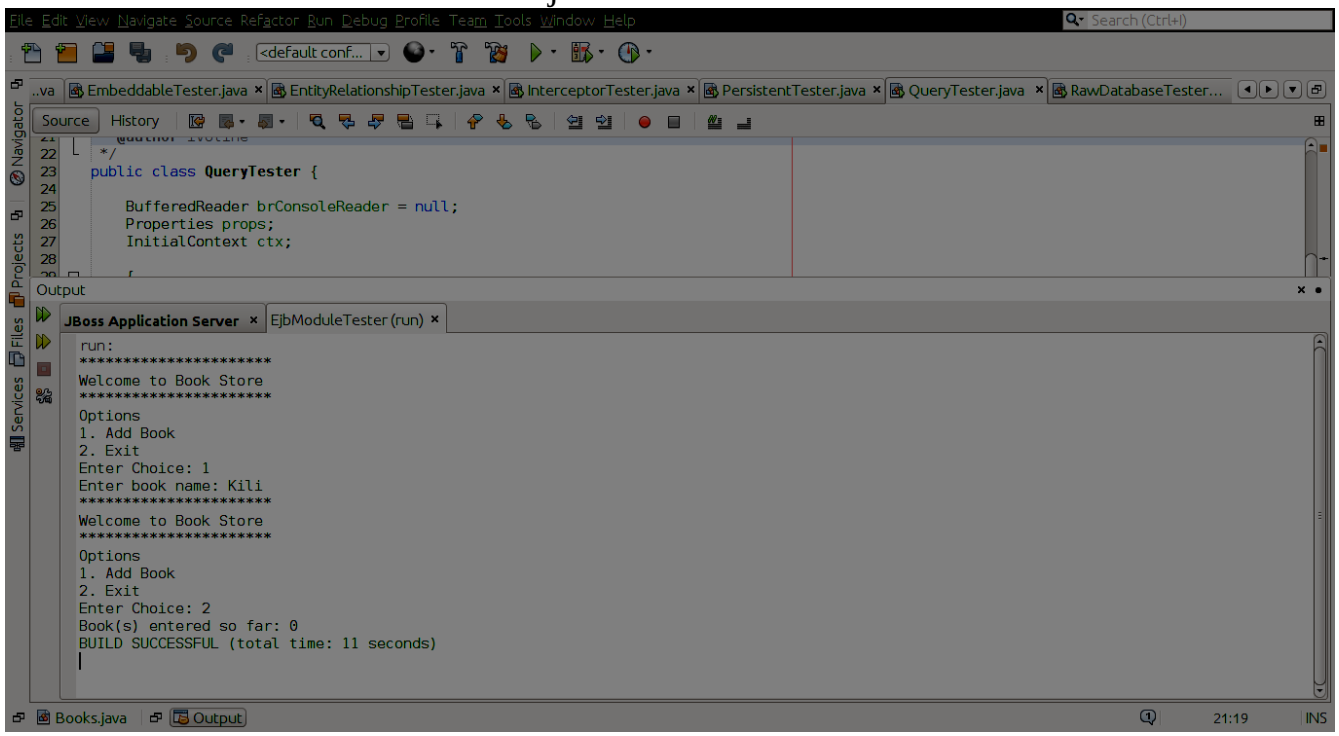
Here, I had to create new tables author, book_author. In EjbModuleTester folder, run the RawDatabaseTester class(Output in figure 13). It makes use of the package **com.ivoline.rawdatabase** and **com.ivoline.entities** found in the EjbModule folder.



Chapter 20: EJB – QUERY LANGUAGE

EJB Query Language is quite handy to write custom queries without worrying about underlying database details

Here, I had to create a Model class .In EjbModuleTester folder, run the QueryTester class(Output in figure 13). It makes use of the package **com.ivoline.model** and **com.ivoline.stateless** found in the EjbModule folder.



Chapter 22: EJB – WEB SERVICES

EJB 3.0 provides an option to expose session EJB as a webservice. @WebService annotation is used to mark a class as a web service end point and @WebMethod is used to expose a method as web method to client.
In WebServiceClientEjb folder, run the WebServiceTester class(Output in figure 13). It makes use of the package **com.ivoline.persistence** and **com.ivoline.stateless** found in the EjbModule folder.

This did not run as there were a lot of errors

Contact Information for Issues Reporting

In case of any issues with the code, report to this address

Email: ivolinengong@gmail.com
Phone: 675568060