Internet Engineering

Tomasz Babczyński, Zofia Kruczkiewicz
Tomasz Kubik

Information systems modelling– UML and service description languages

Laboratory 4

Choose yourself and new technologies

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!

Wrocław University of Technology

EUROPEAN SOCIAL FUND

Project co-financed from the EU European Social Fund

Design patterns used to build the Integration nad ResourcesTiers

D.Alur, J.Crupi, D. Malks, Core J2EE. Desin Patterns

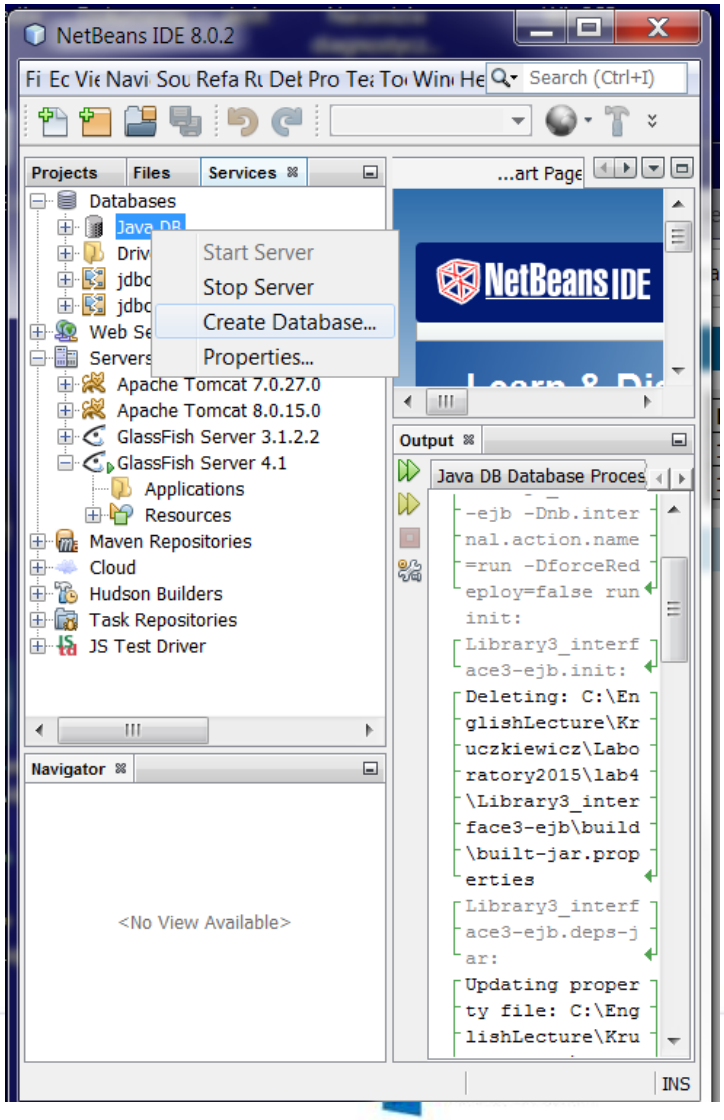
Outline of creating the Library Catalogue Java Application

1. Create a database in the Derby database system
2. Create Annotations in the object model of the Business Tier
3. You may add annotation to your own new classes and create the proper controllers – for higher assessment (5.0 or 5.5)
4. Create Session Beans for Entity classes

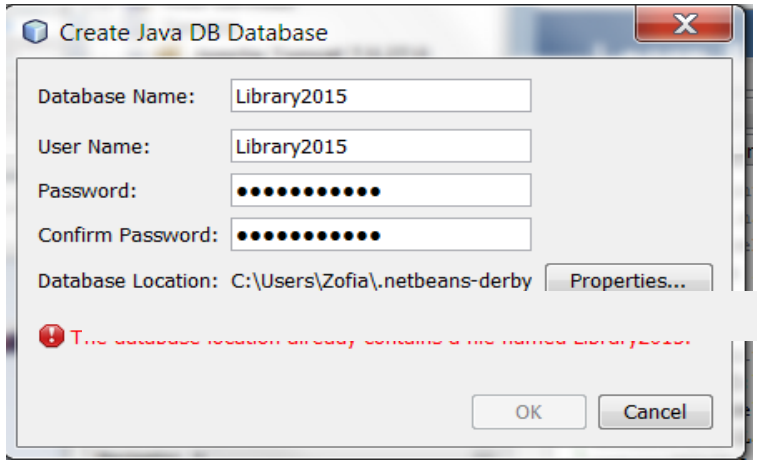
1. Create a database in the Derby database system

1. Select the Services tab and then expand the Databases node. Then right-click on Java DB item and select Create Database – (slide 2)
2. In the form for creating a database (slide 3), enter the data as follows: Database Name, User, Password and the Database Location (default or selected by the user). In the selected directory will be created a directory named database with an empty database (slide 4)
3. Check a directory where you have created from NetBeans an empty database in the system database Derby - - you can see the new katalogue Library1 (slide 4)
4. To connect to the database: Right-click the database connection node and choose the Connect item (slide 6) - **jdbc:derby:///localhost:1527/Library2015:[Library2015 on LIBRARY2015] (jdbc:derby:///localhost:1527/DataBase_Name:[User on DataBaseSchema])** .
5. Now expand the database connection node to browse the database until disconnect from the database. Then expand the database node to see the empty Library Database (slide 7) p – the Library2015 node is lacking.

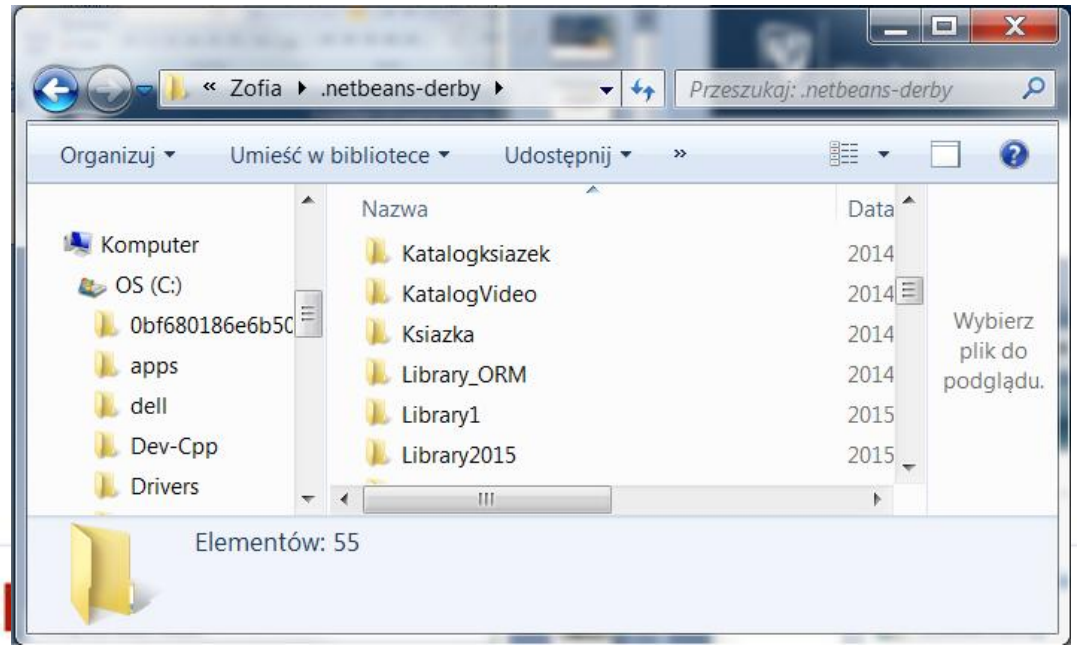
1)



2)

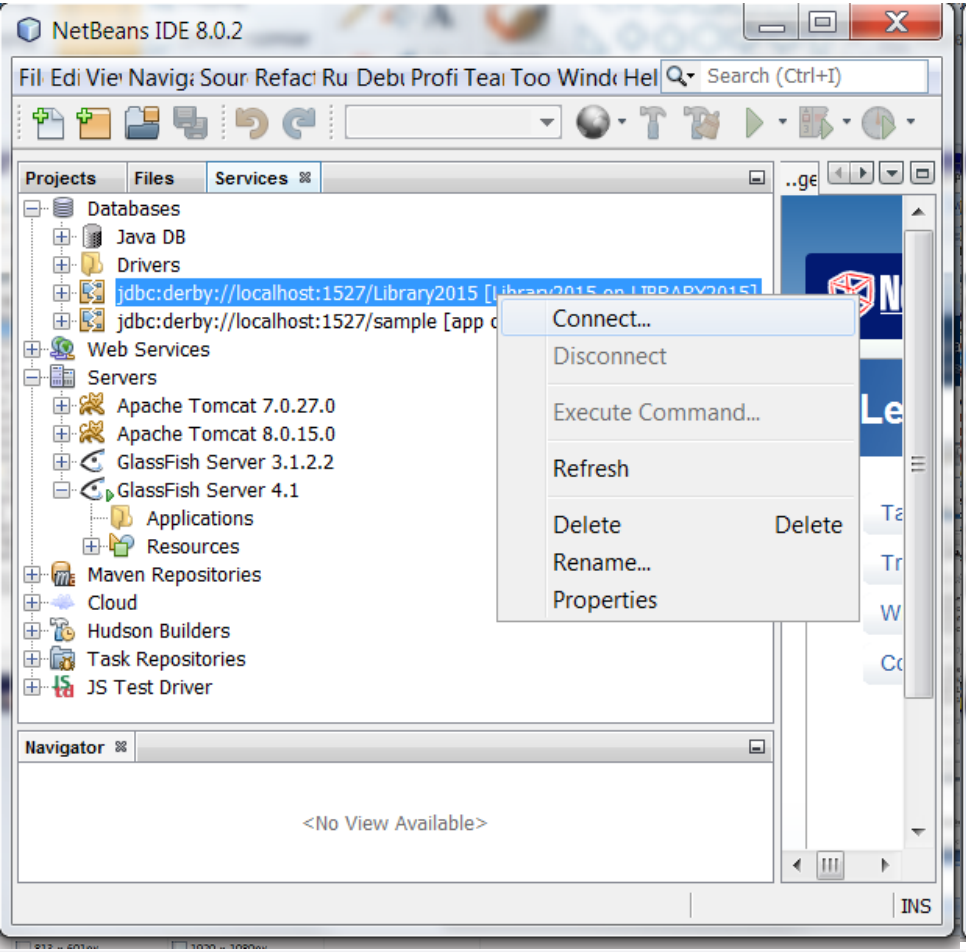


3)

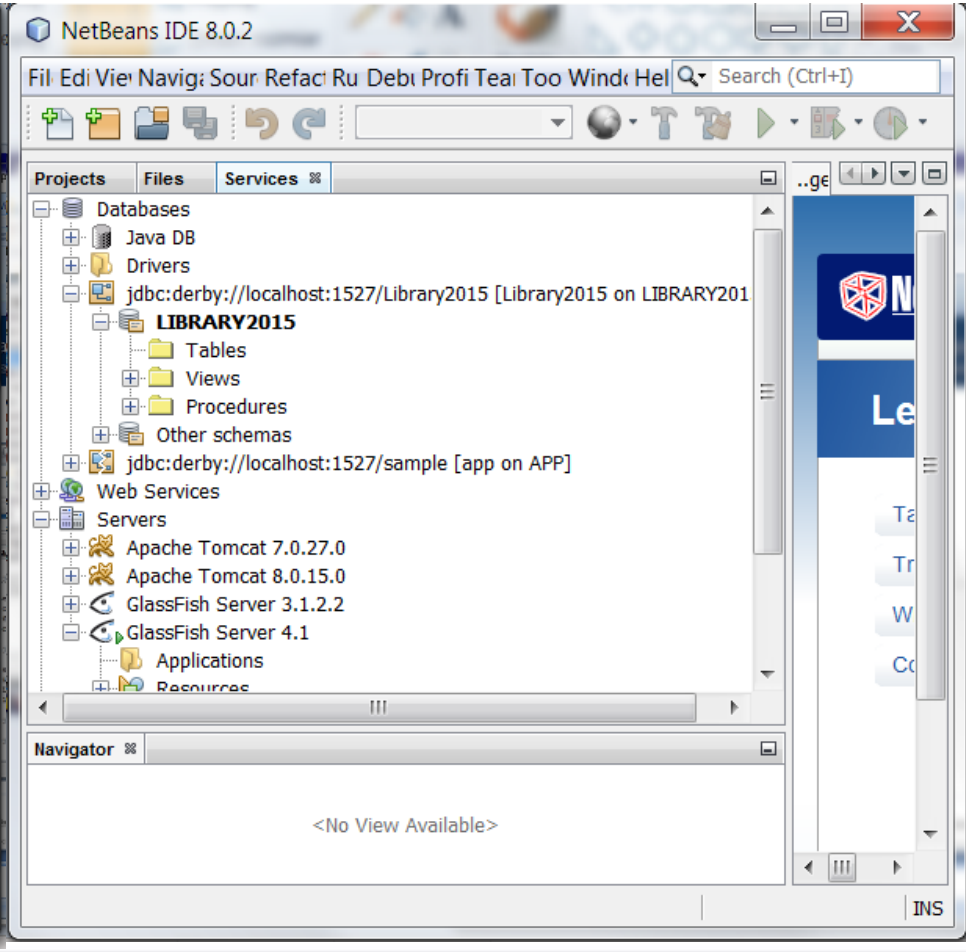


Project co-financed from the EU European Social Fund

4)


NetBeans IDE 8.0.2
Search (Ctrl+I)
Projects
Files
Services
Databases
Java DB
Drivers
jdbc:derby://localhost:1527/Library2015 [
jdbc:derby://localhost:1527/sample [app
Web Services
Servers
Apache Tomcat 7.0.27.0
Apache Tomcat 8.0.15.0
GlassFish Server 3.1.2.2
GlassFish Server 4.1
Applications
Resources
Maven Repositories
Cloud
Hudson Builders
Task Repositories
JS Test Driver
Connect...
Disconnect
Execute Command...
Refresh
Delete
Delete
Rename...
Properties
Navigator
<No View Available>
INS

5)


NetBeans IDE 8.0.2
Search (Ctrl+I)
Projects
Files
Services
Databases
Java DB
Drivers
jdbc:derby://localhost:1527/Library2015 [Library2015 on LIBRARY201
LIBRARY2015
Tables
Views
Procedures
Other schemas
jdbc:derby://localhost:1527/sample [app on APP]
Web Services
Servers
Apache Tomcat 7.0.27.0
Apache Tomcat 8.0.15.0
GlassFish Server 3.1.2.2
GlassFish Server 4.1
Applications
Navigator
<No View Available>
INS

3. Create Annotations in the object model

Mapping Entity Classes (Help of the NetBeans)

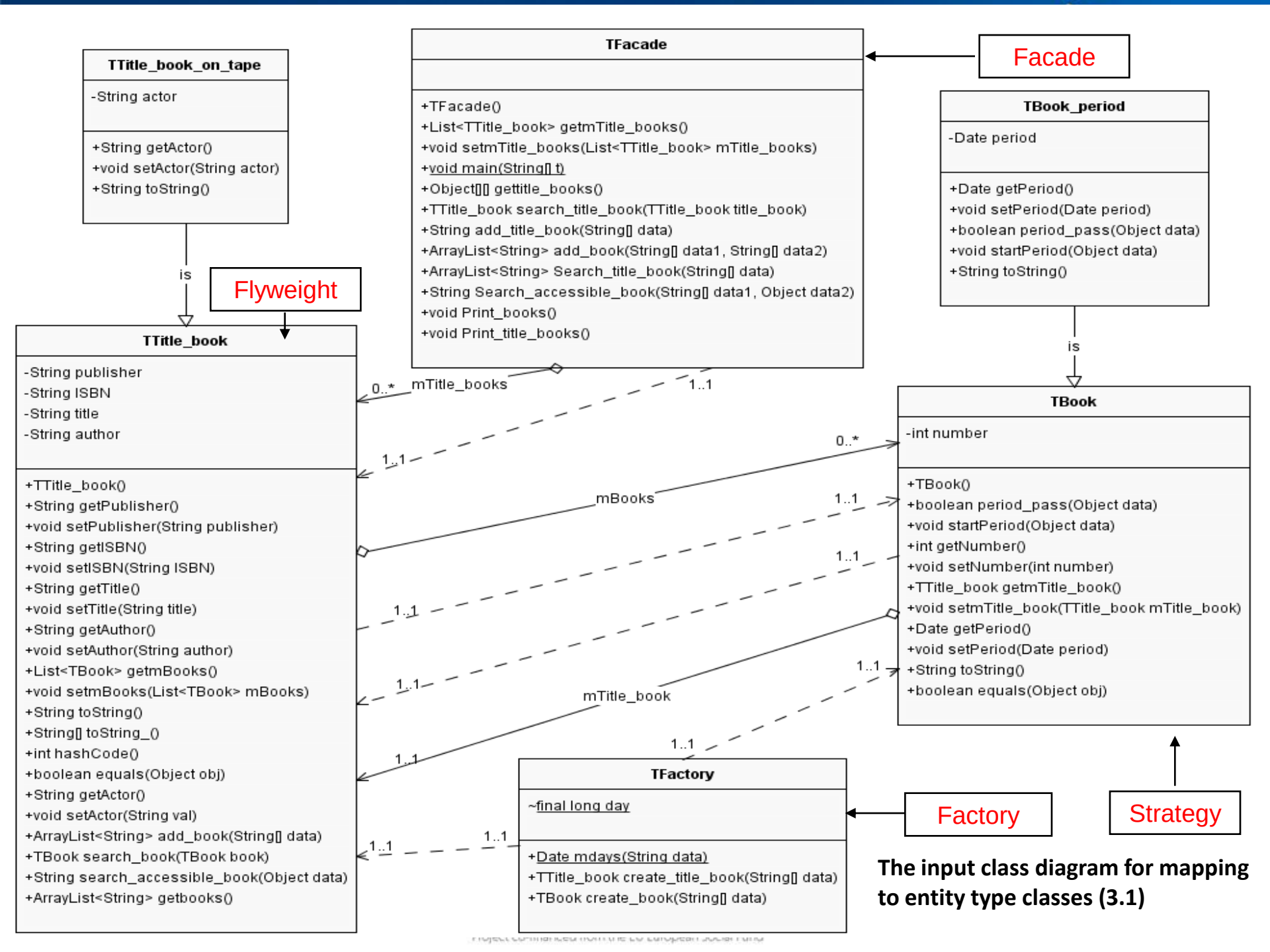
- An entity class is used to represent a table in a database, and the fields in an entity class correspond to columns in that table. In an entity class, you can use annotations to specify how fields in an entity class are mapped to the corresponding database columns and tables.
- If the name of the mapped database column is the same as the field or property name because they are mapped by default.

The following annotations are commonly used when mapping entity classes:

- @Entity defines the Entity class
- @Id Specifies the primary key property or field of an entity.
- @GeneratedValue Allows you to specify the strategy that automatically generates the values of primary keys. Used with @Id.
- @Column Specifies a mapped column for a persistent property or field.
- @ManyToMany Defines a many-valued association with many-to-many multiplicity.
- @ManyToOne Defines a single-valued association to another entity class that has many-to-one multiplicity.
- @OneToMany Defines a many-valued association with one-to-many multiplicity. For more on using annotations and annotation elements to map entities in an enterprise application, see the Java EE 5 Tutorial

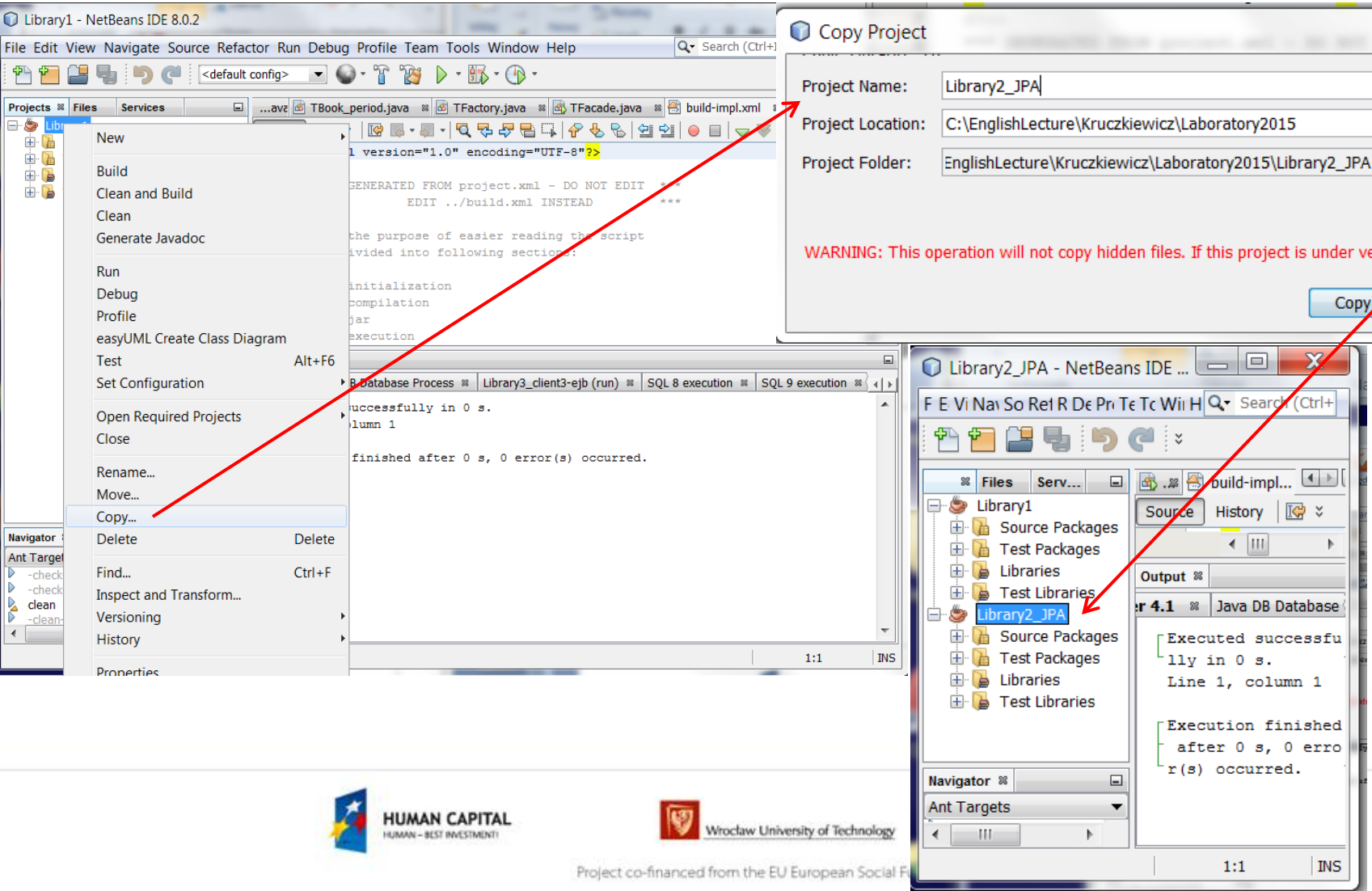
Insert the annotations shown in the next slides to the TTitle_book, TTitle_book_on_tape, TBook, TBook_period

Click the Add Class and select all classes as the Entity classes (slide 5)

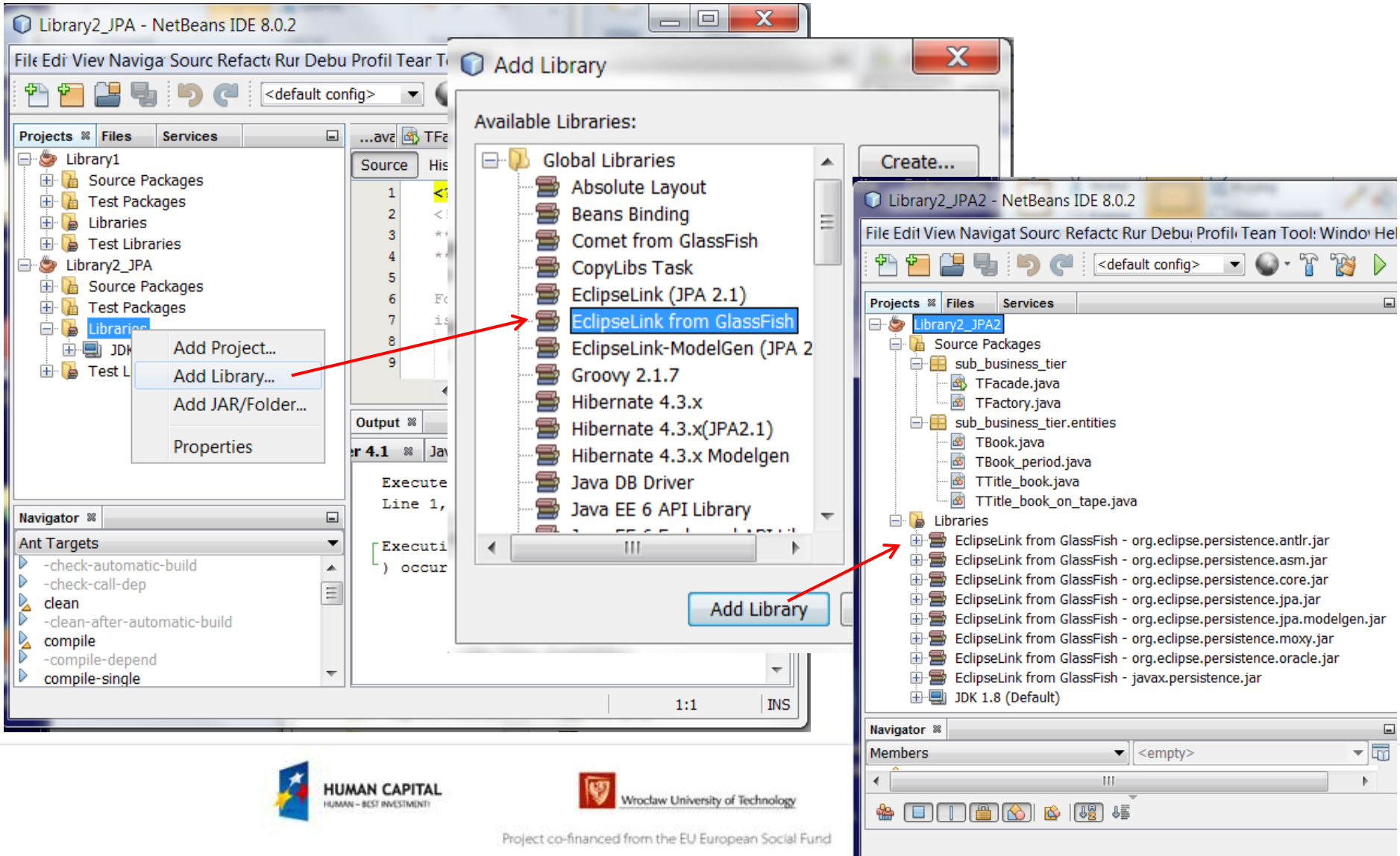


The input class diagram for mapping to entity type classes (3.1)

3.2. Creation the copy of the Library1 project (lab1) as the Library2_JPA project – right click the Library1 project item in the Projects tab, choose the Copy item; fill the name, select the location of new project and click Copy



3.3. Add libraries of EclipseLink (JPA 2.1) technologies to the Library2_JPA project – right click the Libraries folder, select the Add Library… item; choose the EclipseLink from GlassFish and click AddLibrary



Library2_JPA2 - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config>

Projects | Files | Services

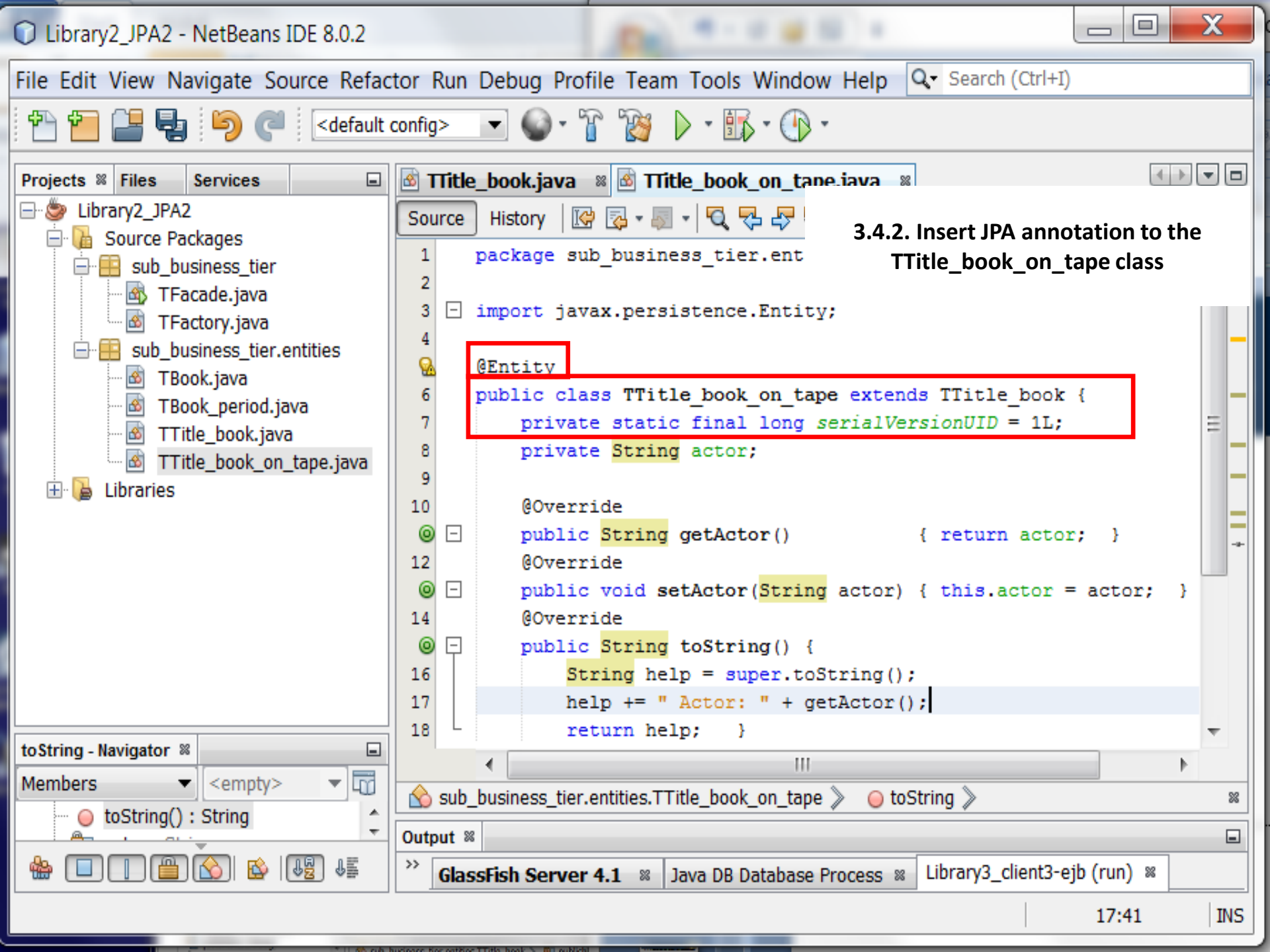
- Library1_client1_SE
- Library2_Client2-ejb
- Library2_EJB2
- Library2_EJB2-ejb
- Library2_EJB2-war
- Library2_interface2-ejb
- Library2_JPA2
  - Source Packages
    - sub_business_tier
    - sub_business_tier.entities
      - TBook.java
      - TBook_period.java
      - TTitle_book.java
      - TTitle_book_on_tape.java
  - Libraries
- Sprzedaz_web

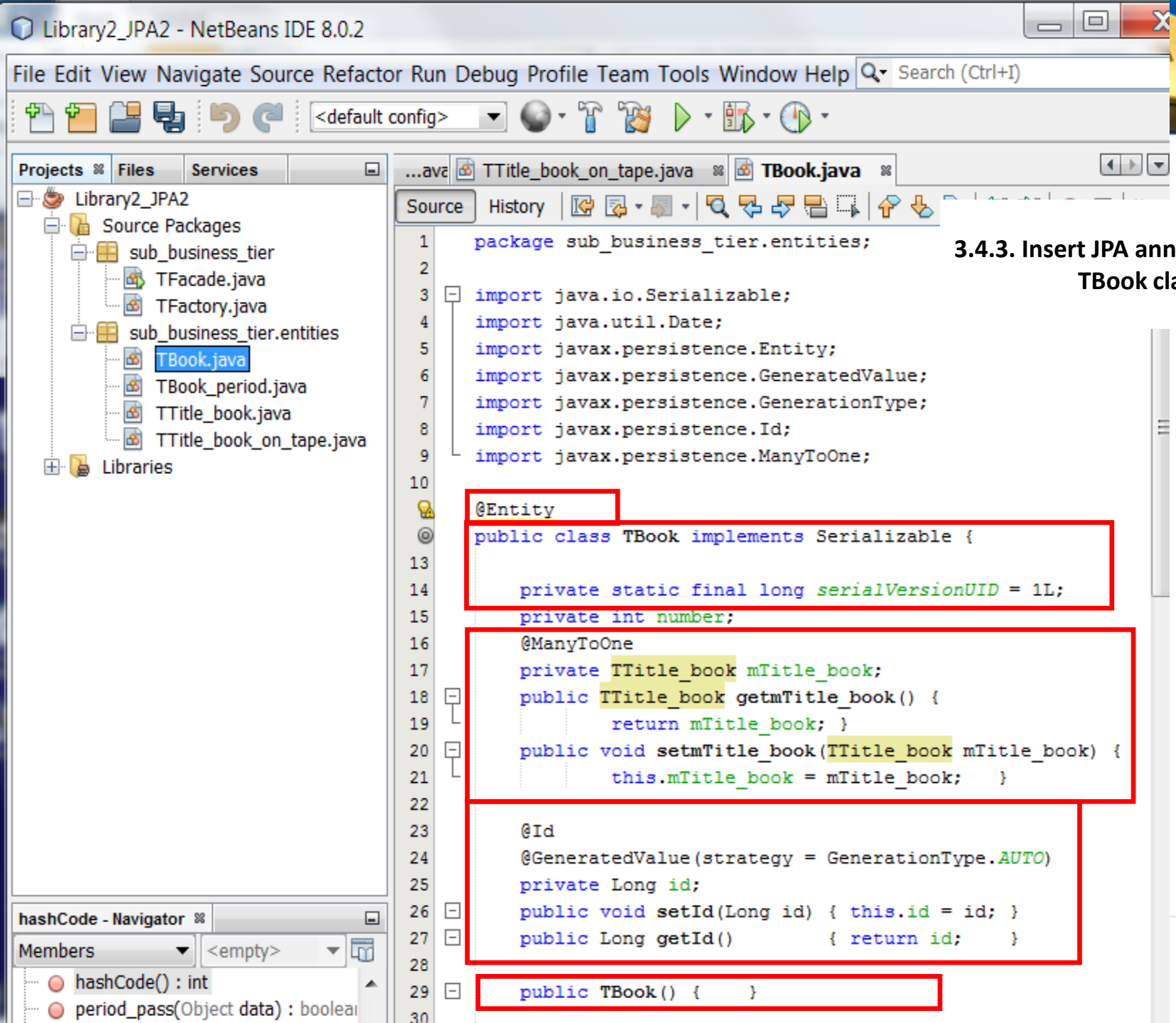
...ava | TTitle_book_on_tape.java | TTitle_book.java | Book_form.java | Card0.jav...

Source | History

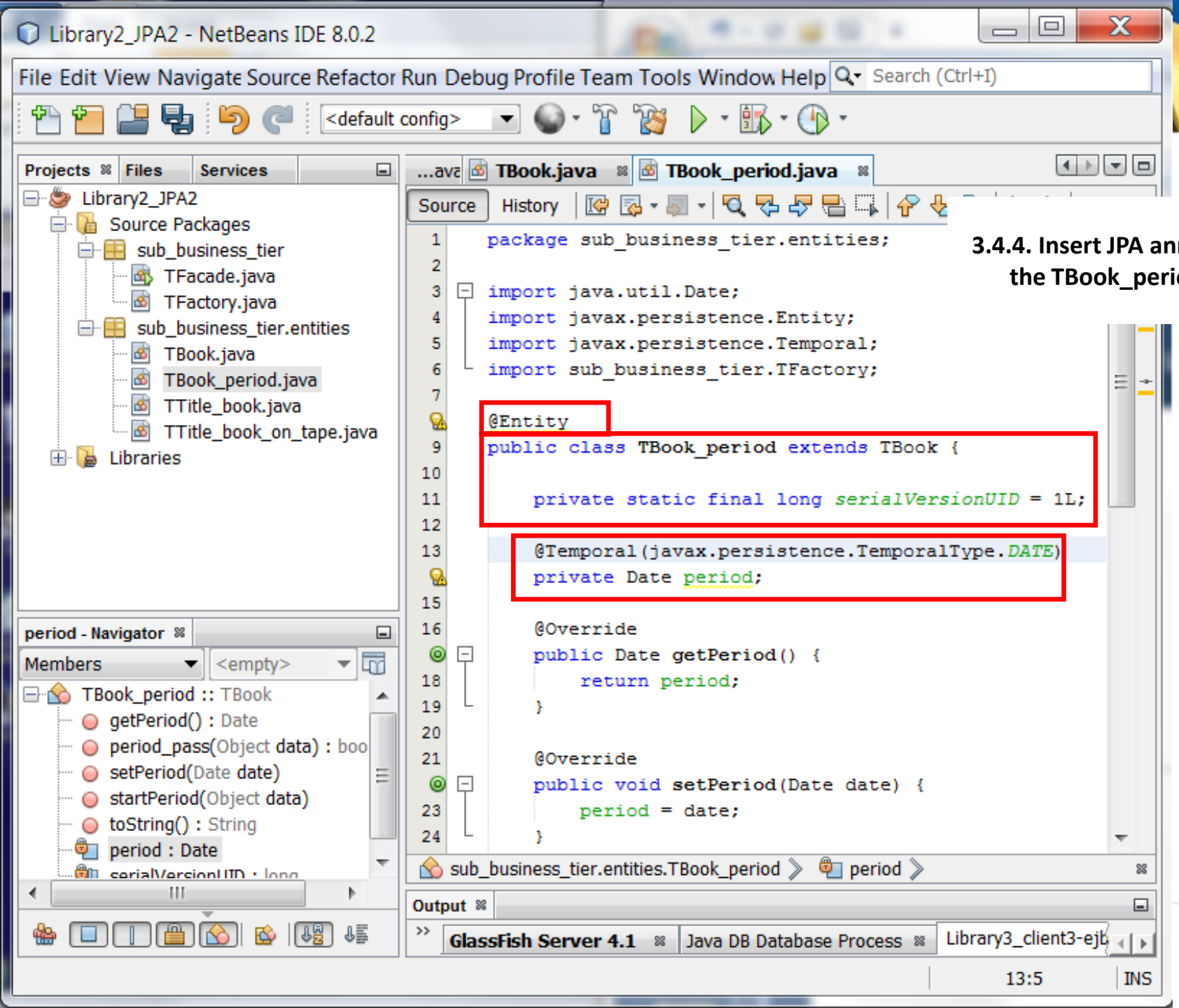
```
1  package sub_business_tier.entities;
2
3  import java.io.Serializable;
4  import java.util.ArrayList;
5  import java.util.Collection;
6  import java.util.Iterator;
7  import java.util.List;
8  import java.util.Objects;
9  import javax.persistence.Entity;
10 import javax.persistence.GeneratedValue;
11 import javax.persistence.GenerationType;
12 import javax.persistence.Id;
13 import javax.persistence.OneToMany;
14 import javax.persistence.Transient;
15 import sub_business_tier.TFactory;
16
   @Entity
   public class TTitle_book implements Serializable {
19     private static final long serialVersionUID = 1L;
20     private String publisher;
21     private String ISBN;
22     private String title;
23     private String author;
24     @Id
25     @GeneratedValue(strategy = GenerationType.AUTO)
26     private Long id;
27     public Long getId()              {  return id;   }
28     public void setId(Long val)      {  id = val;    }
29
30     @OneToMany(mappedBy = "mTitle_book")
31     private Collection<TBook> books;
32     public Collection<TBook> getBooks()          { return books;    }
33     public void setBooks(Collection<TBook> val) { books = val;    }
34
35     @Transient
36     List<TBook> mBooks;
37     public List<TBook> getmBooks()            {  return mBooks;    }
38     public void setmBooks(List<TBook> mBooks) {  this.mBooks = mBooks; }
39
40     public TTitle_book() {    mBooks = new ArrayList<>();    }
41
```

3.4. /3.4.1. Insert JPA annotation to the TTitle_book class


Library2_JPA2 - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
Library2_JPA2
Source Packages
sub_business_tier
TFacade.java
TFactory.java
sub_business_tier.entities
TBook.java
TBook_period.java
TTitle_book.java
TTitle_book_on_tape.java
Libraries
TTitle_book.java
TTitle_book_on_tape.java
Source
History
1 package sub_business_tier.ent
2
3 import javax.persistence.Entity;
4
@Entity
6 public class TTitle_book_on_tape extends TTitle_book {
7 private static final long serialVersionUID = 1L;
8 private String actor;
9
10 @Override
public String getActor() { return actor; }
12 @Override
public void setActor(String actor) { this.actor = actor; }
14 @Override
public String toString() {
16 String help = super.toString();
17 help += " Actor: " + getActor();
18 return help; }
3.4.2. Insert JPA annotation to the TTitle_book_on_tape class
toString - Navigator
Members
<empty>
toString() : String
sub_business_tier.entities.TTitle_book_on_tape
toString
Output
GlassFish Server 4.1
Java DB Database Process
Library3_client3-ejb (run)
17:41
INS

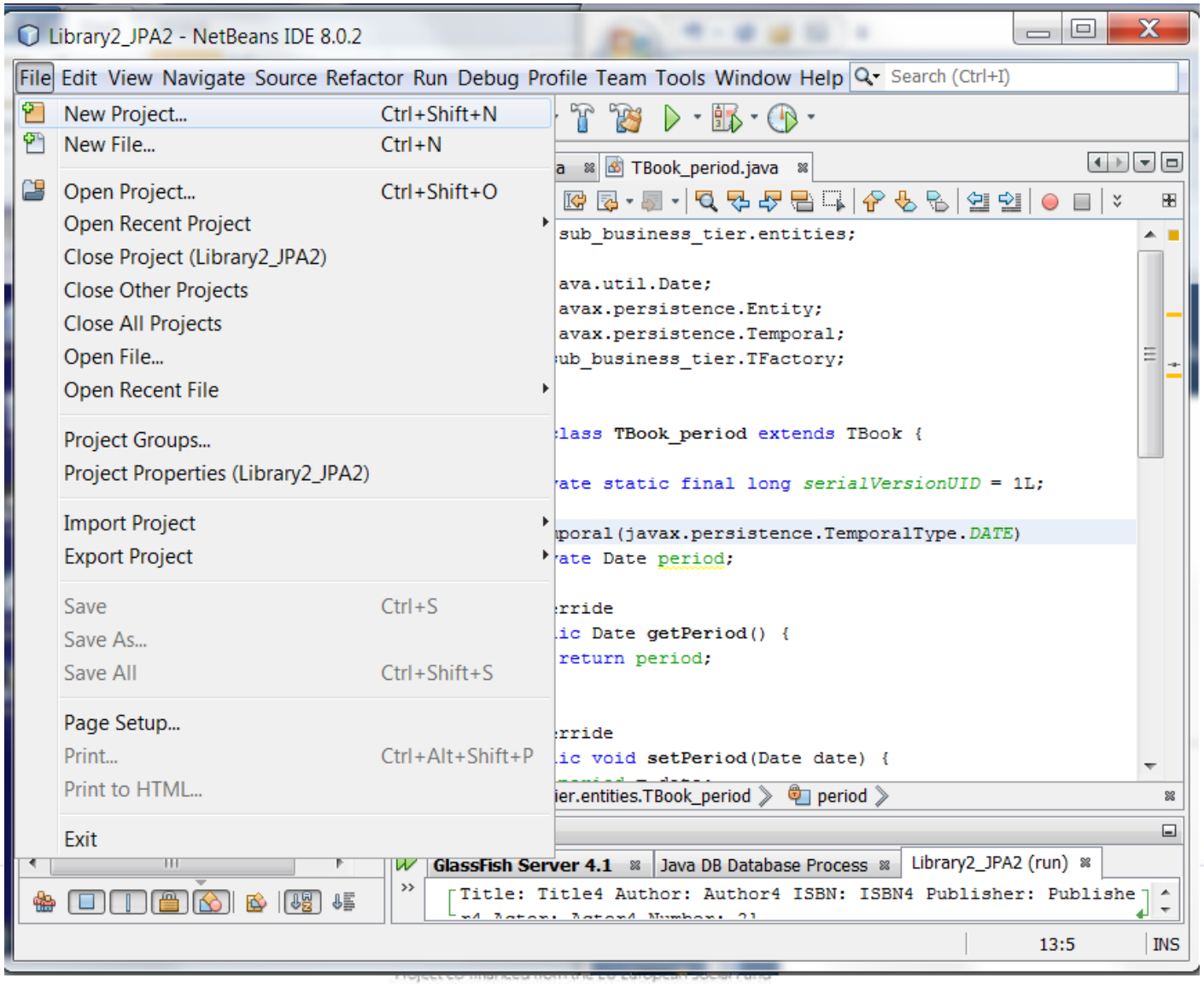


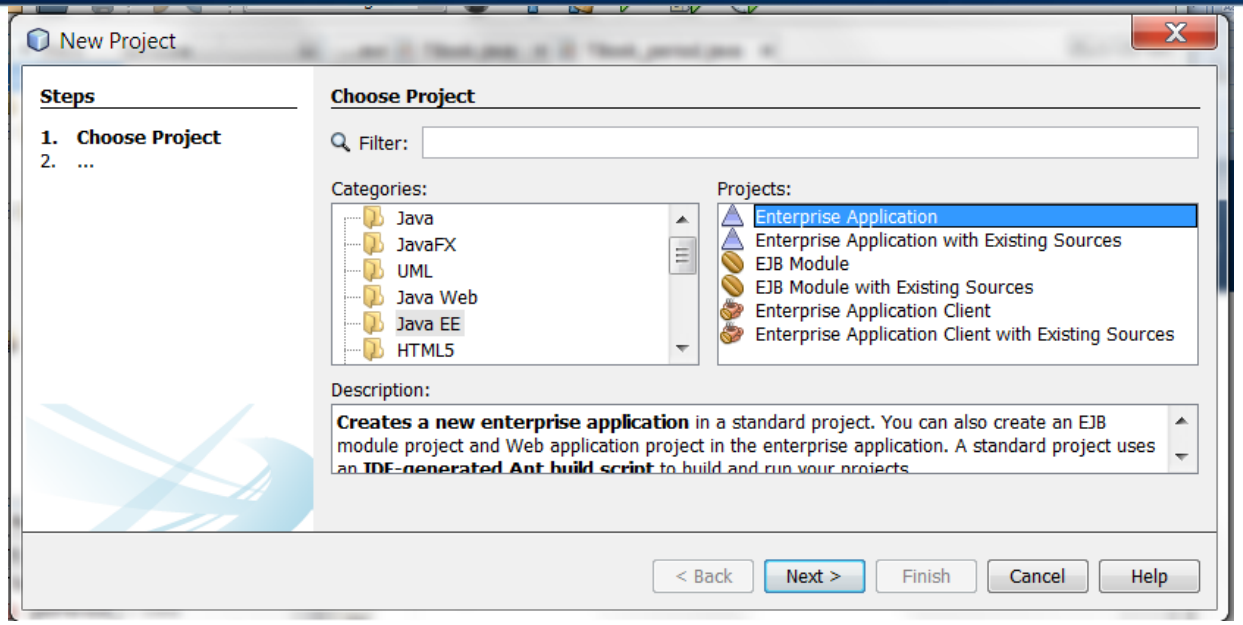
3.4.3. Insert JPA annotation to the TBook class



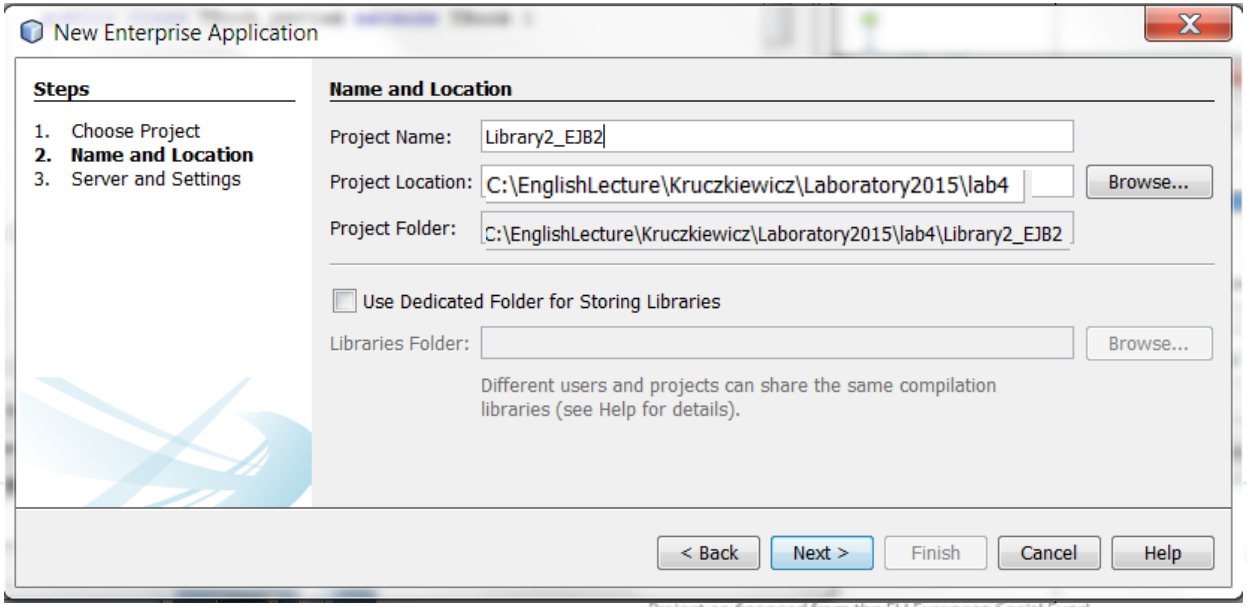
3.4.4. Insert JPA annotation to the TBook_period class

4. /4.1./4.1.1. Create the new Enterprise Application project – click File/New Project…



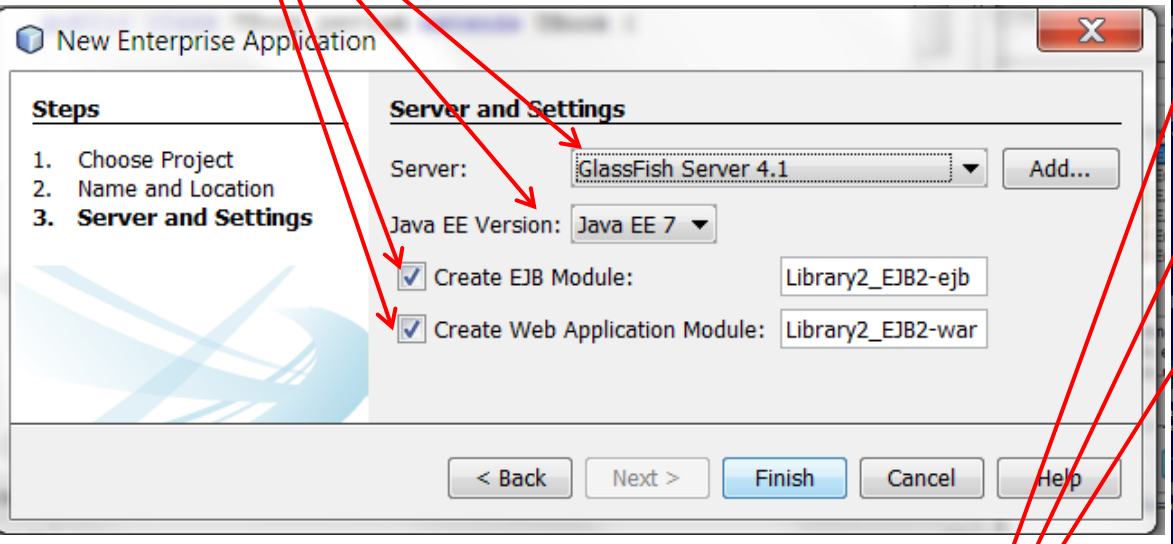


4.1.2. Select the Java EE/ Enterprise Application options

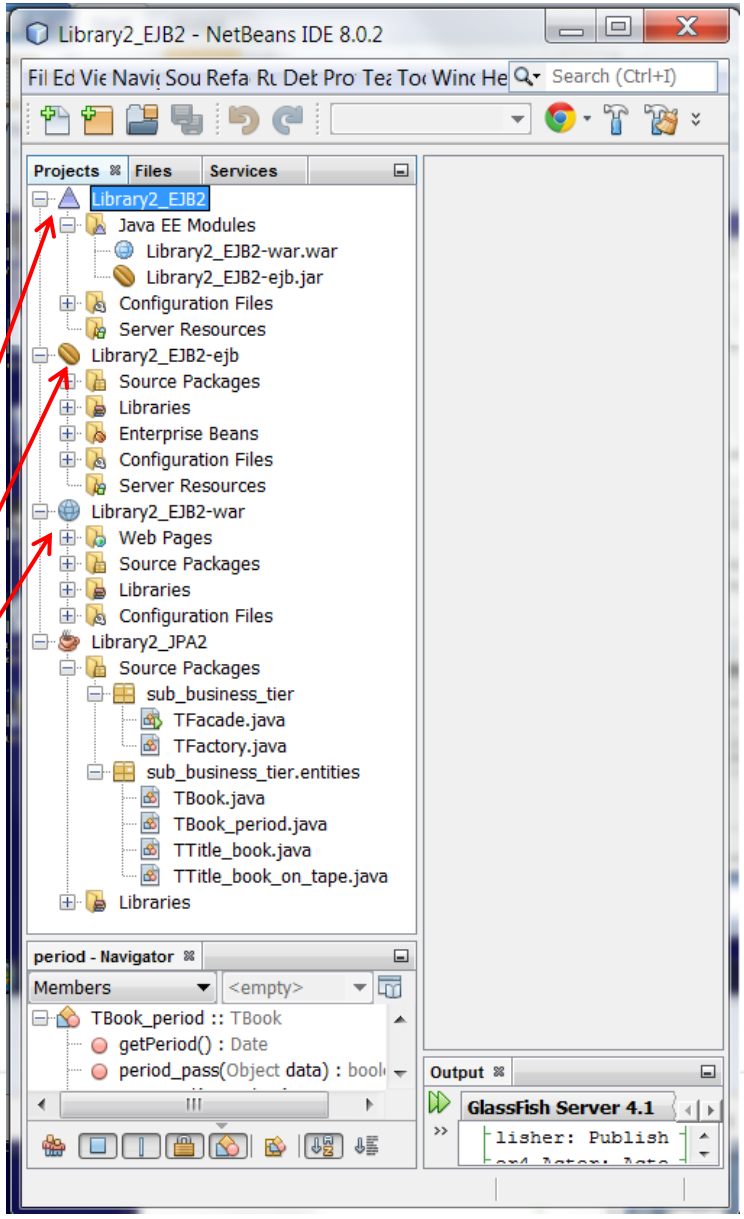


4.1.3. Fill the Library2_EJB2 name of Java EE project and select location of this project

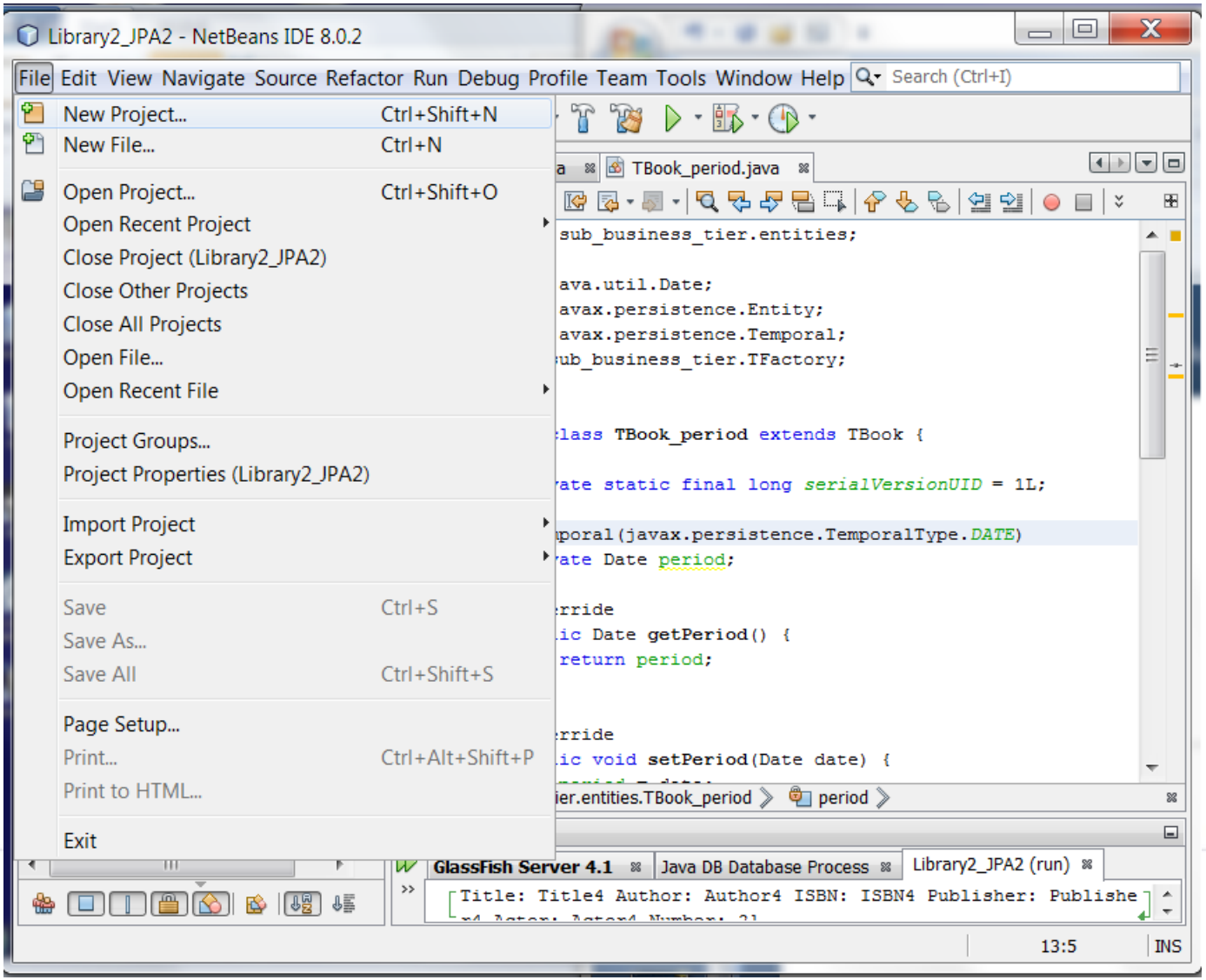
4.1.4. Select the Application Server, JavaEE 7 platform and two types of modules: EJB modelu and Web Application module.

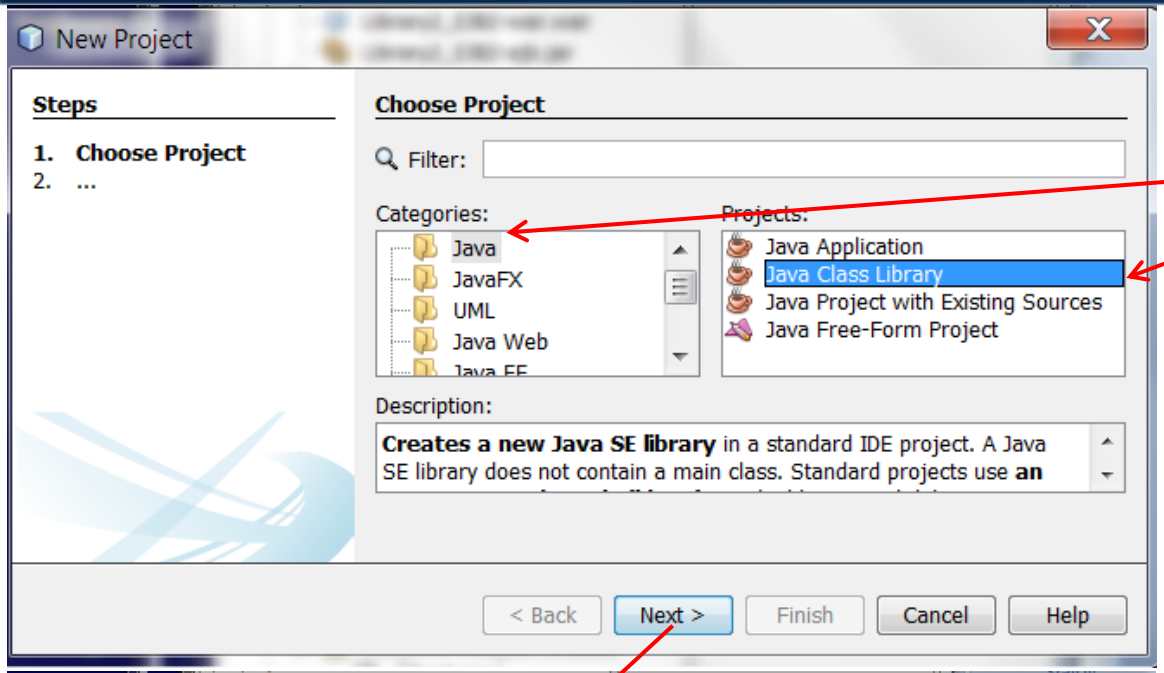


4.1.5. Result

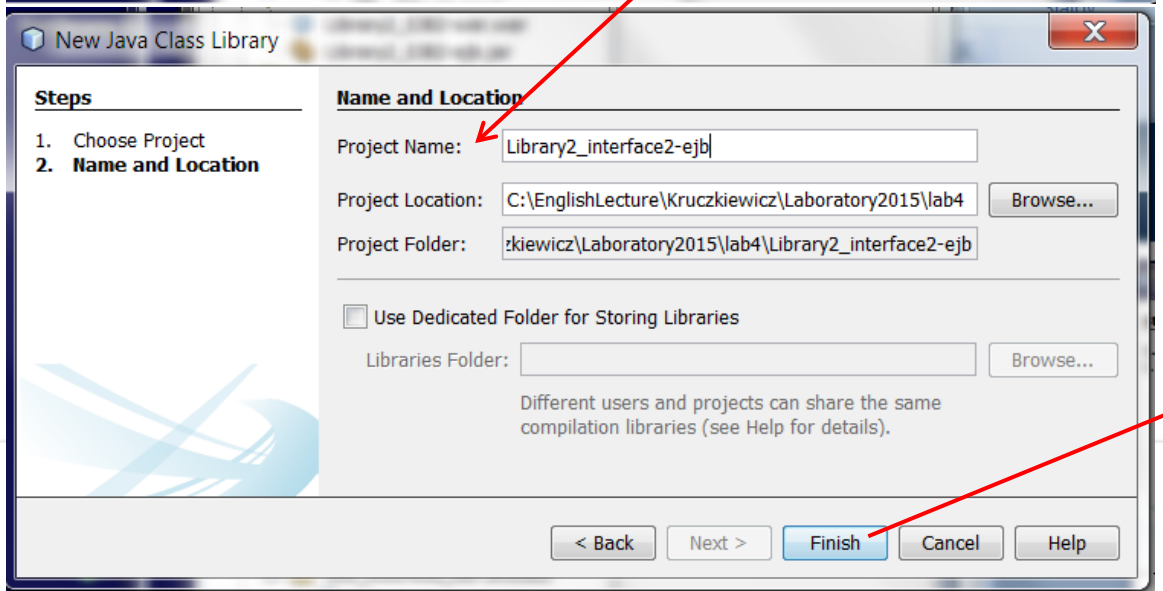


4.2./4.2.1. Create the Java Class Library project as the interface for remote Session Bean in the Library2_EJB2-ejb module of the Library2_EJB2 Enterprise Application project – click File/New Project…

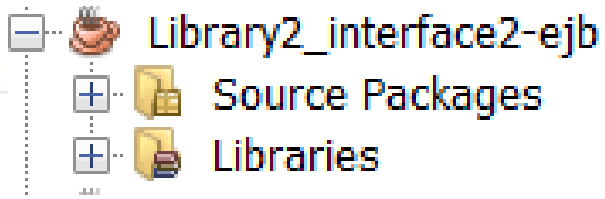




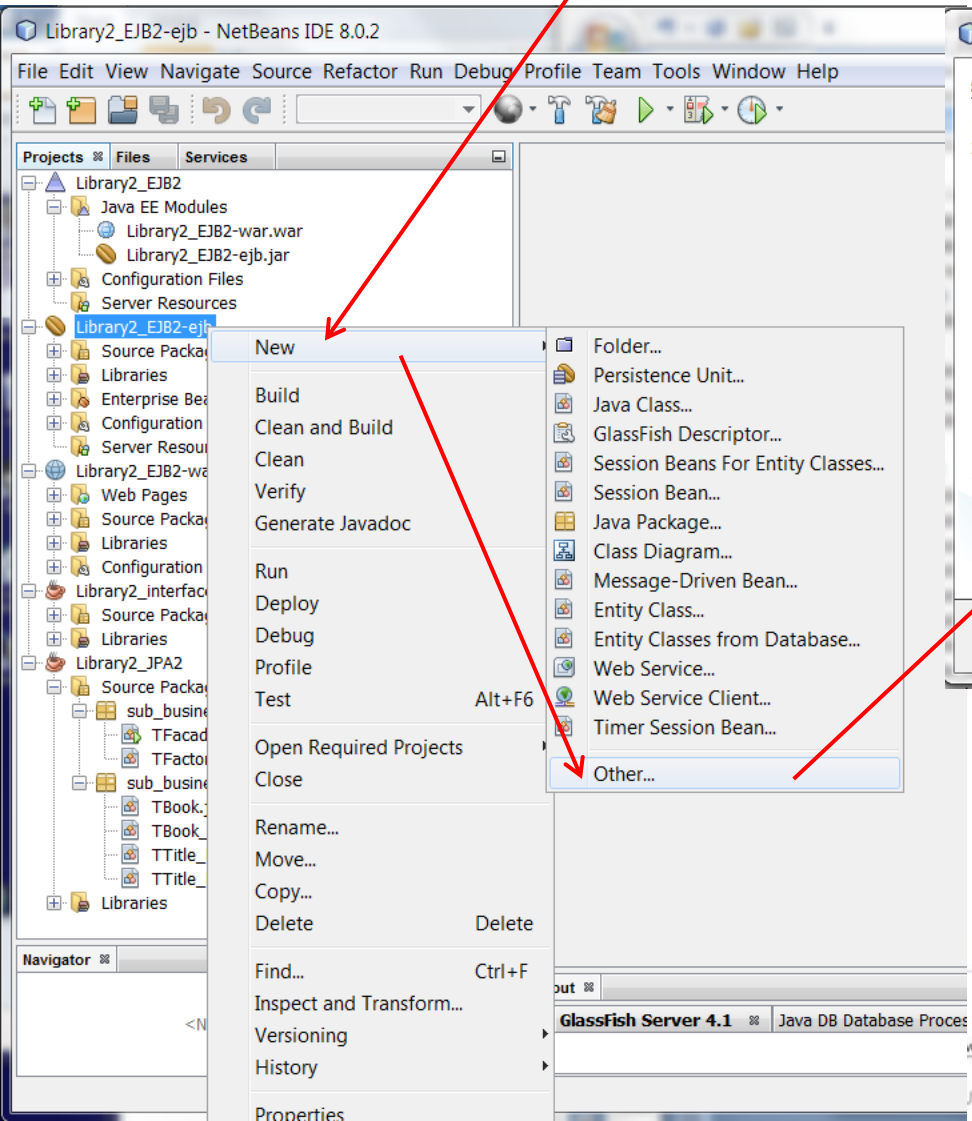
4.2.2. Select the Java/ Java Class Library options and click Next

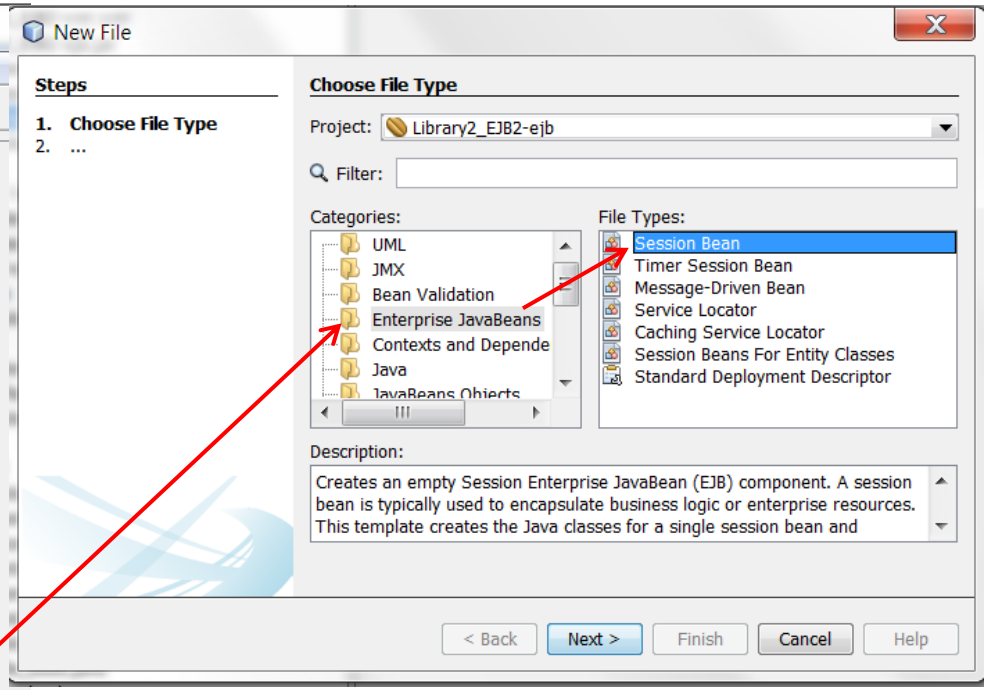


4.2.3. Fill the Library2_interface2-ejb name of Java Class Library project , select location of this project and click Finish. The result:

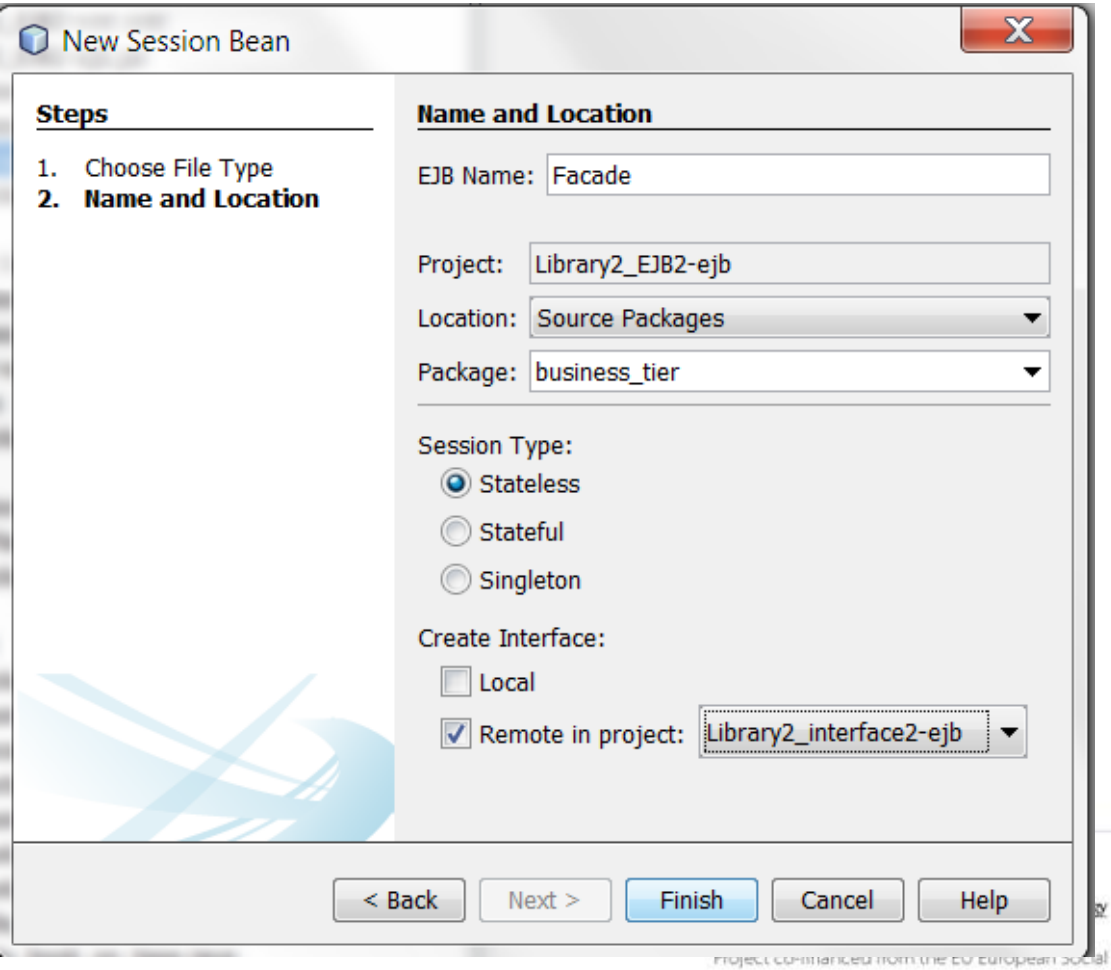


4.3./4.3.1. Create the remote Session Bean in the Enterprise Application project – right click the Library2_EJB2-ejb module in the Projects tab and select the New/Other items; select the Enterprise JavaBeans and Session Bean options; click Next

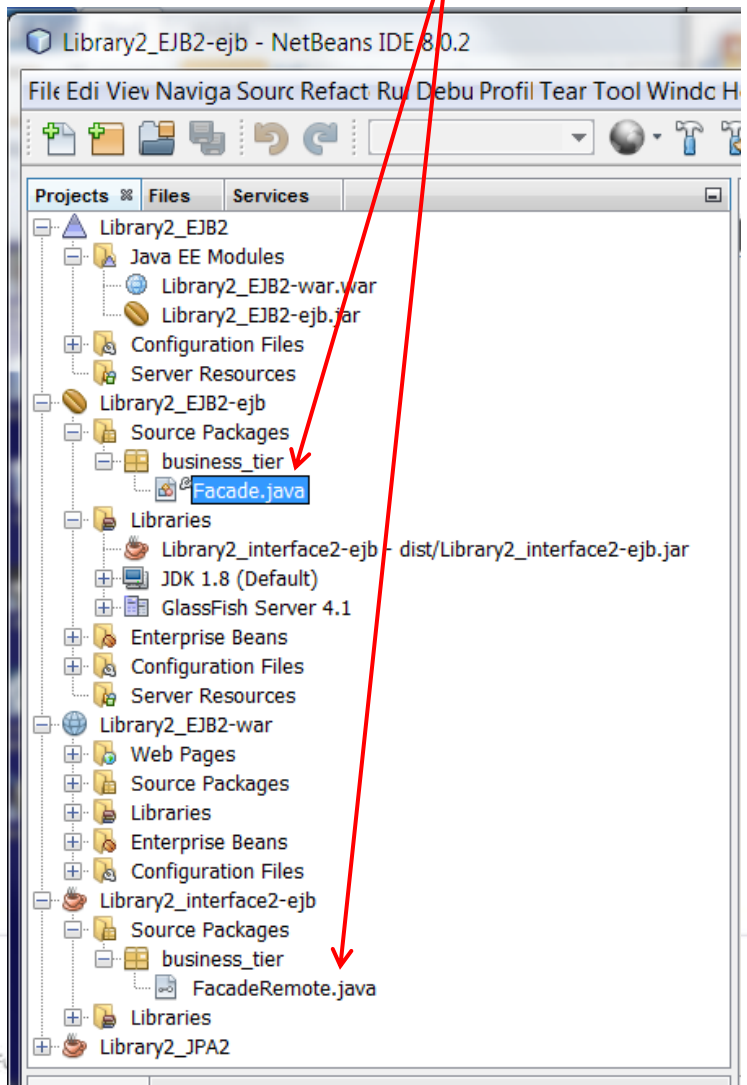




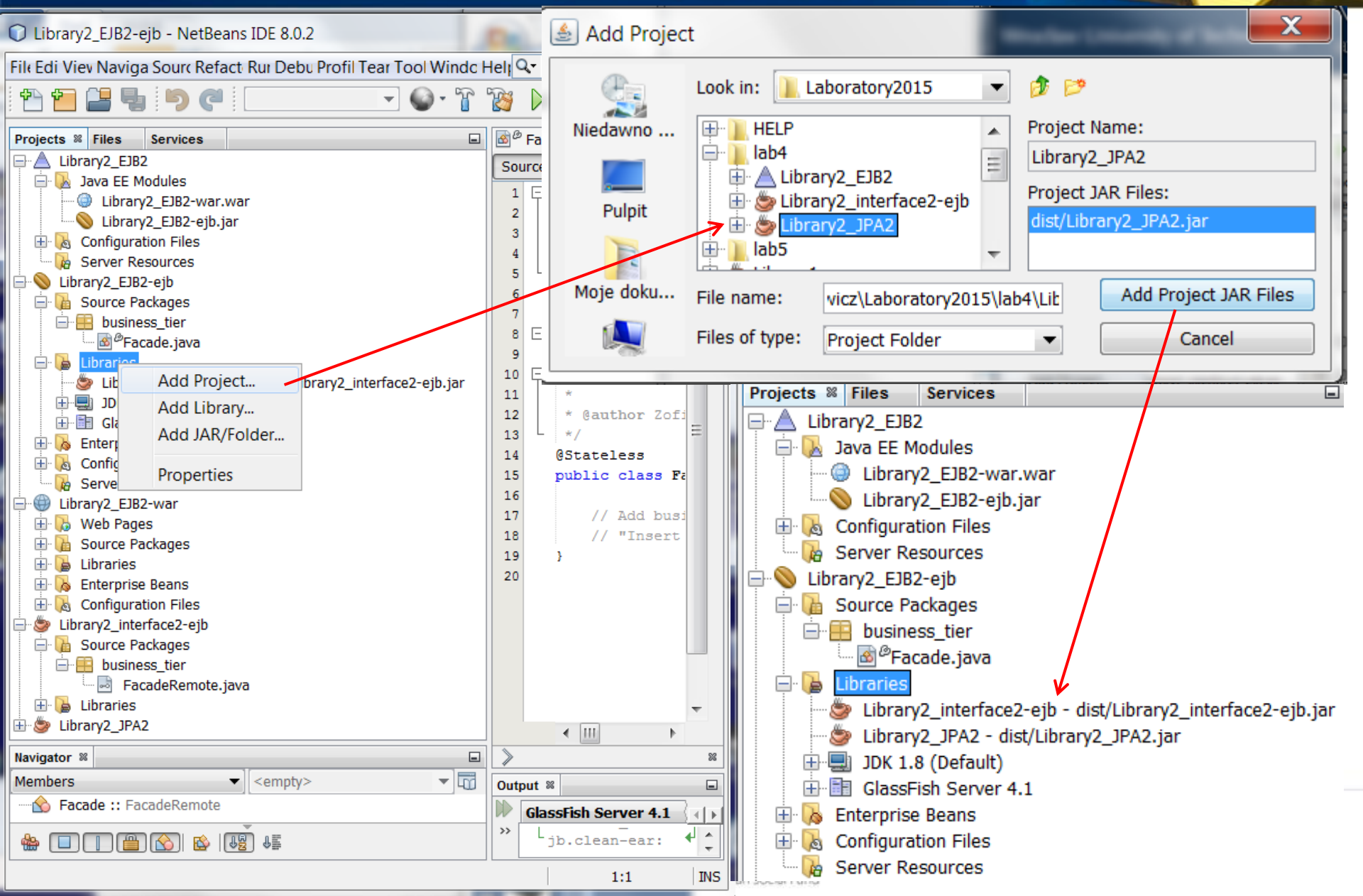
4.3.2. Fill the name of the remote Session Bean; create the new package (business_tier) in the Library2_EJB2-ejb Enterprise Application; select the the Stateless Session type; select the Remote and the Library2_interface2-ejb interface for the Facade as the remote Session Bean



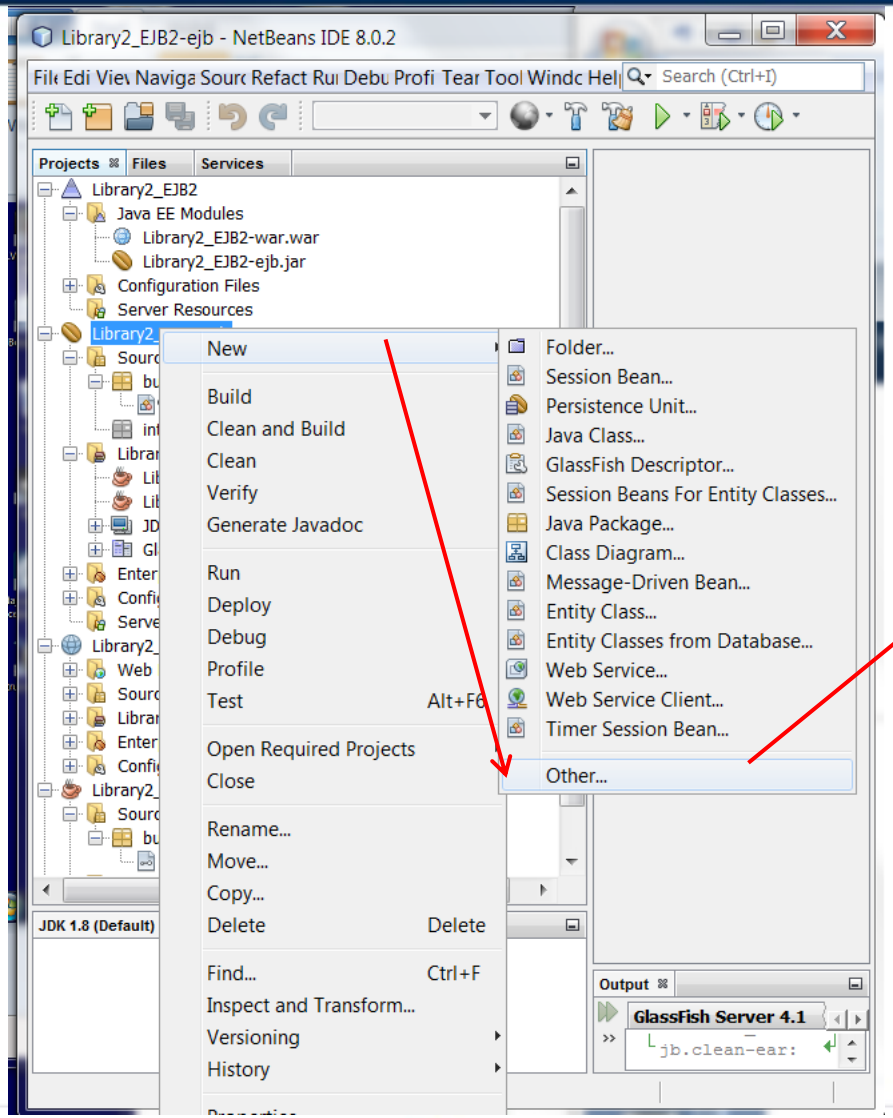
4.3.3. The result

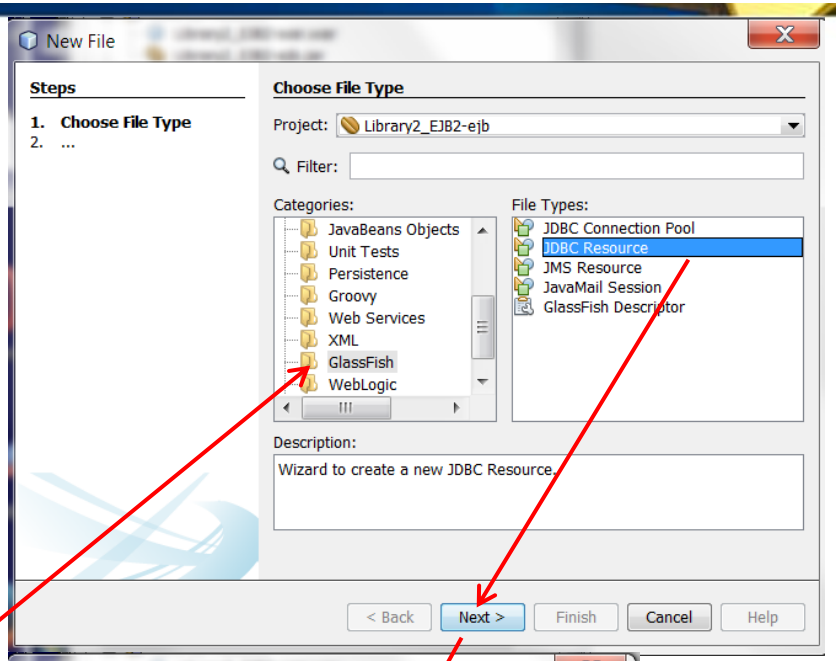


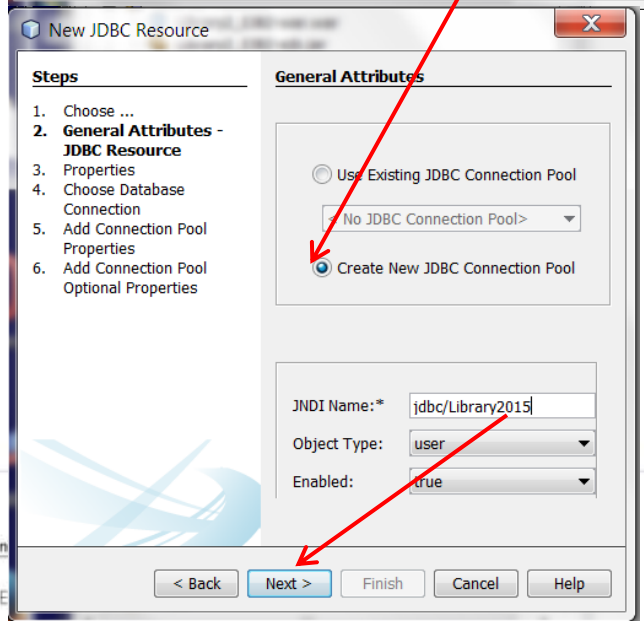
4.4. Add the access to the business classes of the Library2_JPA2 Java Application Project (SE type) in the Library2_EJB2-ejb Enterprise Application – right click the Libraries folder and select the Add Project.. item



4.5./ 4.5.1 Create the JDBC resource of GlassFish for the access to the database by the Library2_EJB2-ejb Enterprise Application – right click the project item in the Projects tab, select: New/Other/GlassFish/JDBC Resource/Next/ Create New JDBC Connection Pool and fill the JNDI Name/Next

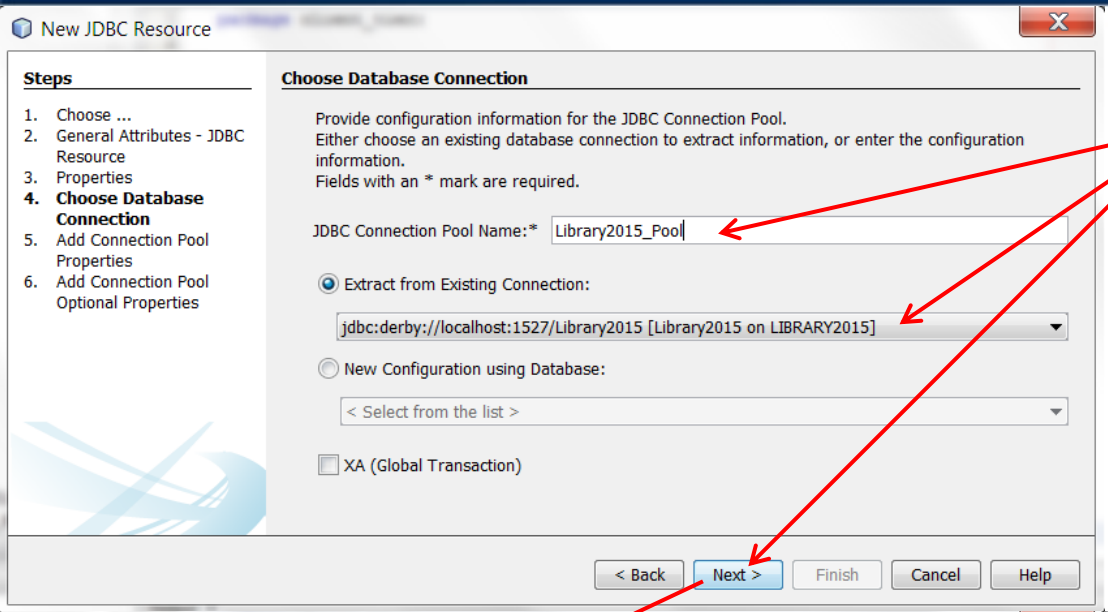


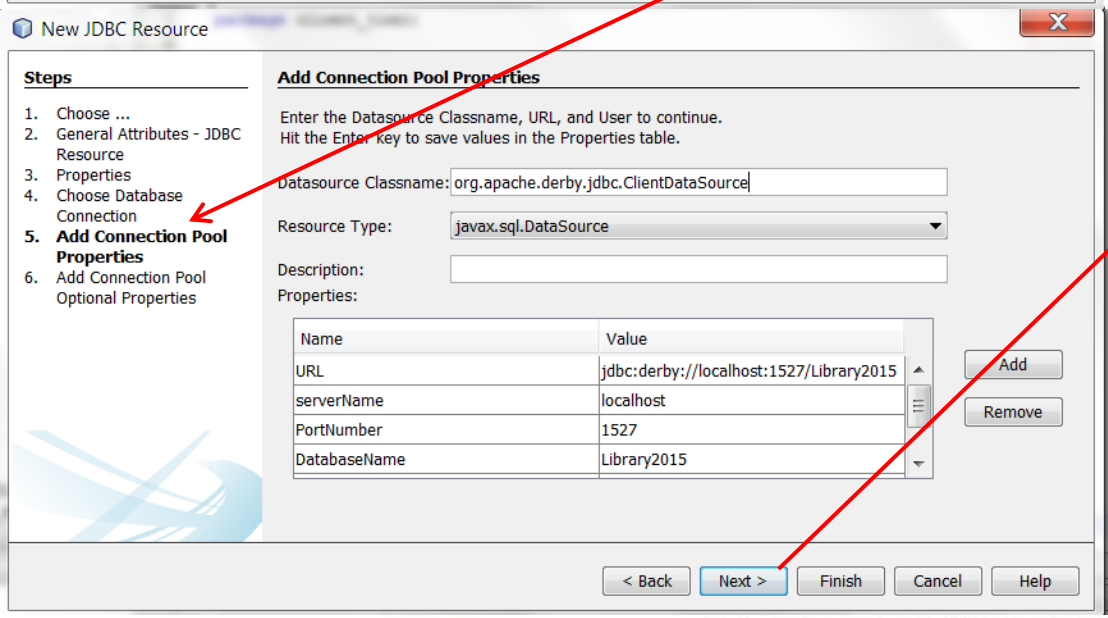


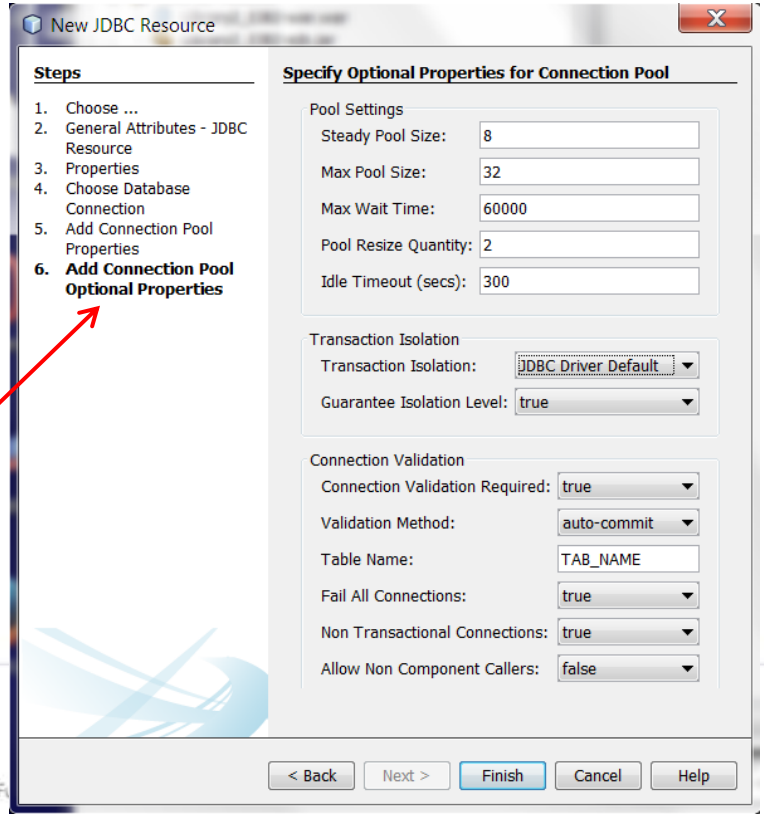


4.5.2. Continuation

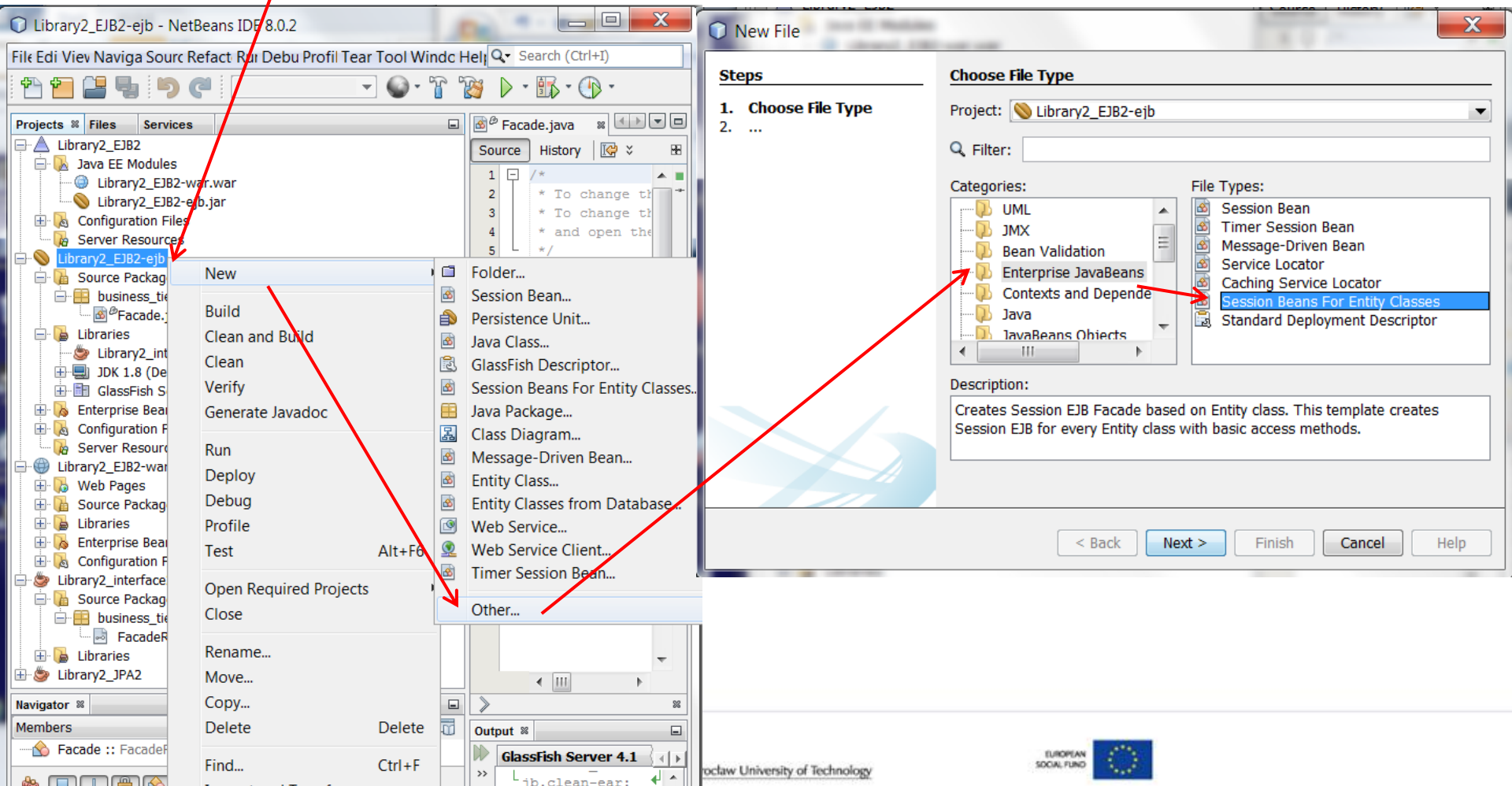
1) Go to the Choose Database Connection option: fill the JDBC Connection Pool Name, select the connection to Data source, created during the first step of this instruction (3 slide);
2) Add Connection Pool Properties
3) Add Connection Pool Optional Properties: click Finish



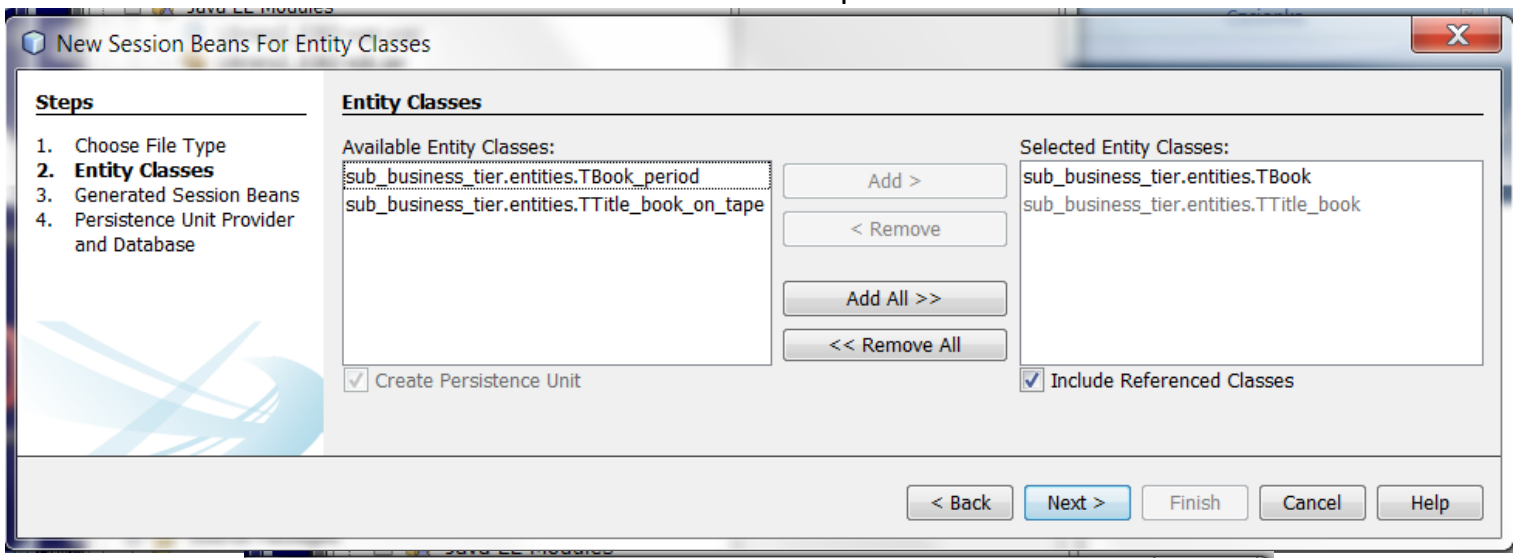


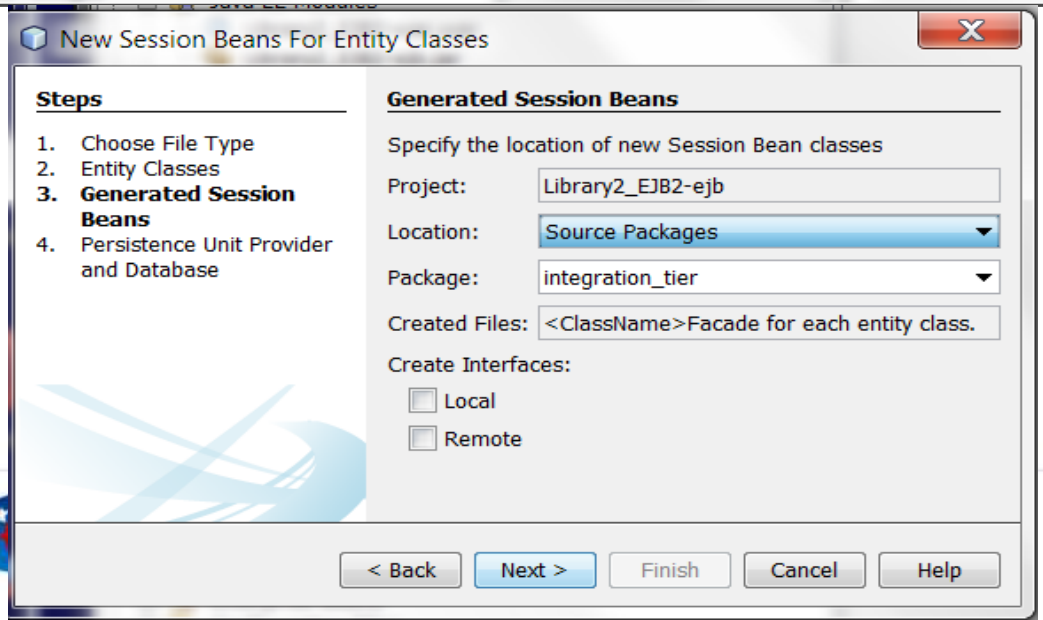


4.6./4.6.1. Create the local Session Bean for Entity Classes in the Library2_EJB2-ejb Enterprise Application project – right click the Library2_EJB2-ejb module in the Projects tab and select the New/Other items; select the Enterprise JavaBeans/Session Bean For Entity Classes options and click Next



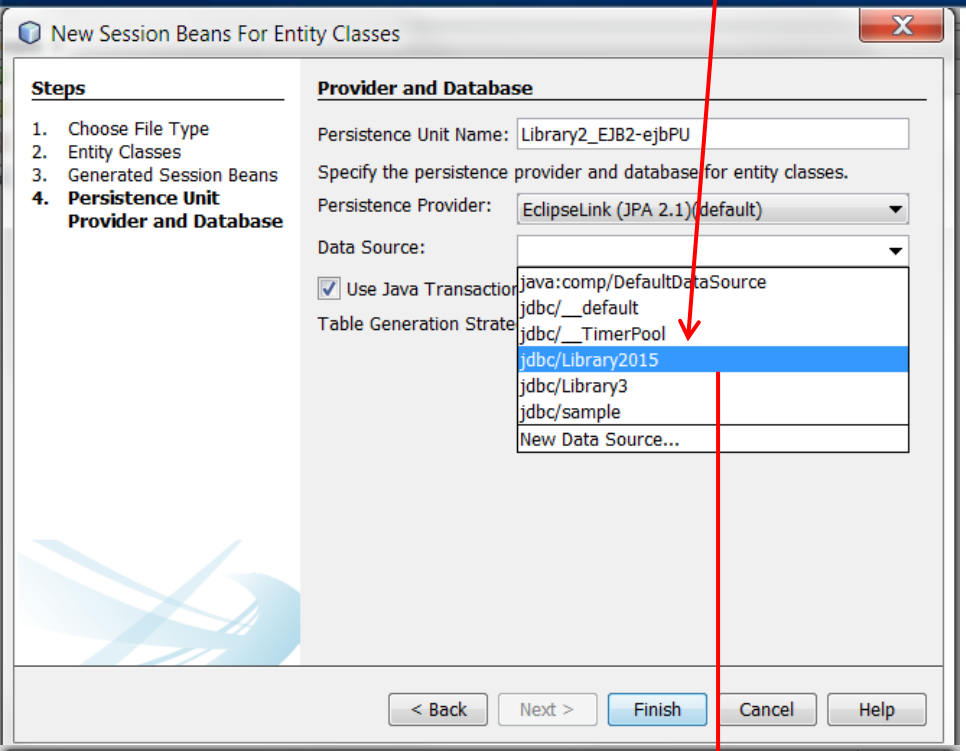
4.6.2. Continuation – select the TTitle_book and TBook entity classes and add to the right with the selected Include Referenced Classes option and click Next

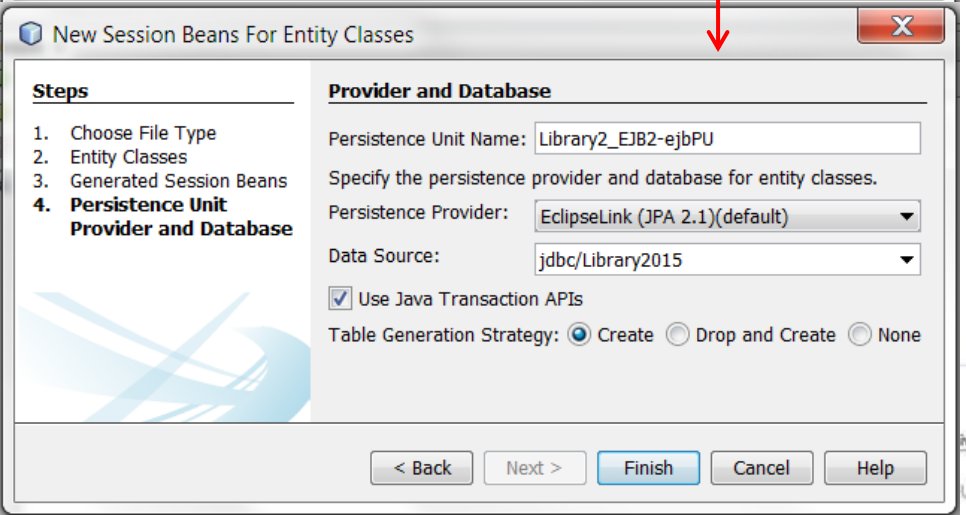


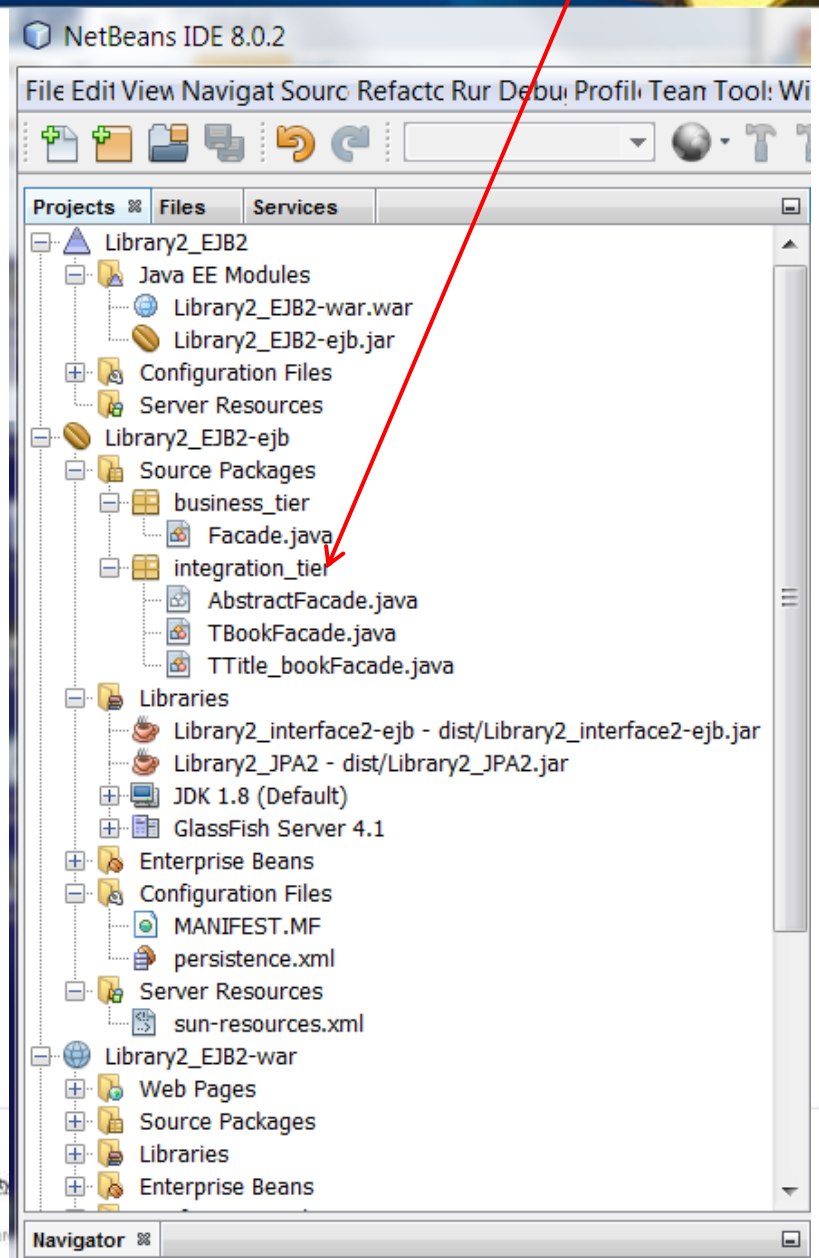


4.6.3. Continuation – fill the field Package with the integration_tier name of the new package for created beans and click Next

4.6.4. Continuation (Persistence Unit Provider and Database) – select the DataSource created during the 4.5 step and click Finish. The result is shown on the right.

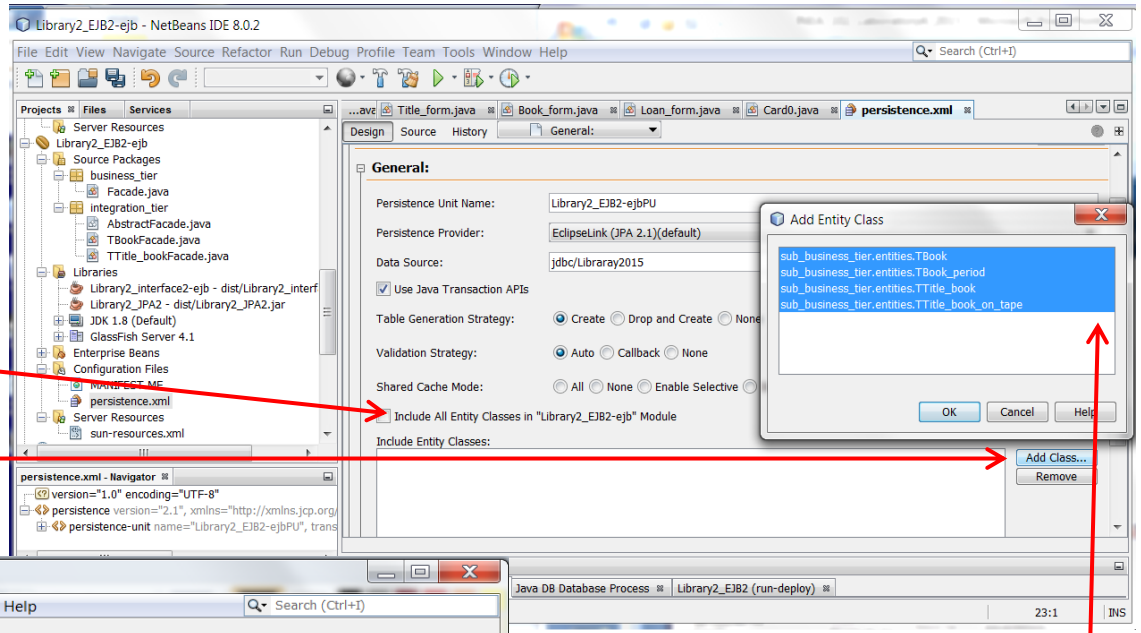


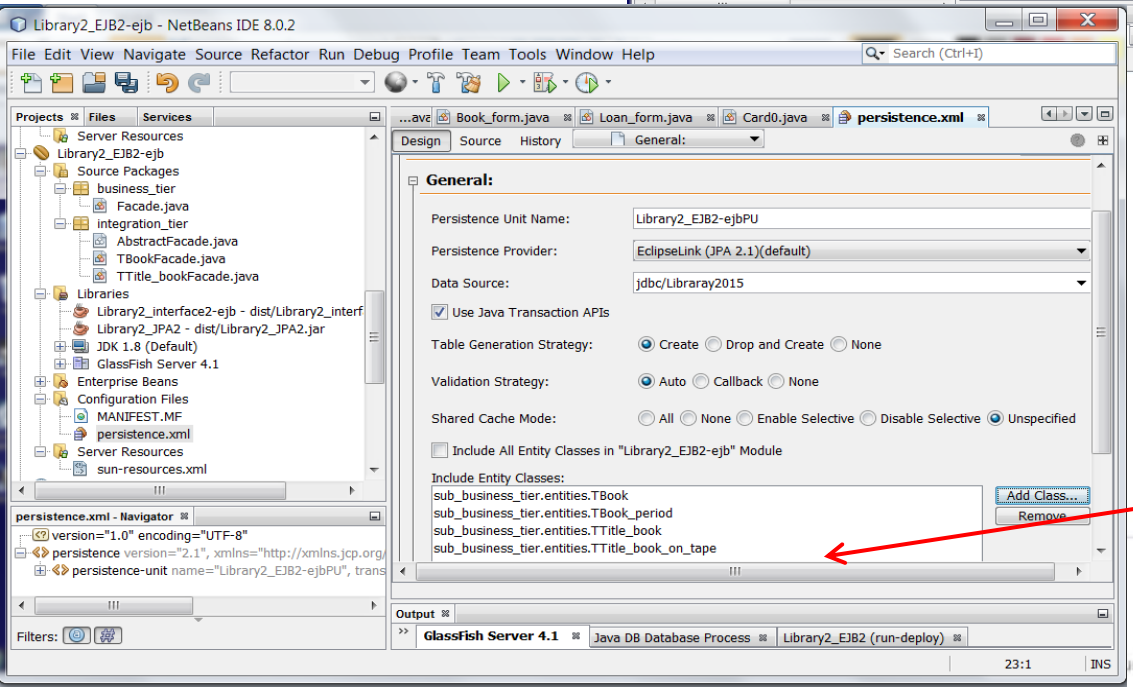




4.7./ 4.7.1 Update the Include Entity Classes option in the created Persistence Unit in the 4.6 step

1. Unselect the **Include All Entity Classes in „Library2_EJB2-ejb" Module**.
2. Click the Add Class..





4.7.2 Continuation

1. Select all entity classes
2. The result

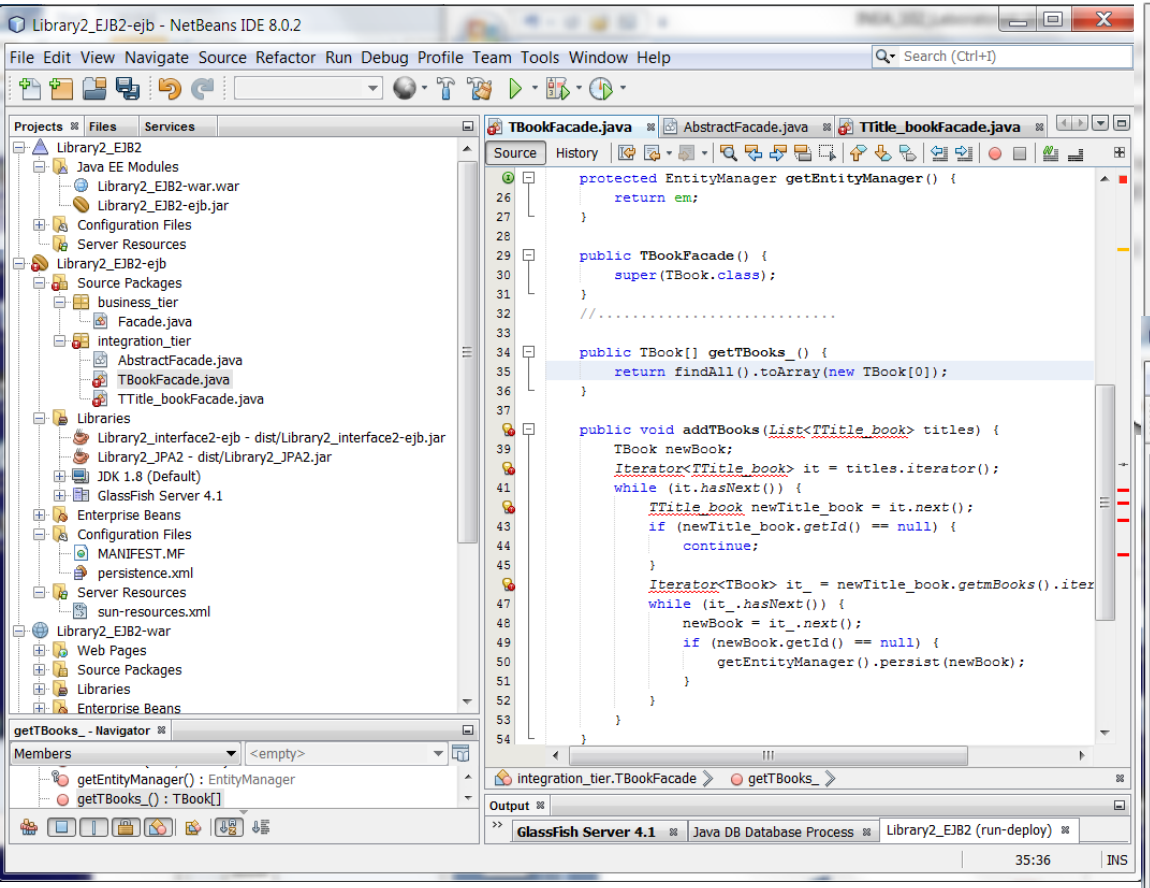
4.8./4.8.1. Update generated code of TTitle_bookFacade Session Bean for TTitle_book and TTitle_book_on_tape entity classes, generated **in the 4.6 steps**

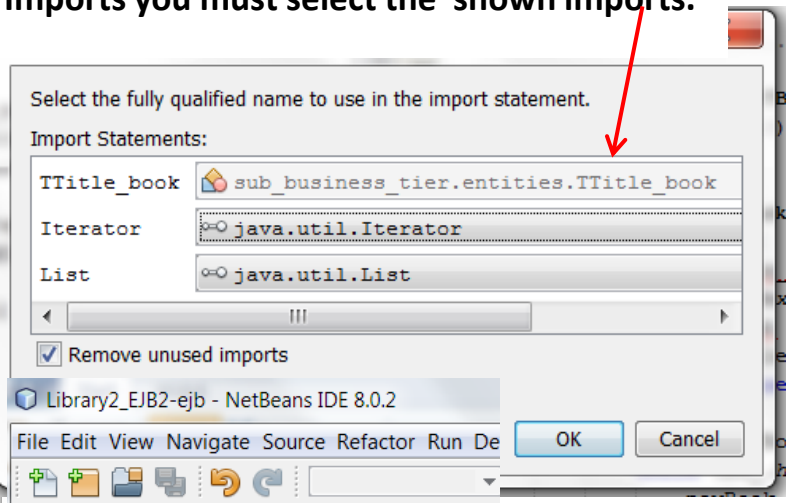
```
package integration_tier;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import sub_business_tier.entities.TTitle_book;
@Stateless
public class TTitle_bookFacade extends AbstractFacade<TTitle_book> {

    @PersistenceContext(unitName = "Library2_EJB2-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {       return em;    }
    public TTitle_bookFacade() {        super(TTitle_book.class);    }
    //Added code
    public TTitle_book[] getTTitle_books_() {
      return findAll().toArray(new TTitle_book[0]);
    }
    public void addTTitle_books(List<TTitle_book> titles) {
       TTitle_book newTTitle_book;
      Iterator<TTitle_book> it = titles.iterator();
       while (it.hasNext()) {
          newTTitle_book = it.next();
          if (newTTitle_book.getId() == null) {
            getEntityManager().persist(newTTitle_book);
          }
       }
    }
}
```

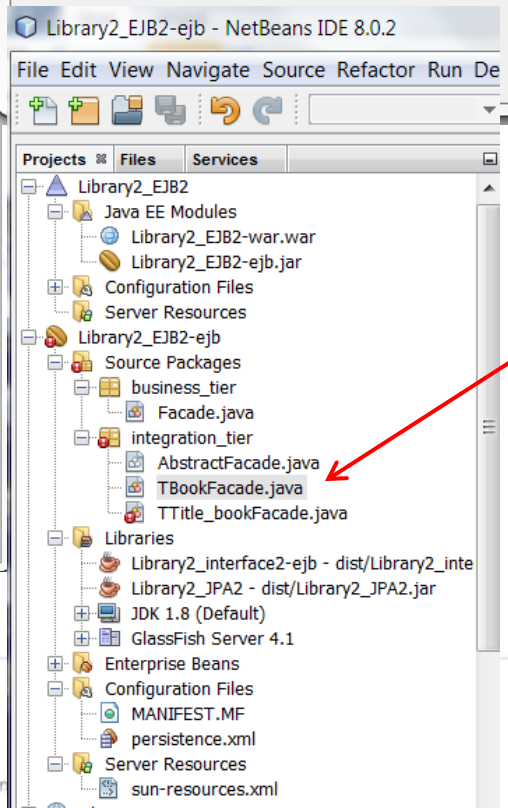
4.8.2. Update generated code of TBookFacade Session Bean for TBook and TBook_period entity classes, generated **in the 4.6 steps**

```
package integration_tier;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import sub_business_tier.entities.TBook;
import sub_business_tier.entities.TTitle_book;

@Stateless
public class TBookFacade extends AbstractFacade<TBook> {

    @PersistenceContext(unitName = "Library2_EJB2-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {       return em;    }
    public TBookFacade()                               {       super(TBook.class);    }
    //.............................
    public TBook[] getTBooks_()   {       return findAll().toArray(new TBook[0]);  }

    public void addTBooks(List<TTitle_book> titles) {
        TBook newBook;
        Iterator<TTitle_book> it = titles.iterator();
        while (it.hasNext()) {
            TTitle_book newTitle_book = it.next();
            if (newTitle_book.getId() == null) {
                continue;   }
            Iterator<TBook> it_ = newTitle_book.getmBooks().iterator();
            while (it_.hasNext()) {
                newBook = it_.next();
                if (newBook.getId() == null) {
                    getEntityManager().persist(newBook);    }
            }
        }
    }
}
```

4.8.3. Right click and select the Fix Imports.. item. In the form of Fix Imports you must select the shown imports.

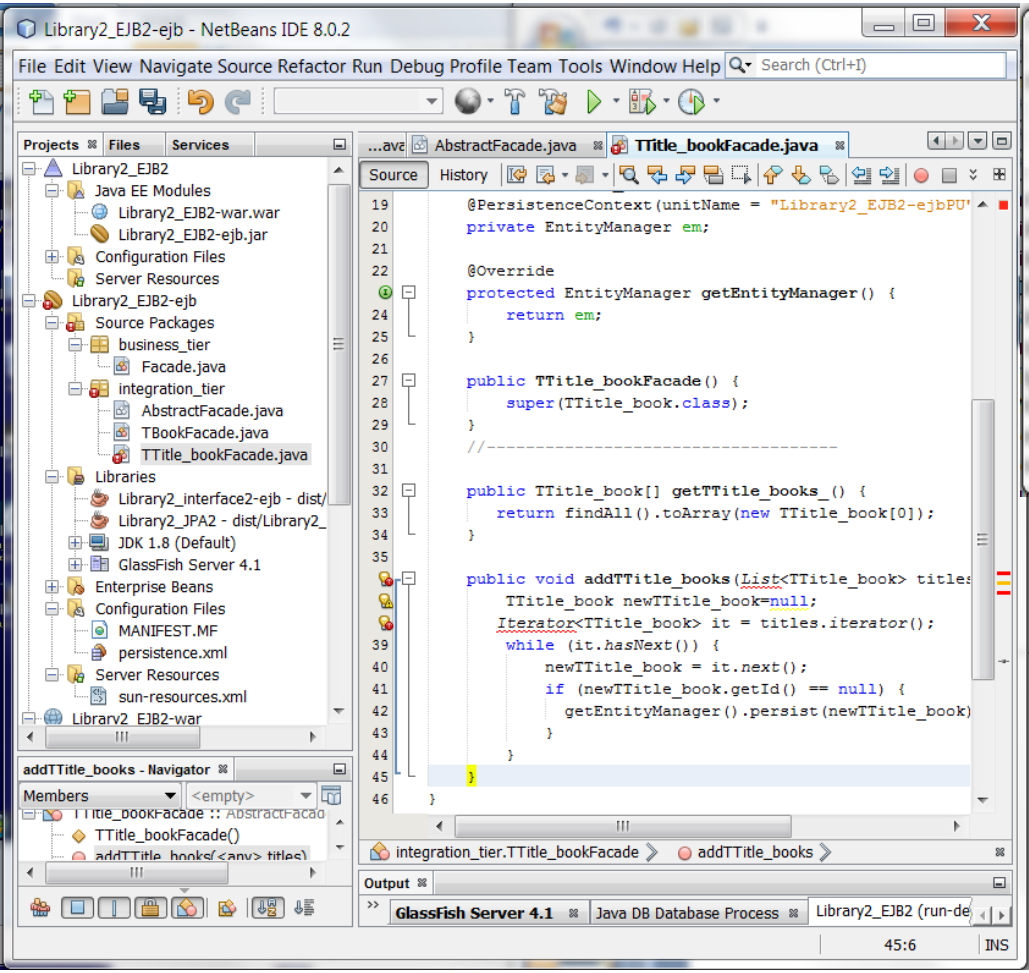


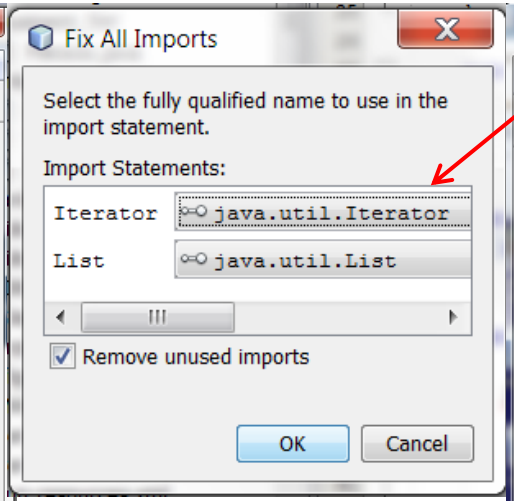




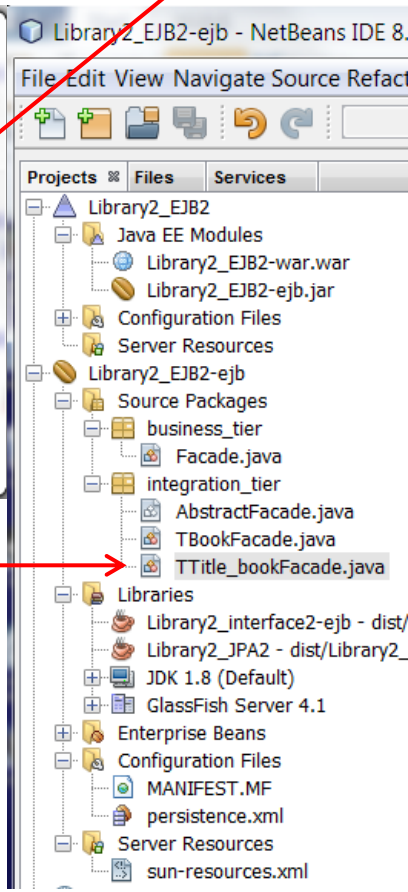
4.8.4. The result

4.8.5. Right click and select the Fix Imports.. item. In the form of Fix Imports you must select the shown imports.



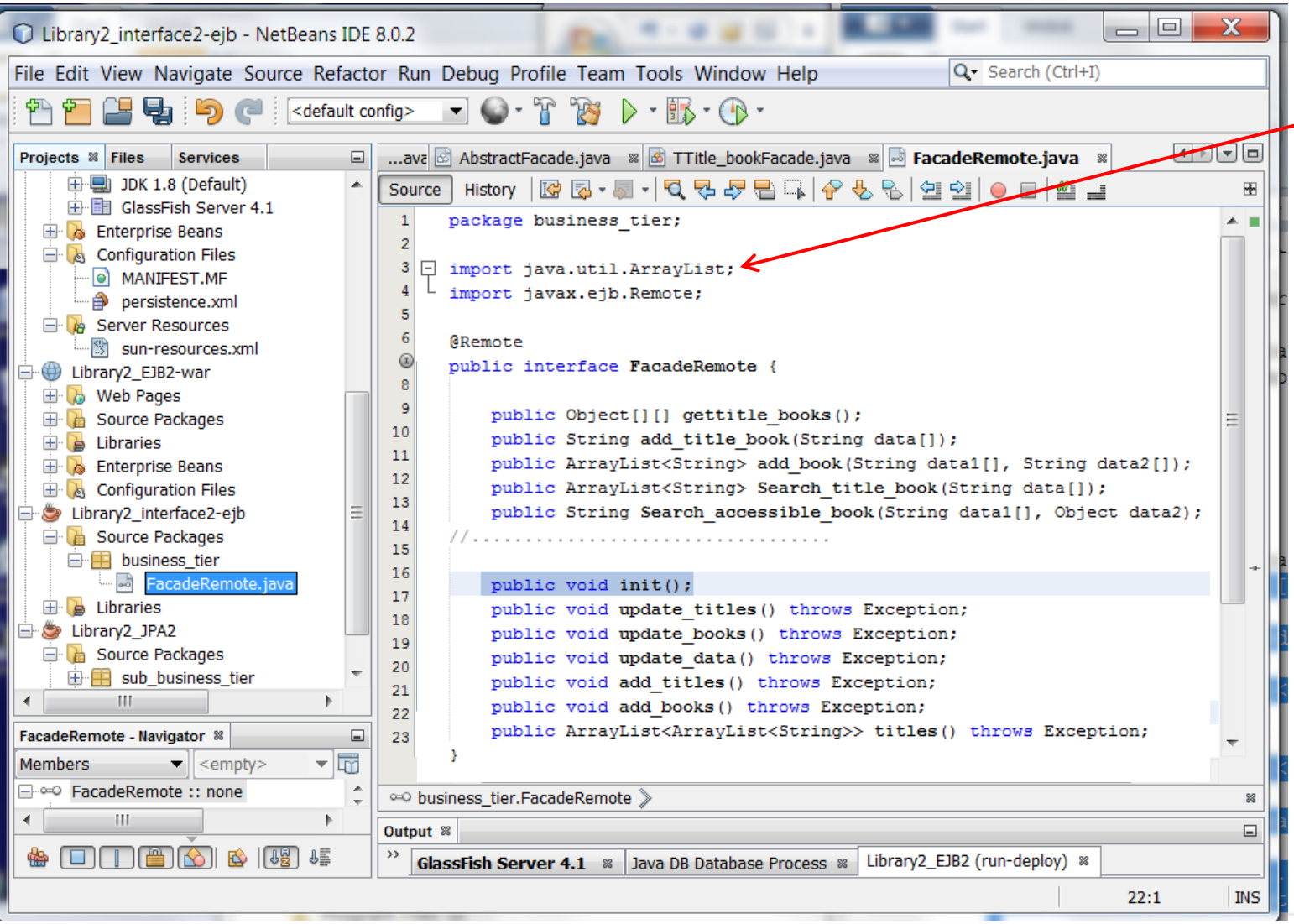


4.8.6. The result



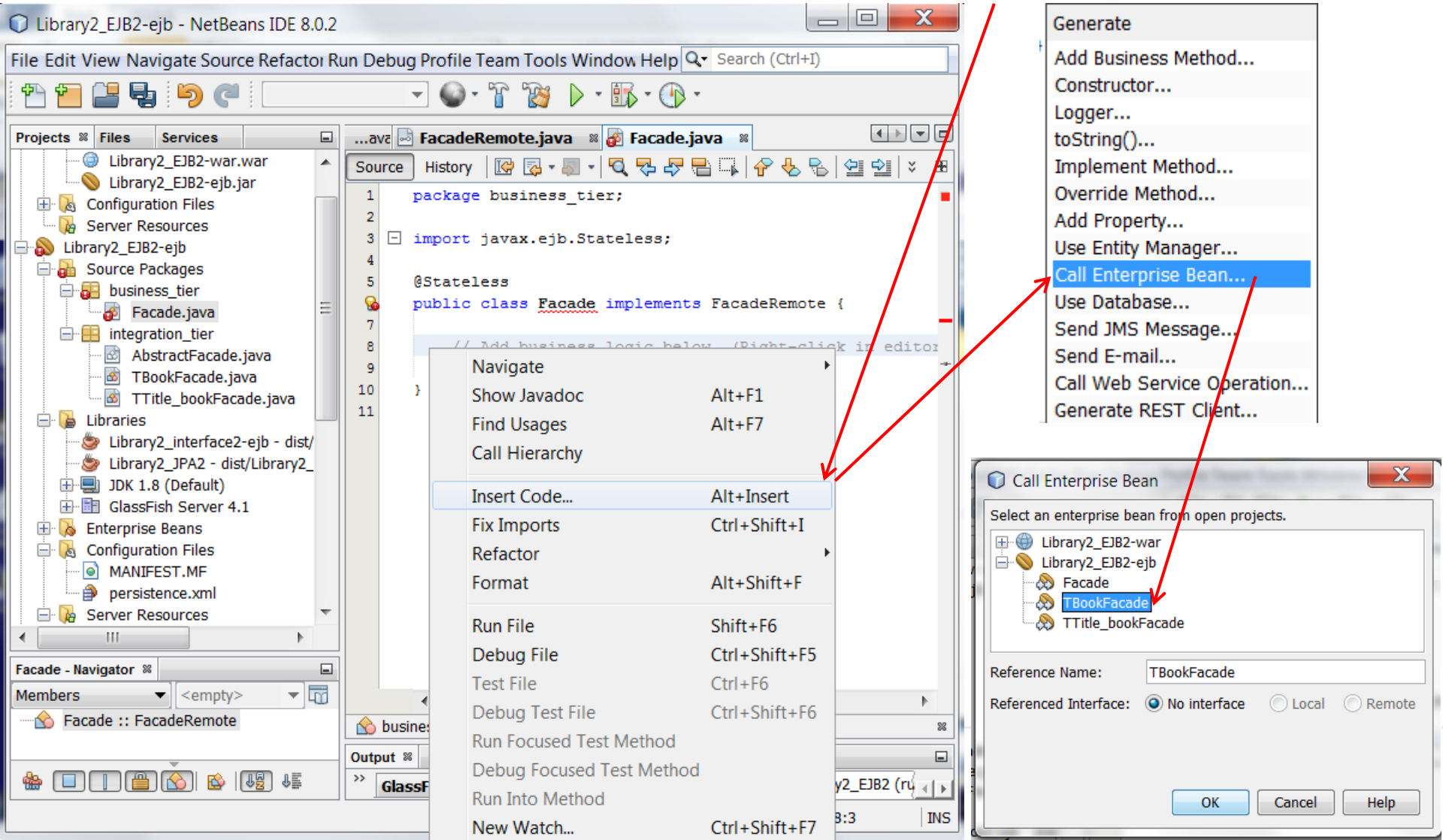
4.9./ 4.9.1 . Update code of the FacadeRemote interface from the project of lab3 – for accessing the database by using the TTitle_bookFacade and TBookFacade

```
package business_tier;

import javax.ejb.Remote;

@Remote
public interface FacadeRemote {
    public Object[][] gettitle_books();

    public String add_title_book(String data[]);

    public ArrayList<String> add_book(String data1[], String data2[]);

    public ArrayList<String> Search_title_book(String data[]);

    public String Search_accessible_book(String data1[], Object data2);
//.....................................
    public void init();

    public void update_titles() throws Exception;

    public void update_books() throws Exception;

    public void update_data() throws Exception;

    public void add_titles() throws Exception;

    public void add_books() throws Exception;

    public ArrayList<ArrayList<String>> titles() throws Exception;
}
```

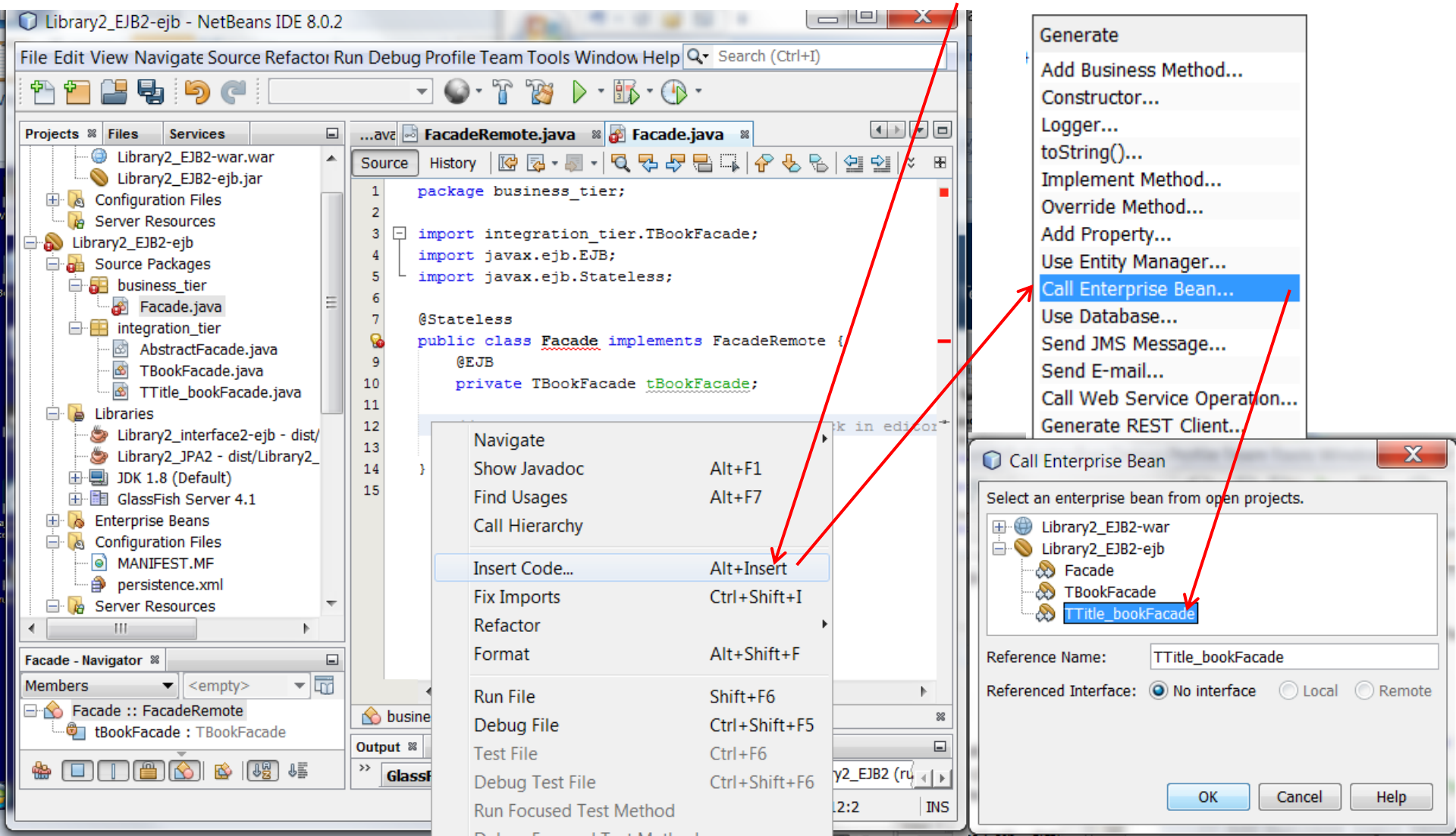


4.9.2. After Fix Imports.. process - right click and select the Fix Imports.. item. In the form of Fix Imports... you must select the ArrayList import.

4.10./ 4.10.1. Create the update of code of the Facade Session Bean - right click the code editor and select the Insert Code… item. In the Insert Code form select the Call Enterprise Bean... item and in its form select the TBookFacade, which is the local Enterprise Bean for persisting the TBook and TBook_period entities.



4.10.2. Create the update of code of the Facade Session Bean - right click the code editor and select the Insert Code… item. In the Insert Code… form select the Call Enterprise Bean... item and in its form select the TTitle_bookFacade, which is the local Enterprise Bean for persisting the TTitle_book and TTitle_book_on_tape entities.



4.10.3. Create the update of code of the Facade class in the Library2_JPA2 project from sub_business_tier – the update_data method for intialization the content of the mTTitle_books collection and the content of each mBooks collection of each instance of the TTitle_book class by using data from database

```
public synchronized void update_data(TTitle_book titles[], TBook books[]) {
    mTitle_books.clear();
    for (TTitle_book t : titles) {
        mTitle_books.add(t);
    }
    for (TTitle_book title : mTitle_books) {
        for (TBook book : books) {
            TTitle_book title1 = book.getmTitle_book();
            if (title1 != null) {
                if (title1.equals(title)) {
                    title.getmBooks().add(book);
                }
            }
        }
    }
}
```

4.10.4. The same part of code of the Facade remote session bean from the project of lab3

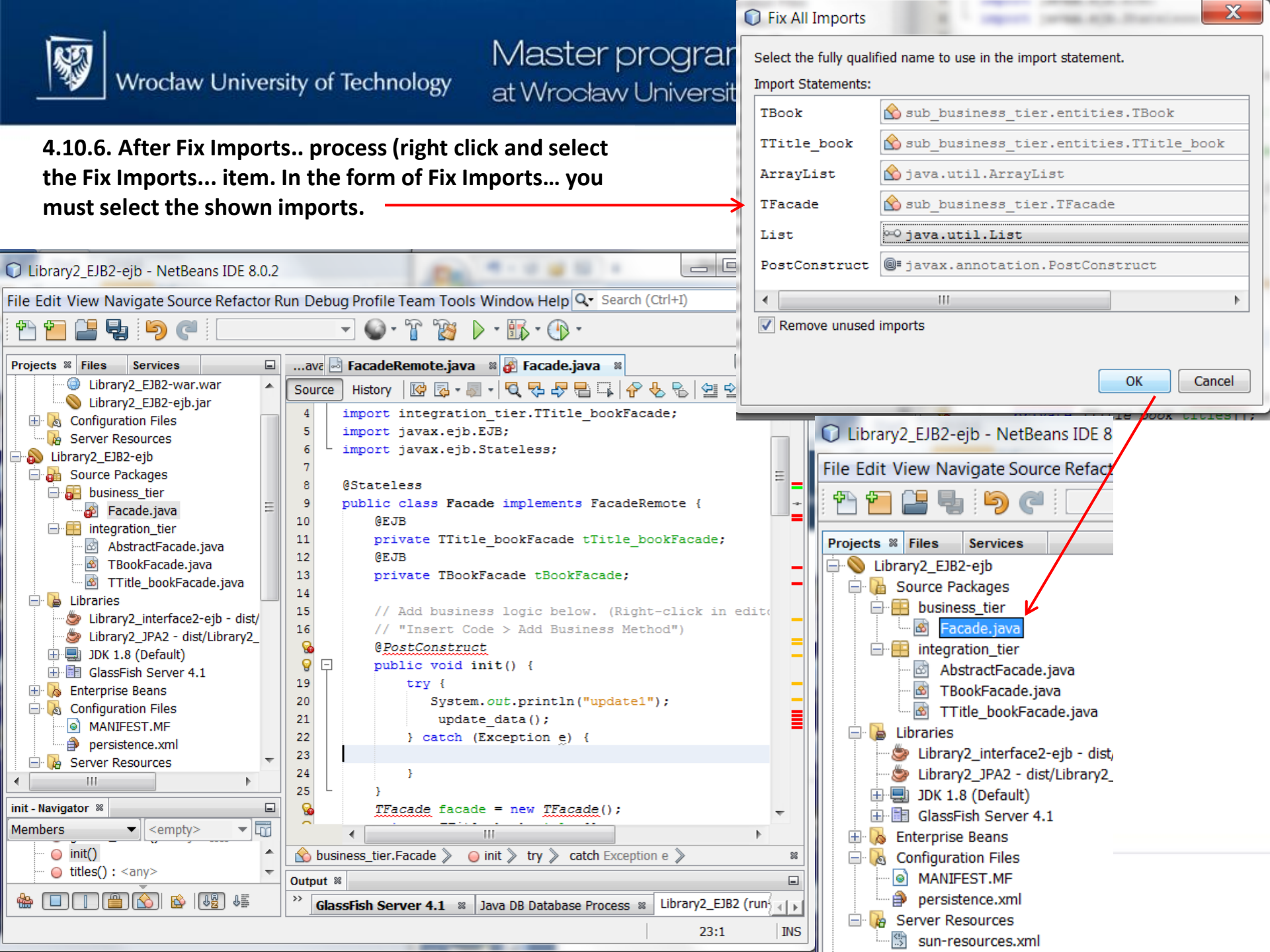
```
package business_tier;

import integration_tier.TBookFacade;
import integration_tier.TTitle_bookFacade;
import java.util.ArrayList;
import java.util.List;
import javax.annotation.PostConstruct;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import sub_business_tier.TFacade;
import sub_business_tier.entities.TBook;
import sub_business_tier.entities.TTitle_book;

@Stateless
public class Facade implements FacadeRemote {
    @EJB
    private TTitle_bookFacade tTitle_bookFacade;
    @EJB
    private TBookFacade tBookFacade;
    private TTitle_book titles[];
    private TBook books[];

   TFacade facade = new TFacade();

   @PostConstruct
   @Override
    public void init() {
        try {
            update_data();
        } catch (Exception e)    {    }
    }
    @Override
    public Object[][] gettitle_books()                                  {       return facade.gettitle_books();    }
    @Override
    public String add_title_book(String data[])                         {       return facade.add_title_book(data);    }
    @Override
    public ArrayList<String> add_book(String data1[], String data2[])   {       return facade.add_book(data1, data2);    }
   @Override
    public ArrayList<String> Search_title_book(String data[])           {       return facade.Search_title_book(data);    }
    @Override
    public String Search_accessible_book(String data1[], Object data2)  { return facade.Search_accessible_book(data1, data2);    }
```

The new method for initial activities od Session Bean

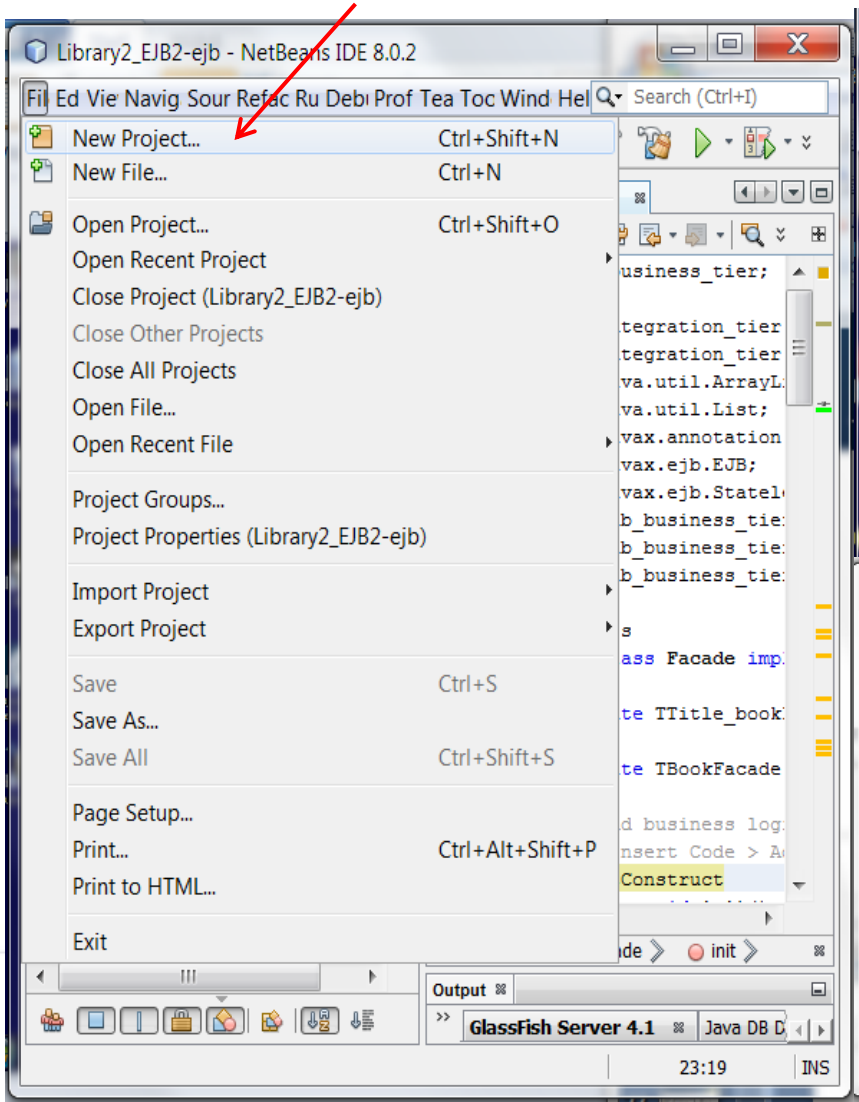
4.10.5. Update code of the Facade remote session bean from the project of lab3 – for accessing the database by using the TTitle_bookFacade and TBookFacade

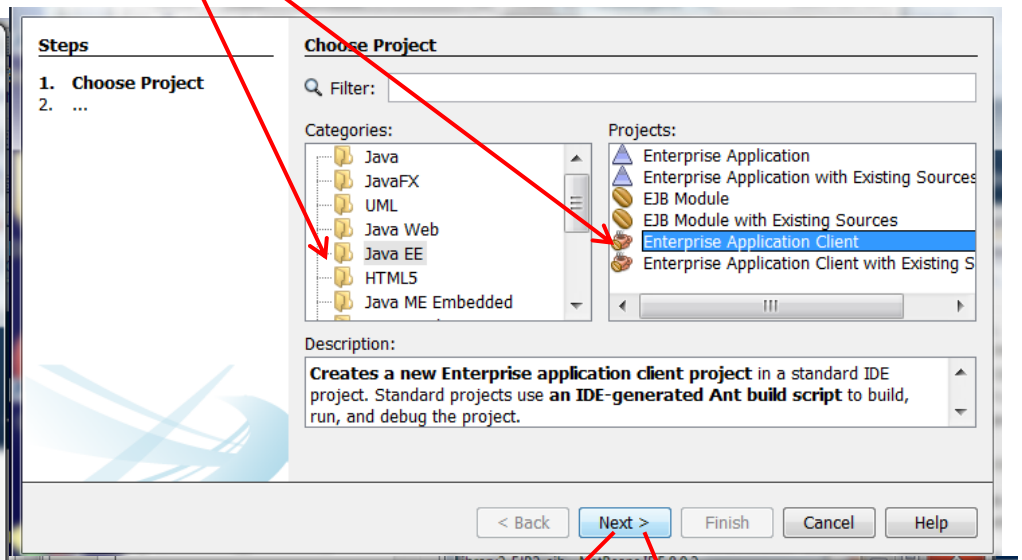
```
//.....
  public void update_titles() throws Exception          {  titles = tTitle_bookFacade.getTTitle_books_();  }

  public void update_books() throws Exception          {  books = tBookFacade.getTBooks_();    }

  public void update_data() throws Exception {
     update_titles();
     update_books();
     facade.update_data(titles, books);  }

  public void add_titles() throws Exception   {  tTitle_bookFacade.addTTitle_books(facade.getmTitle_books());
}

  public void add_books() throws Exception {  tBookFacade.addTBooks(facade.getmTitle_books());   }

 public ArrayList<ArrayList<String>> titles() throws Exception {
     List<TTitle_book> help1 = tTitle_bookFacade.findAll();
     ArrayList<ArrayList<String>> help2;
     help2 = new ArrayList();
     for (TTitle_book t : help1) {
        ArrayList<String> help3 = new ArrayList();
        help3.add(t.getPublisher());
        help3.add(t.getISBN());
        help3.add(t.getTitle());
        help3.add(t.getAuthor());
        help3.add(t.getActor());
        help2.add(help3);
     }
     return help2;
  }
```

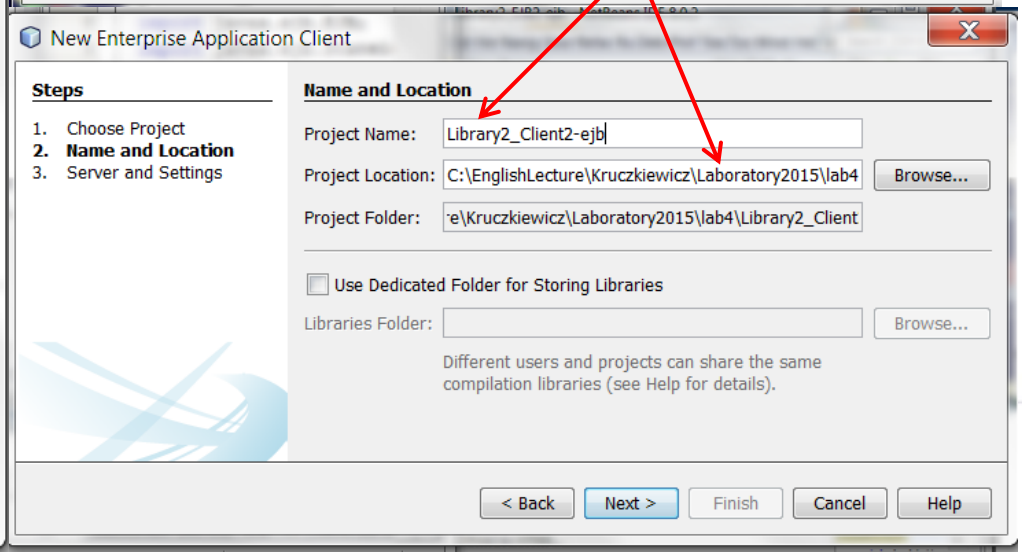
4.10.6. After Fix Imports.. process (right click and select the Fix Imports... item. In the form of Fix Imports… you must select the shown imports.



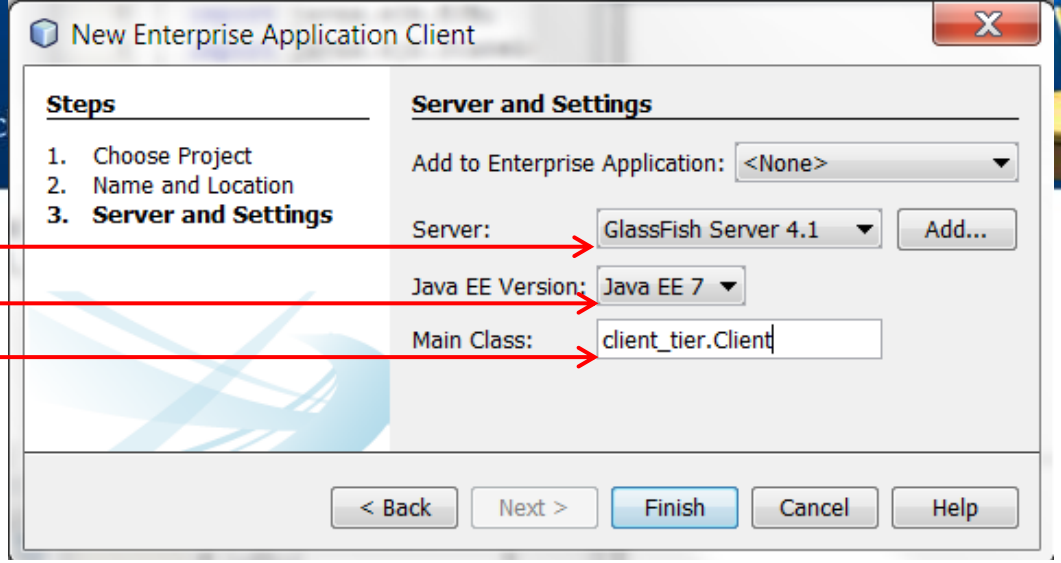
5. /5.1.1. Create the new Enterprise Application Client (the based on code of the Library1_client1-ejb project of lab3) – click File/New Project…/Java EE/ Enterprise Application Client; fill name of project as Library2_Client2-ejb and select the location of the project (the same as the location of other projects)



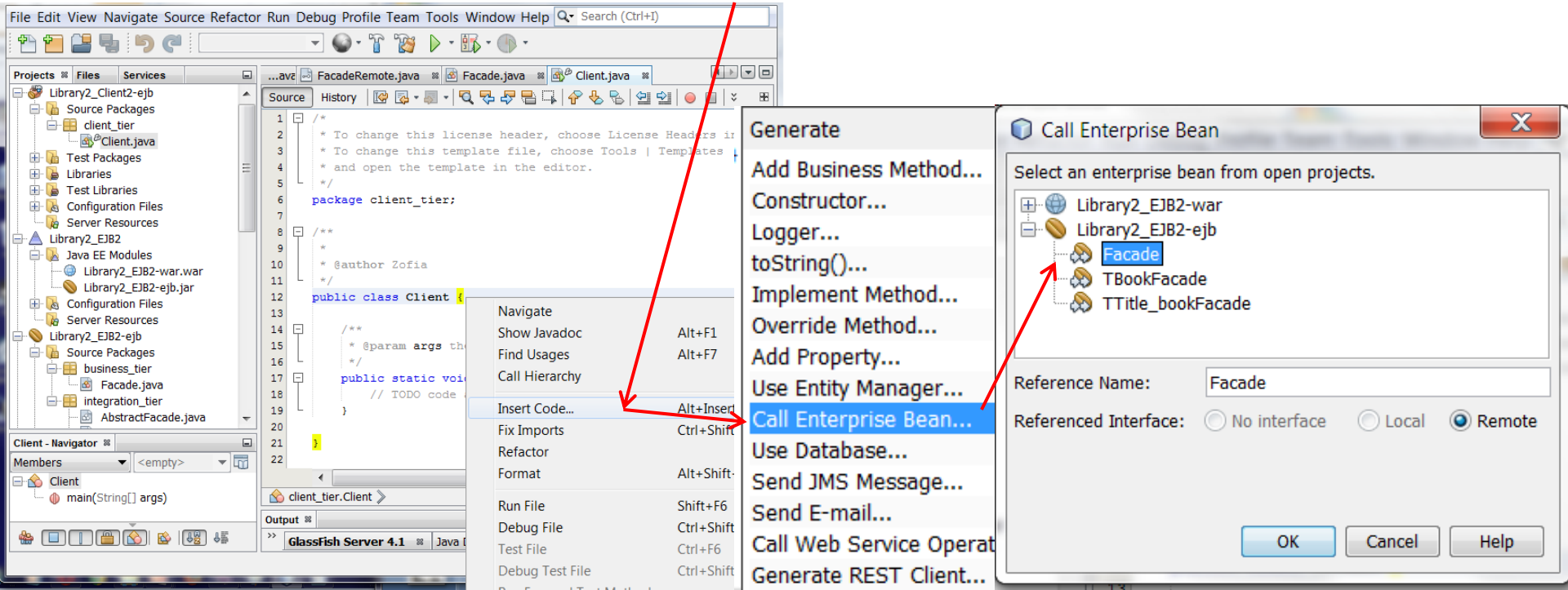




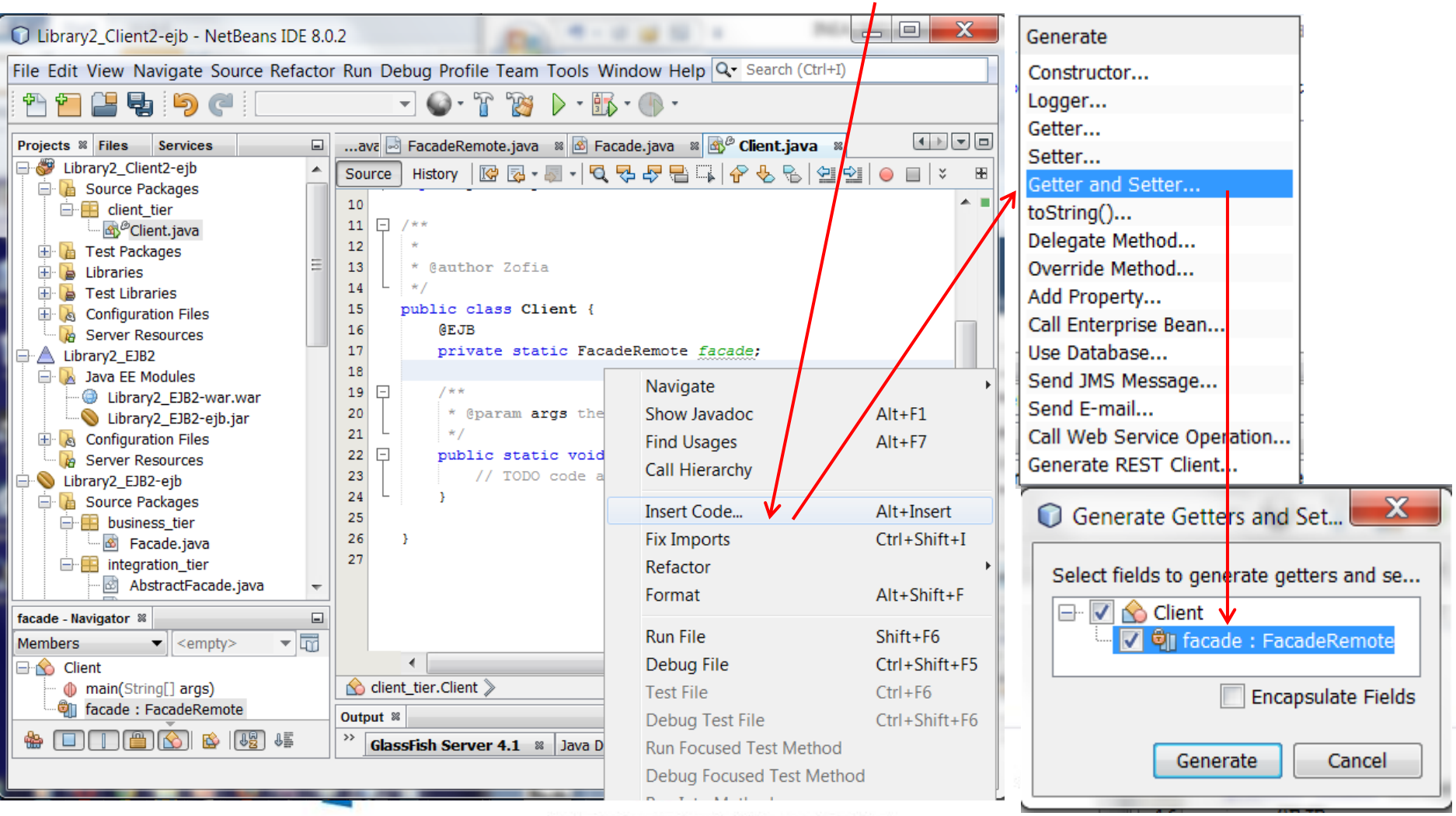
5.1.2. Continuation – select the application server GlassFish 4.1, Java EE 7 platform and fill the name of main class – the same package and name of class as in the SE client project.



5.1.3. Create code of the Client class - right click the code editor and select the Insert Code item. In the InsertCode form select the Call Enterprise Bean.. item and in its form select the Facade, which is the remote Enterprise Bean from the Library2_EJB2-ejb module



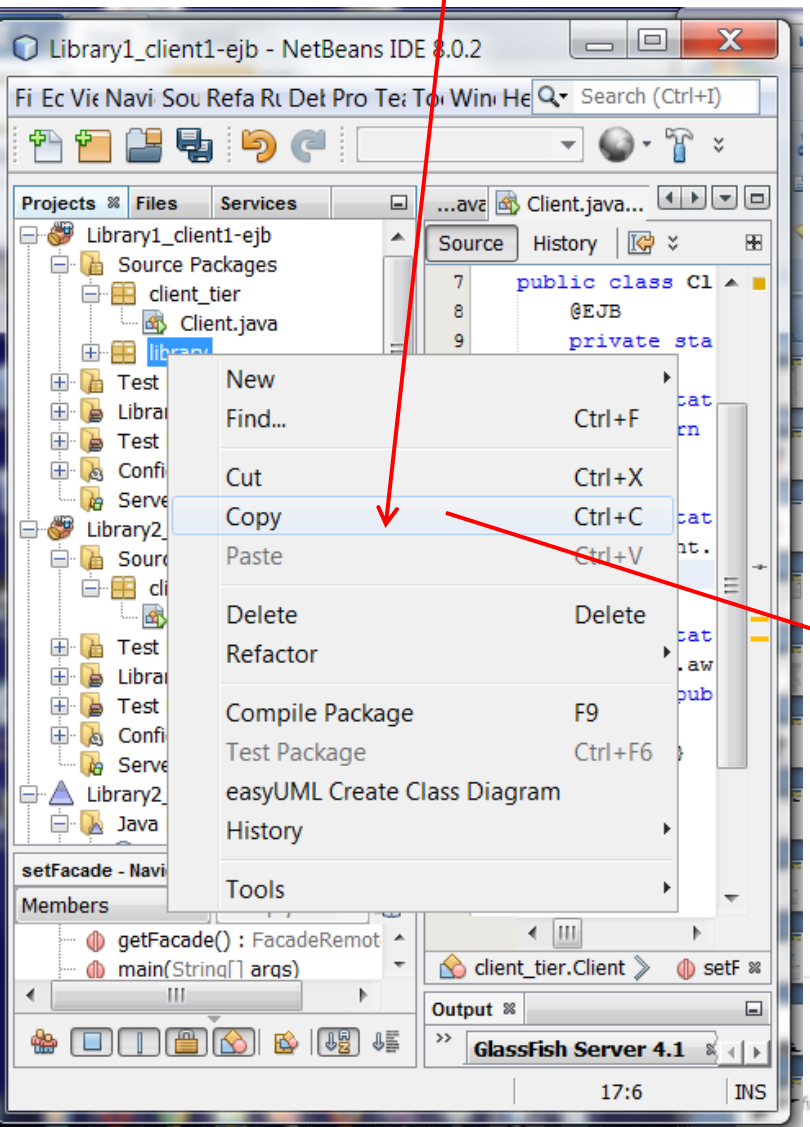
5.1.4. Create the getter and setter methods for the reference of the facade session bean - right click the code editor and select the Insert Code… item. In the Insert Code form select the Getter and Setter... item and in its form select the facade attribute of theClient class. Click Generate.

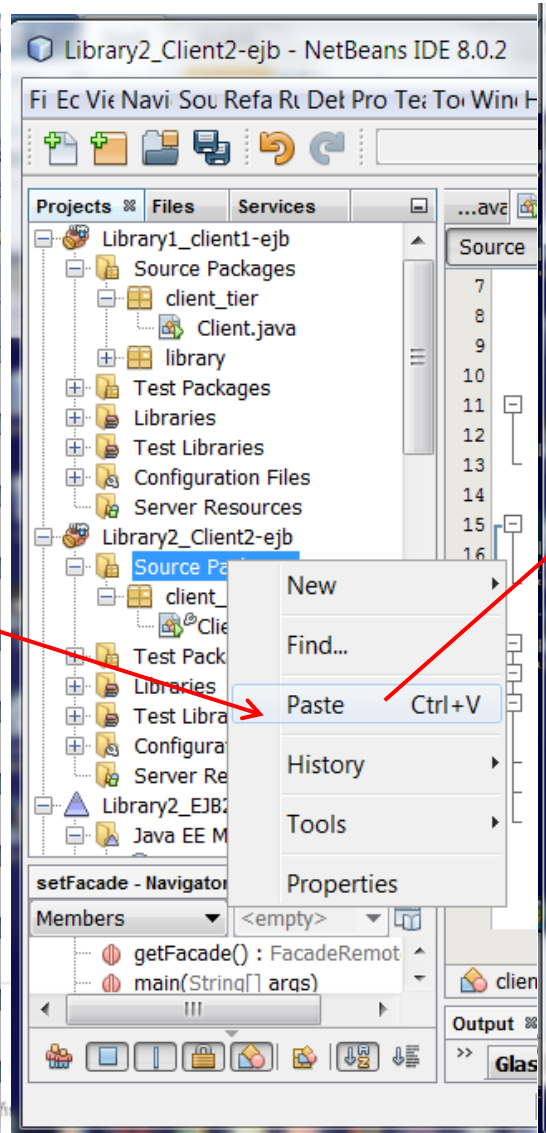


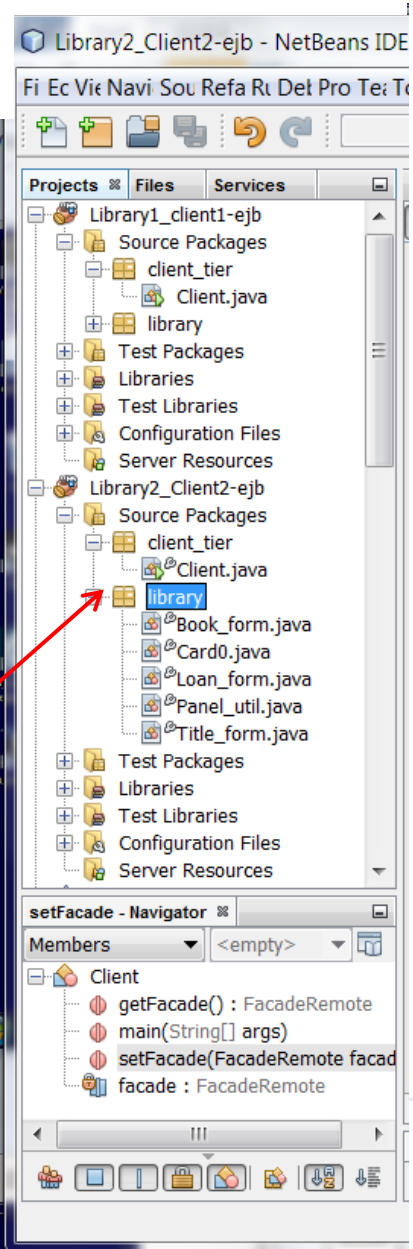
5.1.5. Create the copy of the library package from the Library1_client1-ejb project (lab3) – right click tle library package item of this project and select the Copy item

5.1.6. Right click the Source Packages of the Library2_Client2-ejb project and select the Paste item

5.1.7. The result



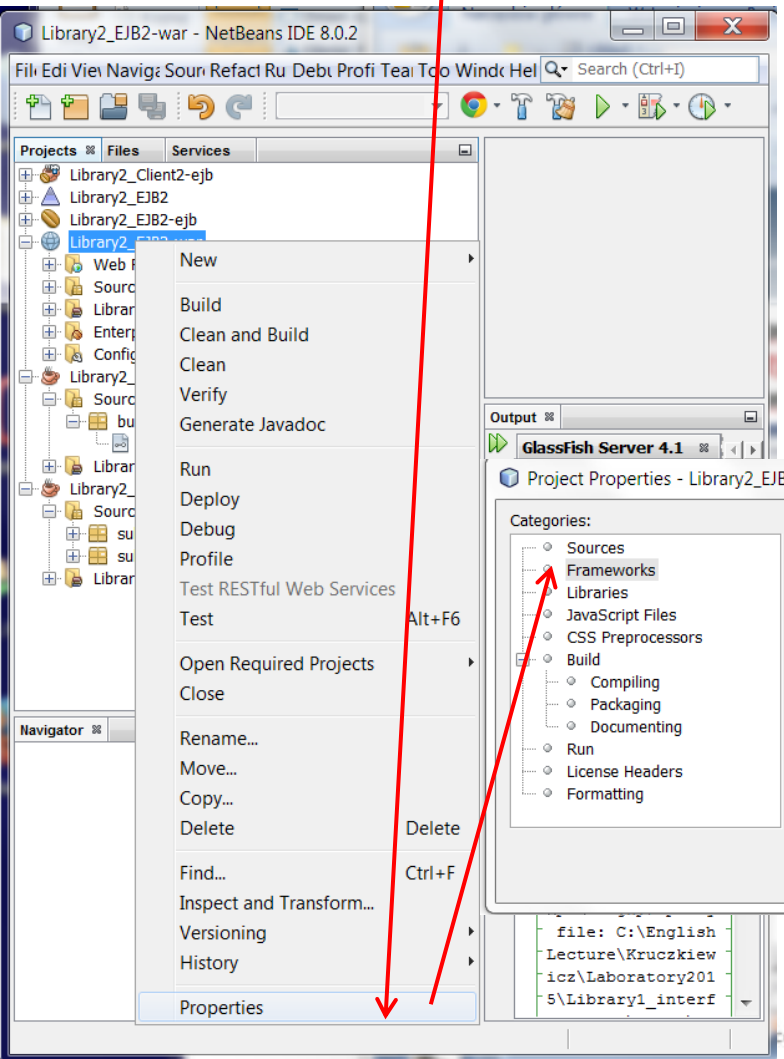


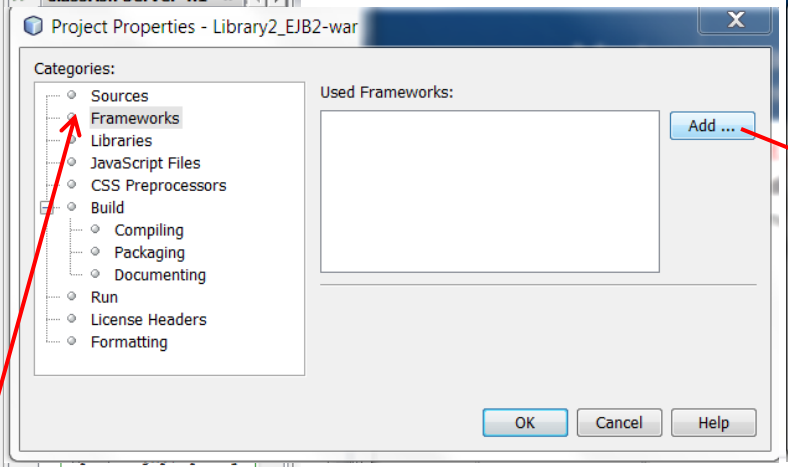


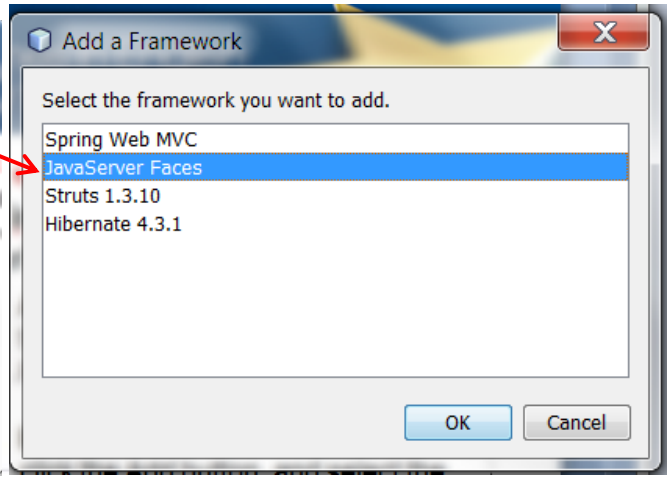
```
package client_tier;

import business_tier.FacadeRemote;
import javax.ejb.EJB;
import library.Panel_util;

public class Client {
   @EJB
   private static FacadeRemote facade;

   public static FacadeRemote getFacade()
         { return facade; }

   public static void setFacade(FacadeRemote facade)
        { Client.facade = facade; }

   public static void main(String[] args) {
     java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            Panel_util.createAndShowGUI();
         }
      });
   }
}
```

5.1.8. Code of the Client class Library2_Client2-ejb project – the same as the code of lab3 client project

6. /6.1. Definition of the Web Client Tier (Library2_EJB2-war, prepared during creation of **Library2_EJB2)** project**) for inserting application data into the database and displaying the data from database** – change the Web Framework to the JavaServer Faces type



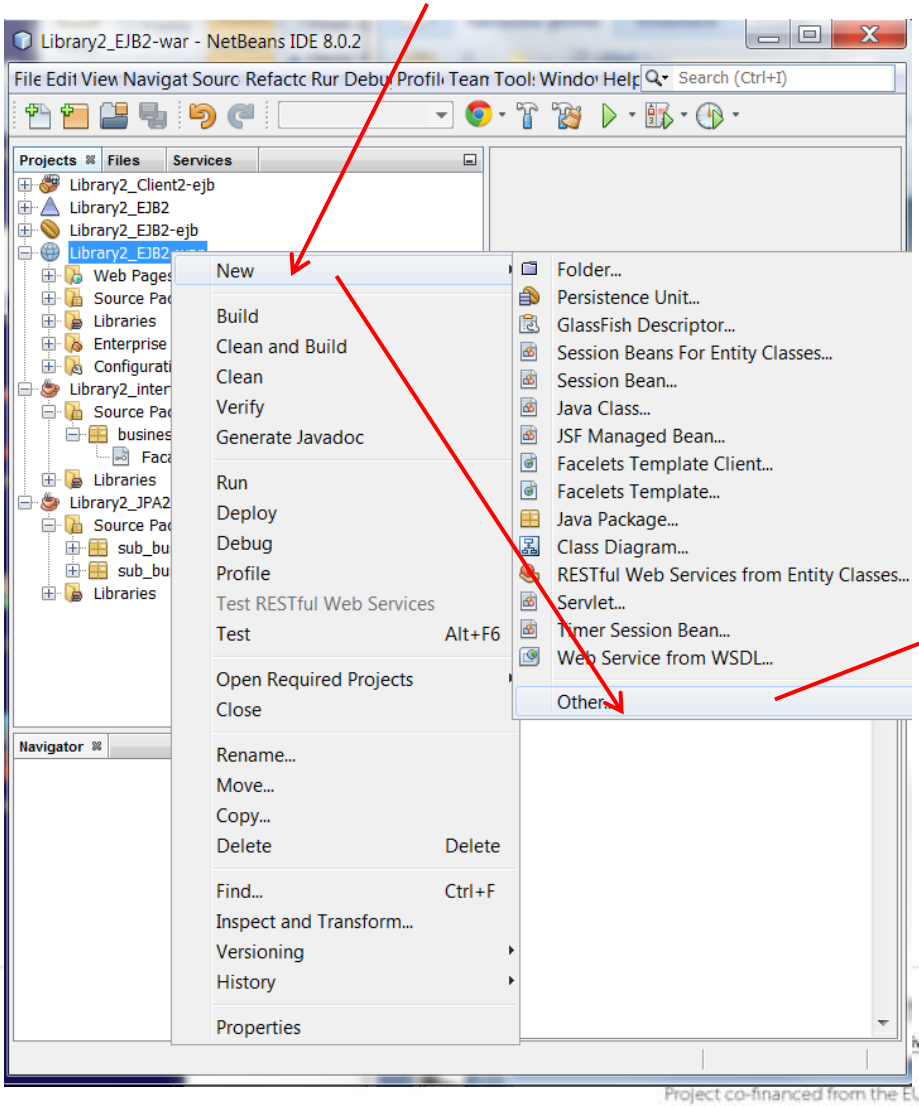


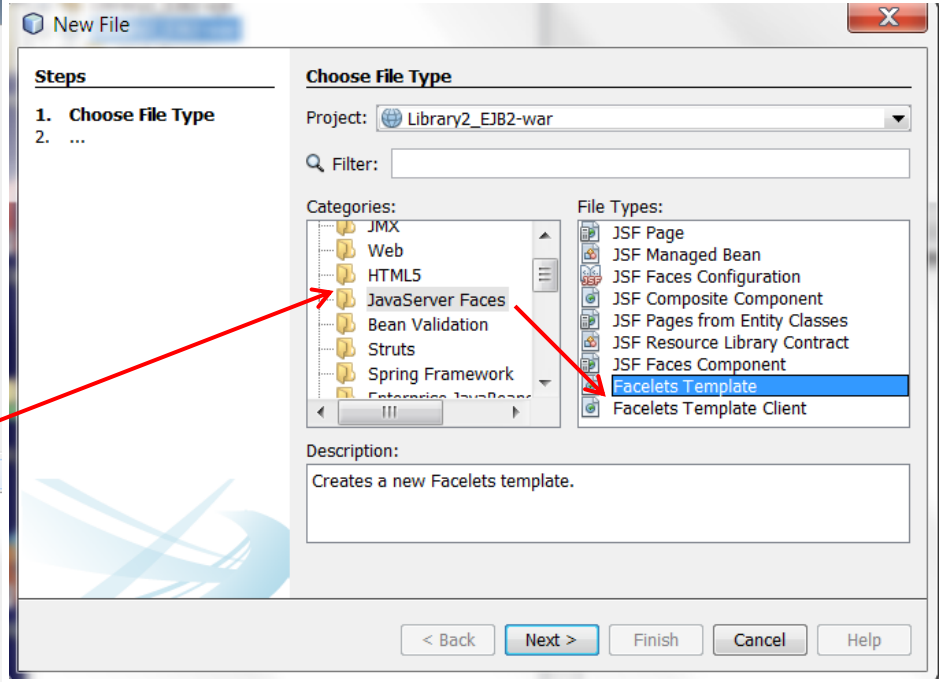


At left: Right click the name of project in the Projects tab, and choose the Properties item

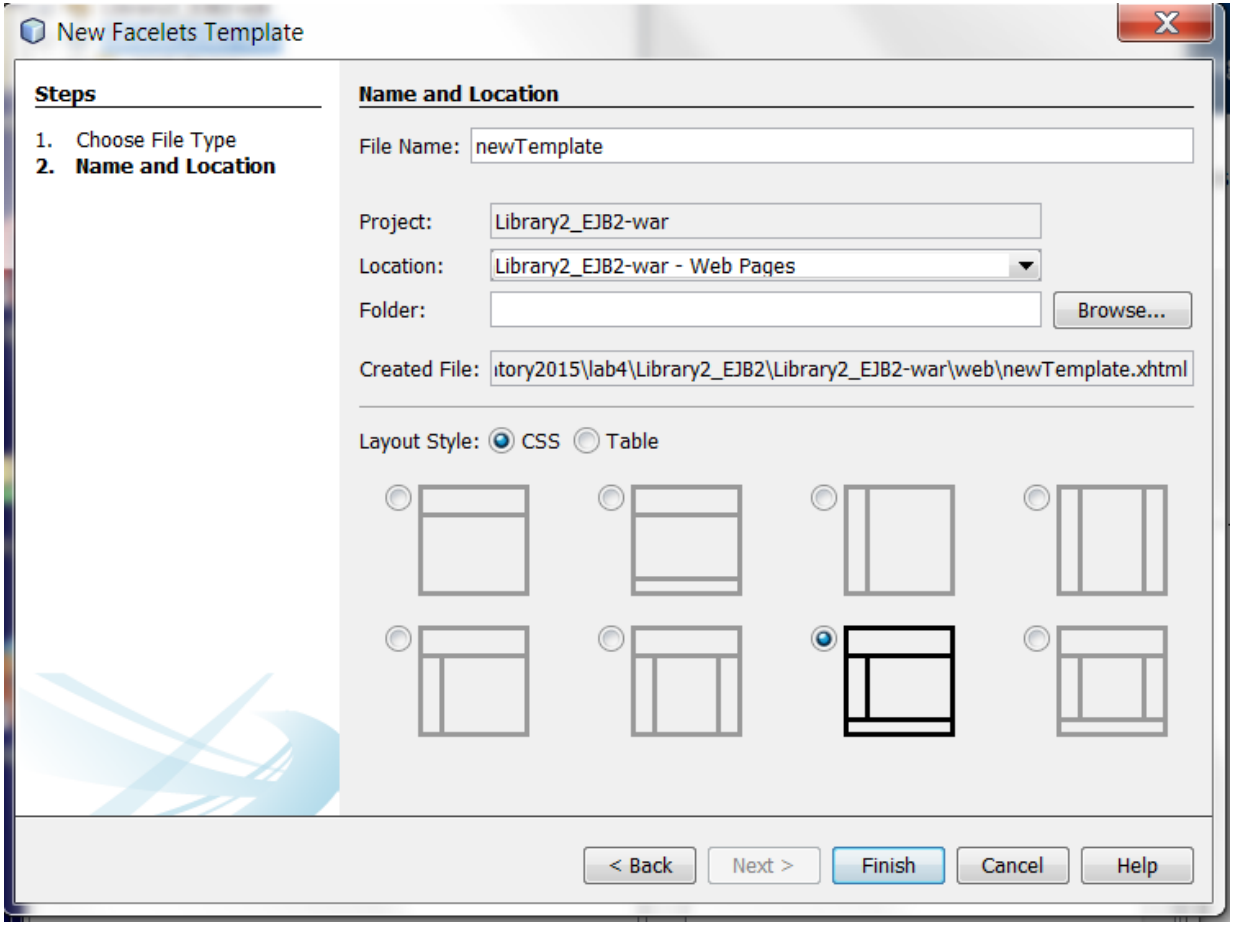
Below: Choose the Framework item, click the Add button, and select the JavaServer Faces framework

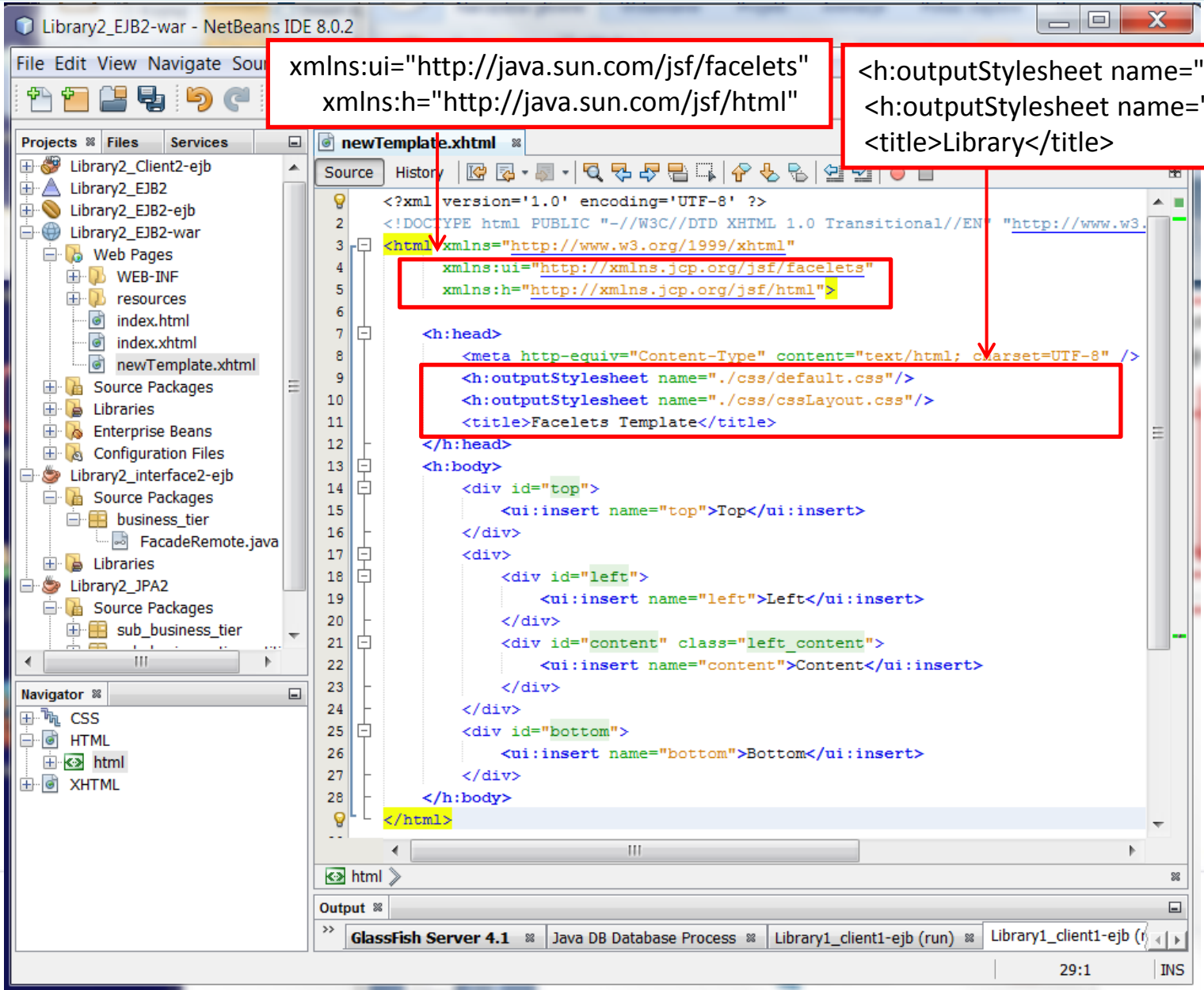
6.2. /6.2.1. Addition the template of web pages: Right click the name of project in the Projects tab, choose the New/Other items, then select JavaServer Faces File type, and the the Facelets Template file. Click Next.





6.2.2. Below: choose the template and set the name of template file. Click Finish.



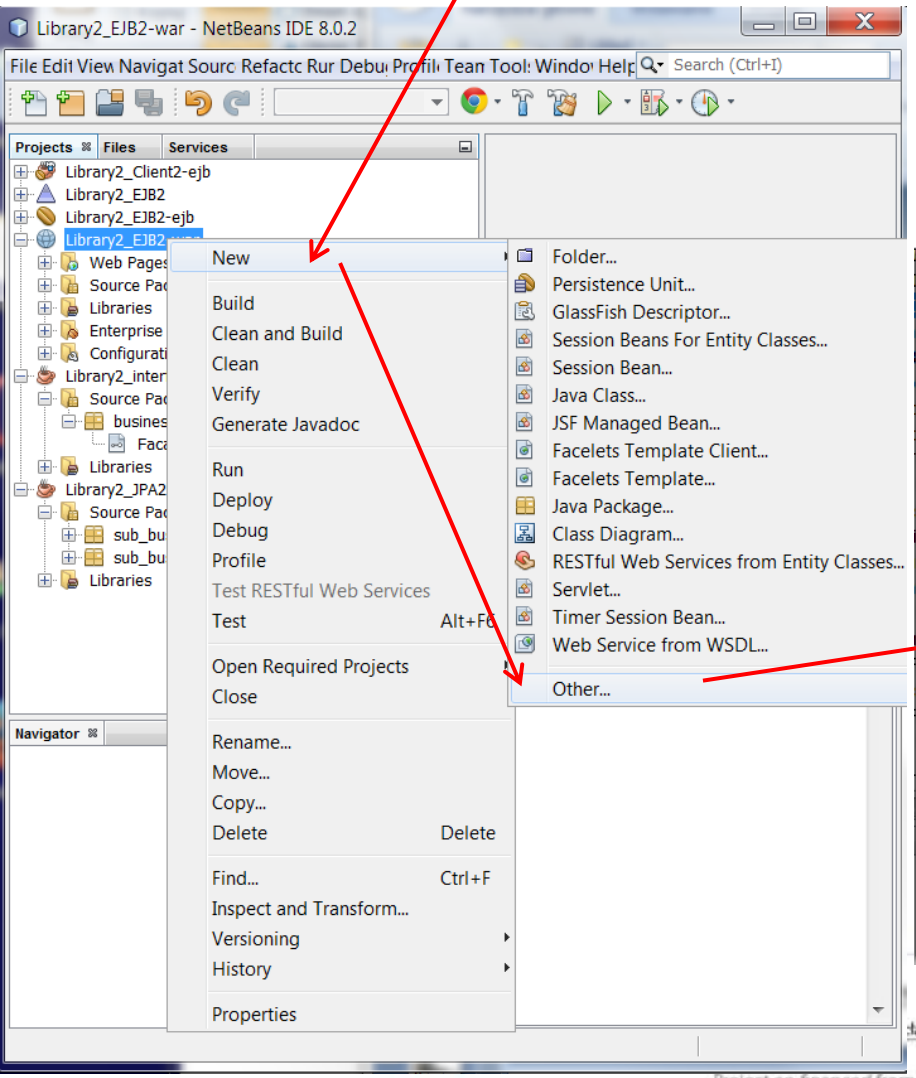


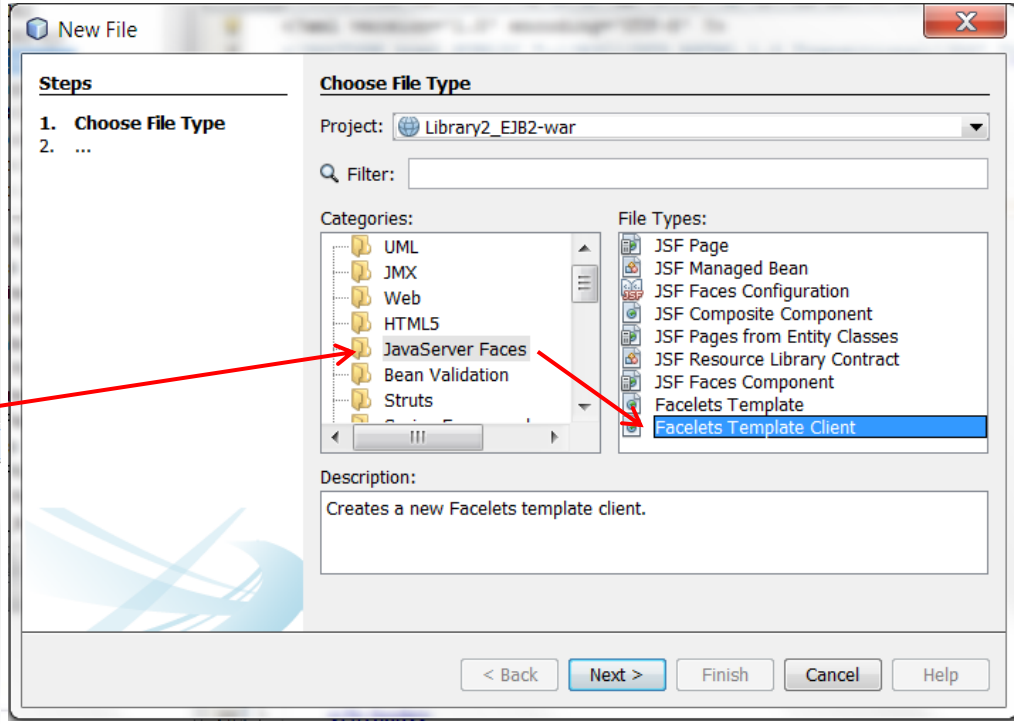
6.3/6.3.1. Addition the template of web pages: Change the the content of attributes of the **h:head** tag and html tag

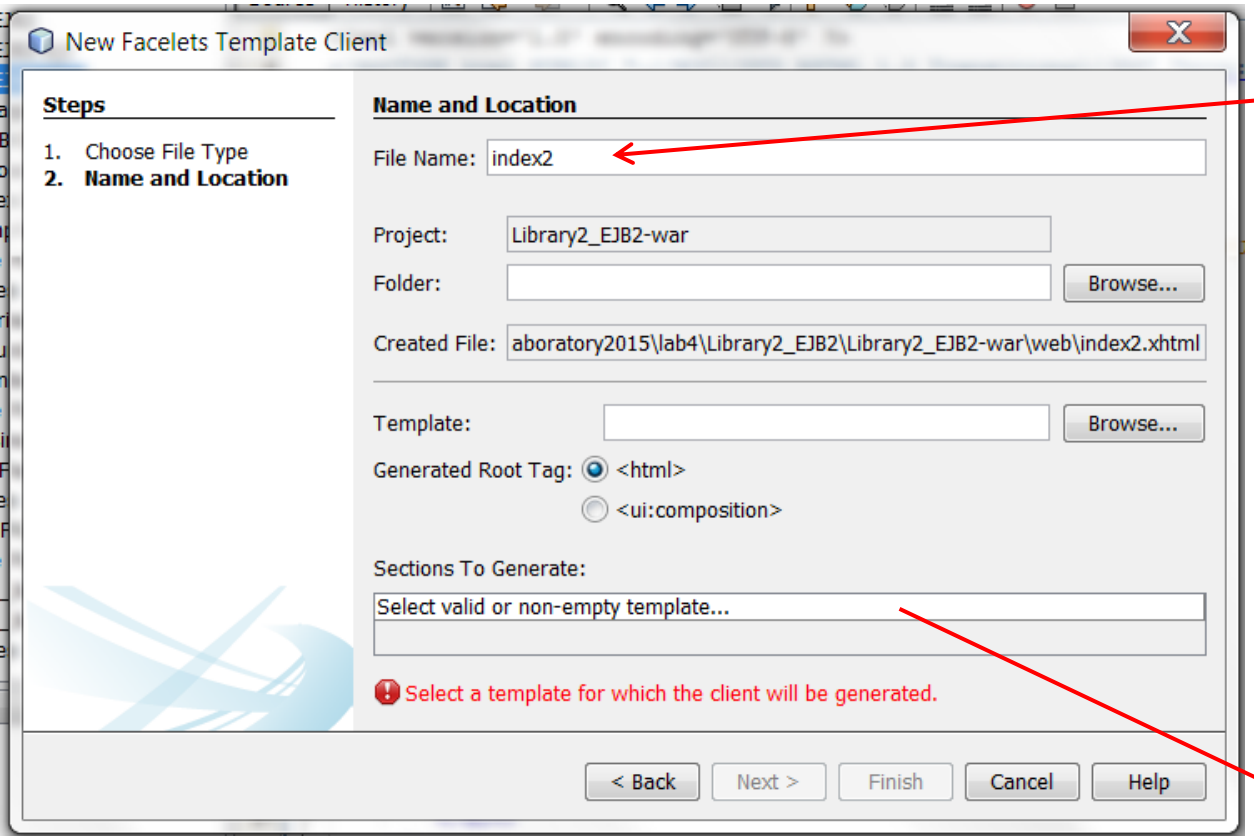
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html">

  <h:head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <h:outputStylesheet name="css/default.css" />
    <h:outputStylesheet name="css/cssLayout.css"/>
    <title>Library</title>
  </h:head>

  <h:body>
    <div id="top">
      <ui:insert name="top">Top</ui:insert>
    </div>
    <div>
      <div id="left">
        <h:link outcome="/faces/presentation_tier_view/Store_data" value="Store data"/><br/>
        <h:link outcome="/faces/presentation_tier_view/Show_data" value="Show data"/>
      </div>
      <div id="content" class="left_content">
        <ui:insert name="content">Content</ui:insert>
      </div>
    </div>
    <div id="bottom">
      <ui:insert name="bottom">Bottom</ui:insert>
    </div>
  </h:body>
</html>
```

6.3.2. Addition the template of web pages:
Insert the new code into the div tag with id equals left – this will be a menu of all web pages based on this template

6.4./ 6.4.1. Creation the main JSF page (index2), based on the previous defined template – right click the Library2_EJB2-war item and select the New/Other… items. In the New File dialog choose the JavaServer Faces/Facelets Template Client options. Click Next.

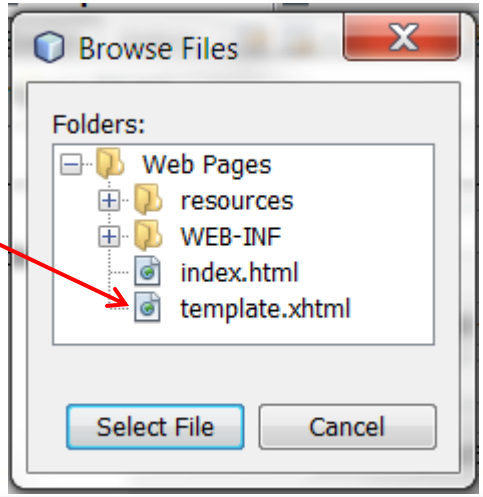






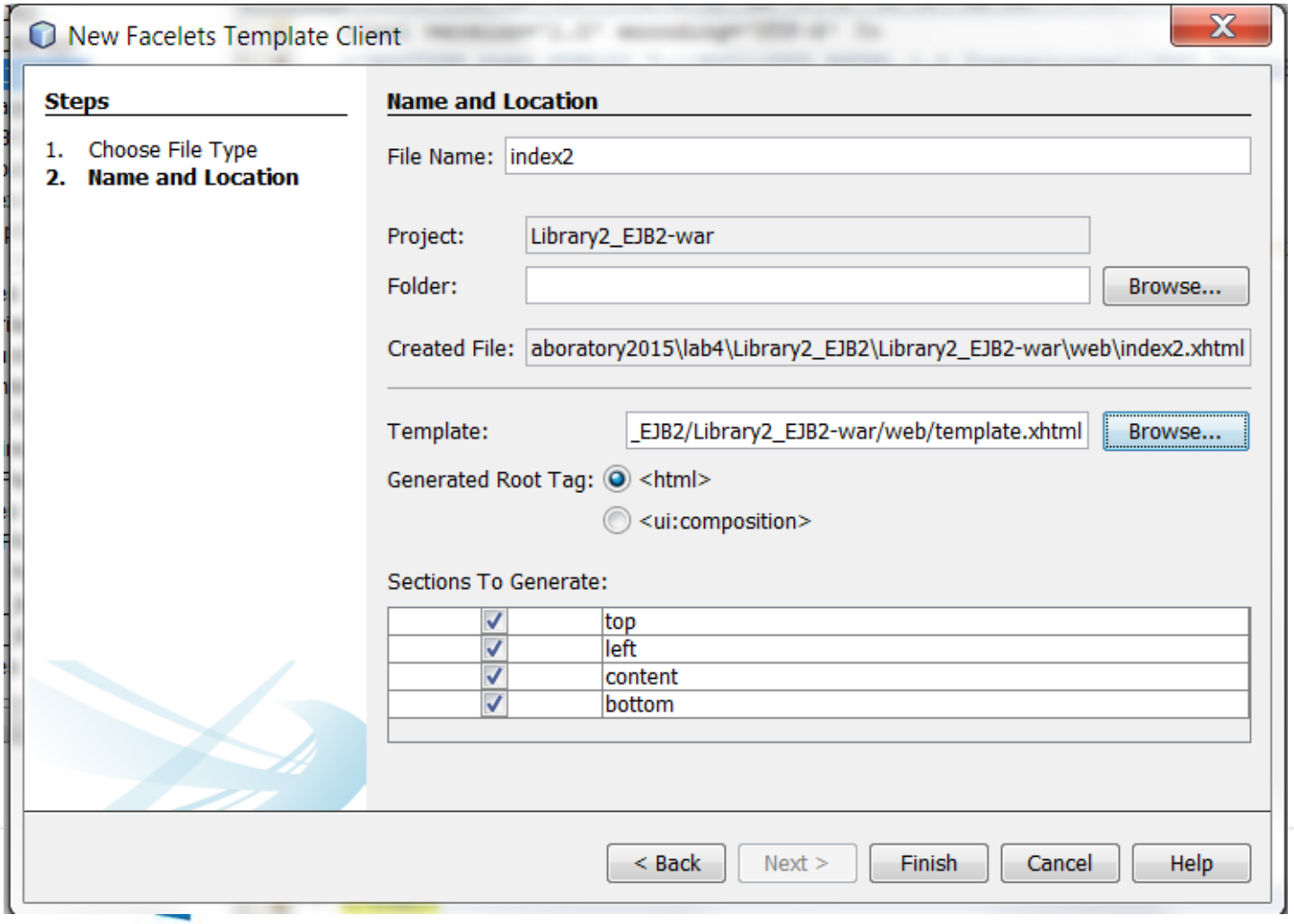
6.4.2. Creation the main JSF page (index2), based on the previous defined template – continuation

Below: choose the proper template – click Select File in the Browse Files dialog

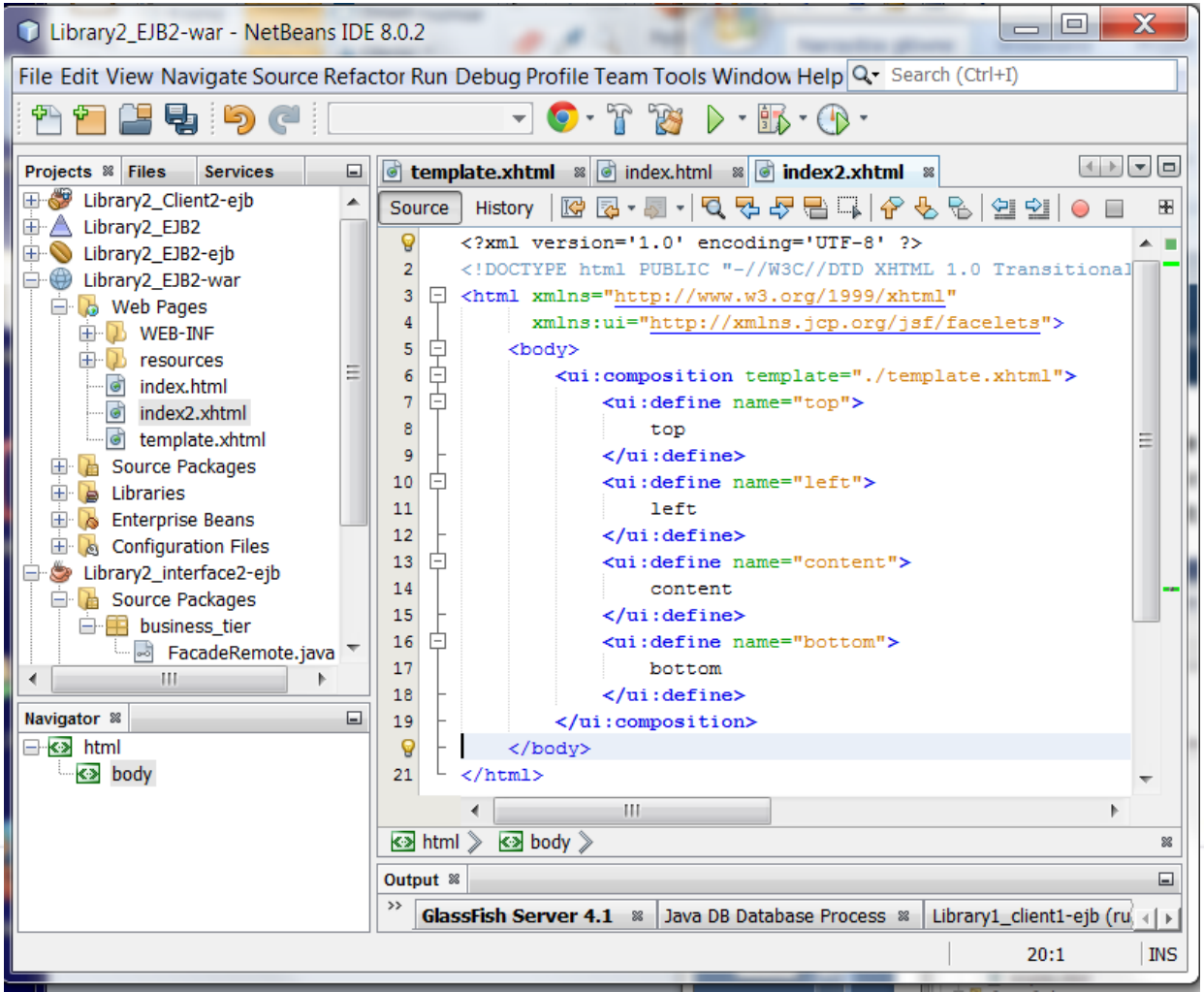


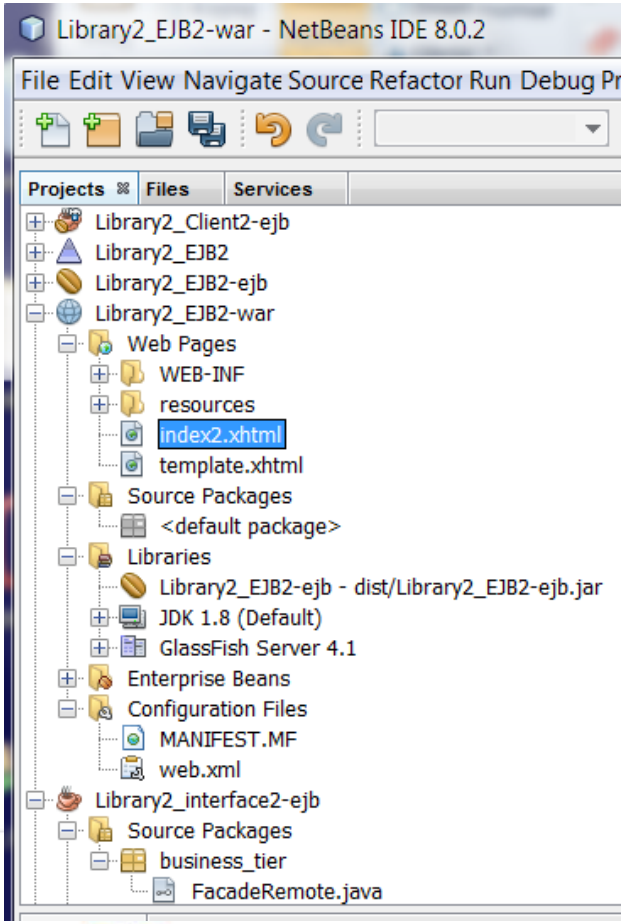
6.4.3. Creation the main JSF page (index2), based on the previous defined template – continuation

Below: after choosing the proper template



6.4.4. Creation the main JSF page (index2), based on the previous defined template – the result
At left: the view of the unused JSF anf JSP main pages (index.jsp and index.xhtm);
At right: after deleting unused pages - the index2.xhtml is the main JSF page

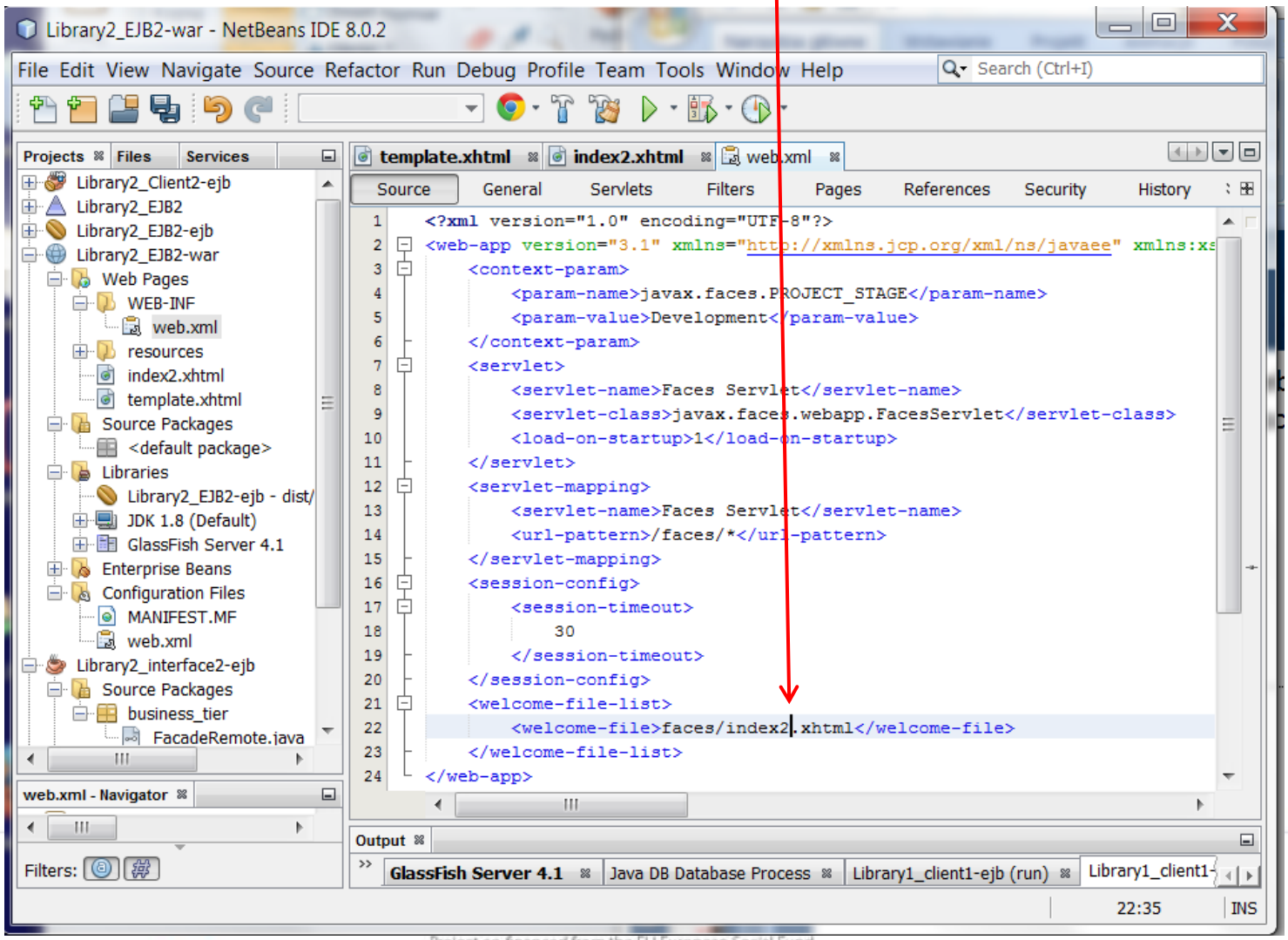


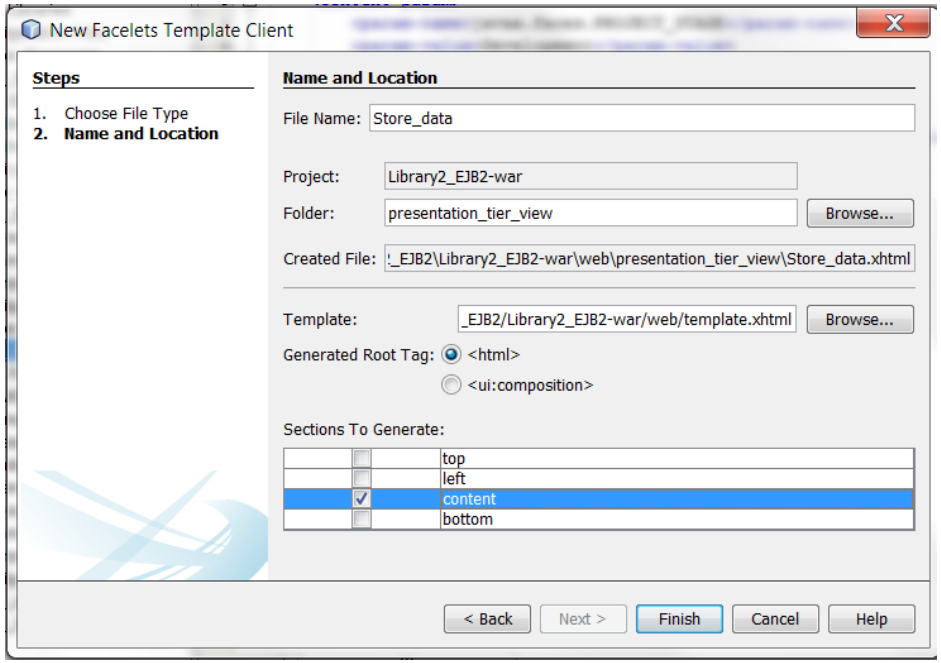


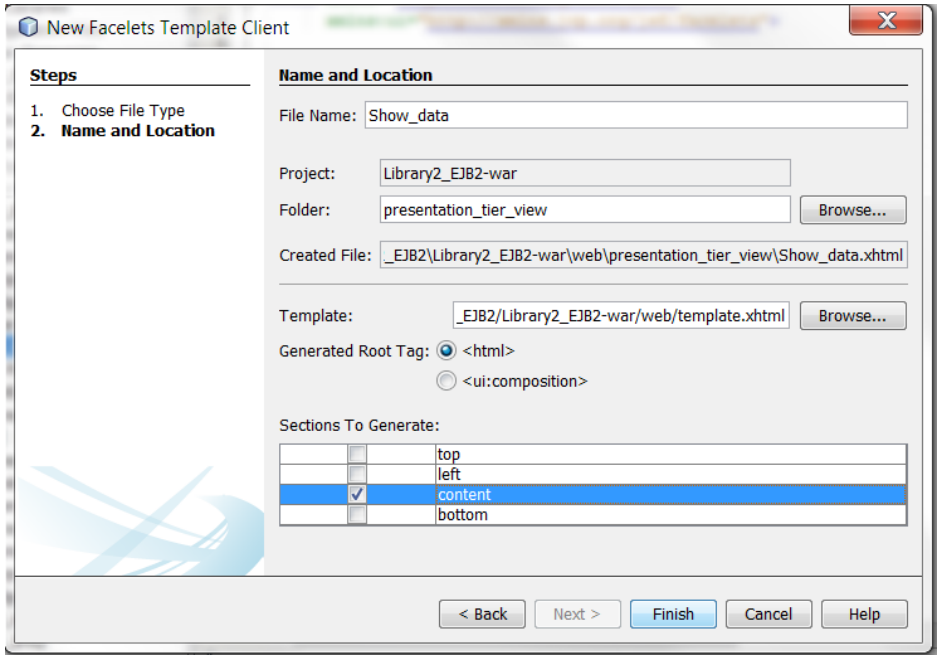
6.4.5. Add the h:link tags in the left part of template

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html">

<body>
       <ui:composition template="./template.xhtml">
          <ui:define name="top">
             top
          </ui:define>
          <ui:define name="left">
             <h:link outcome="/faces/presentation_tier_view/Store_data" value="Store data"/><br/>
             <h:link outcome="/faces/presentation_tier_view/Show_data" value="Show data"/>
          </ui:define>
          <ui:define name="content">
             content
          </ui:define>
          <ui:define name="bottom">
             bottom
          </ui:define>
       </ui:composition>
    </body>
</html>
```

6.4.6. Set up the main web page as the index2.xtml – in the web.xml file (the descriptor of the web module).

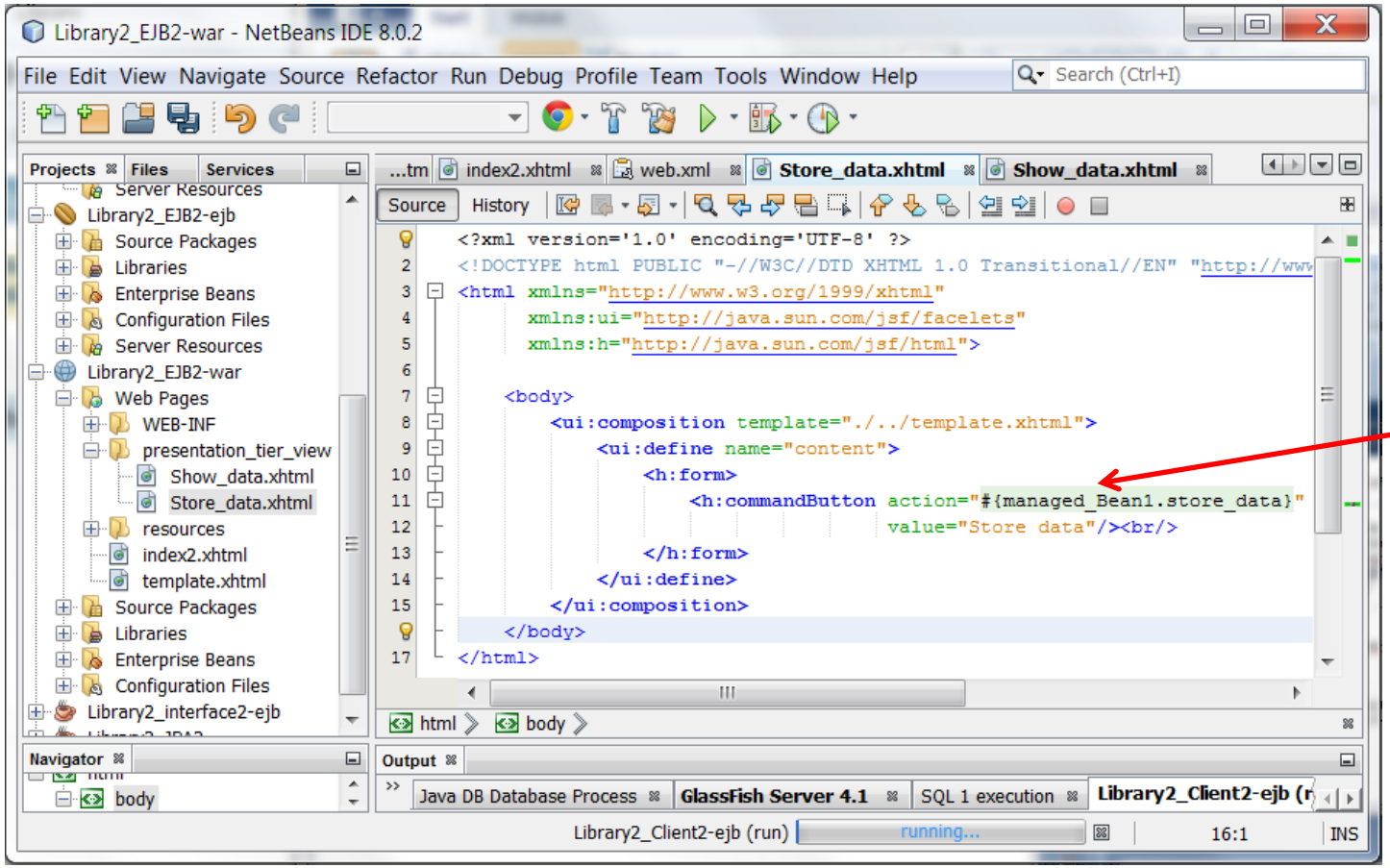






6.5. /6.5.1. Addition the two JSF pages, based on the previous defined template for **inserting data in the database** and **displaying the data from databases**

6.5.2. The new JSF page for inserting data in the database – Store_data.xhtml

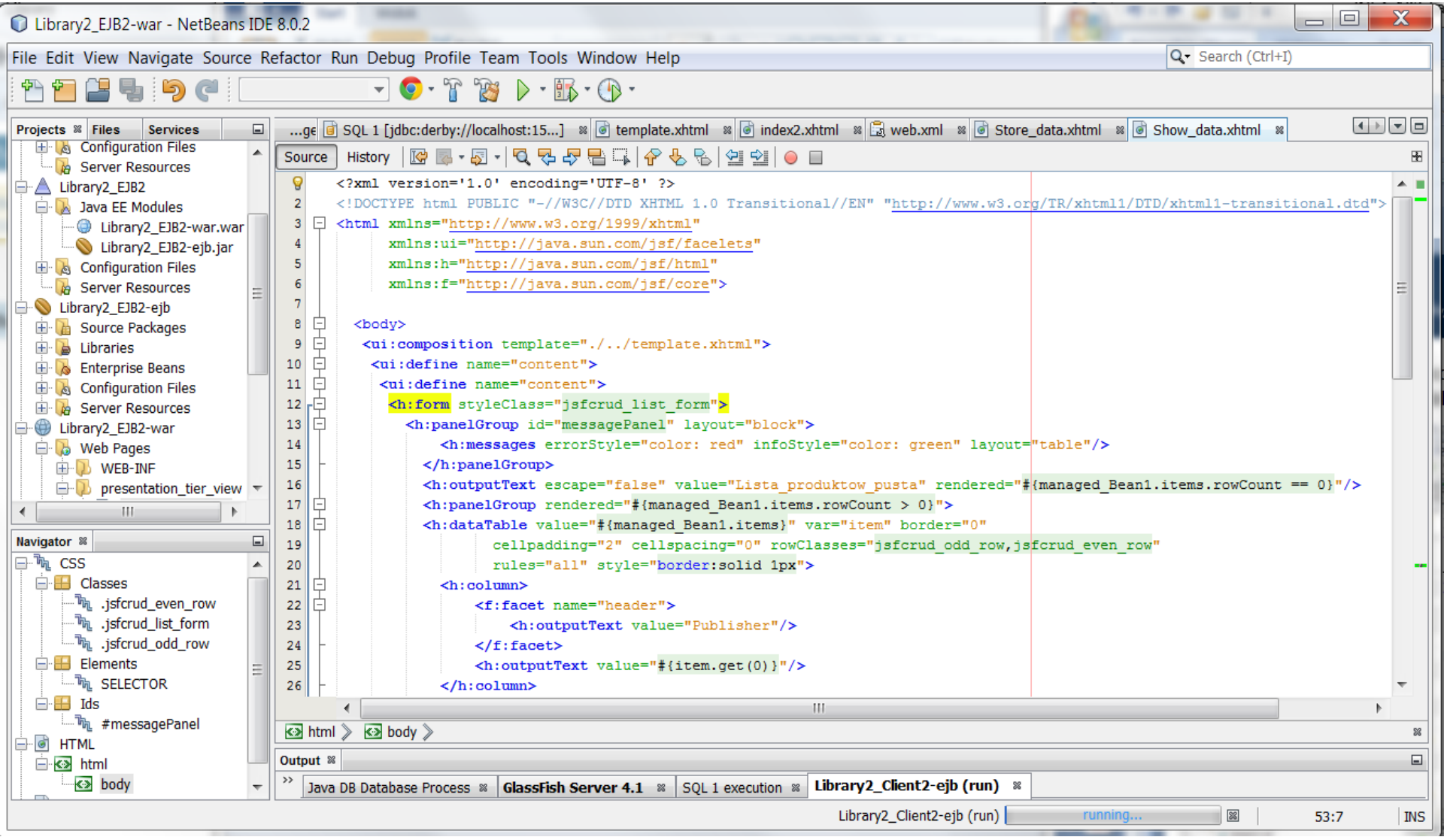


Tag for calling the store_data method from managed_Bean1 object (the Managed Bean type) – its definition is presented further part of this instruction

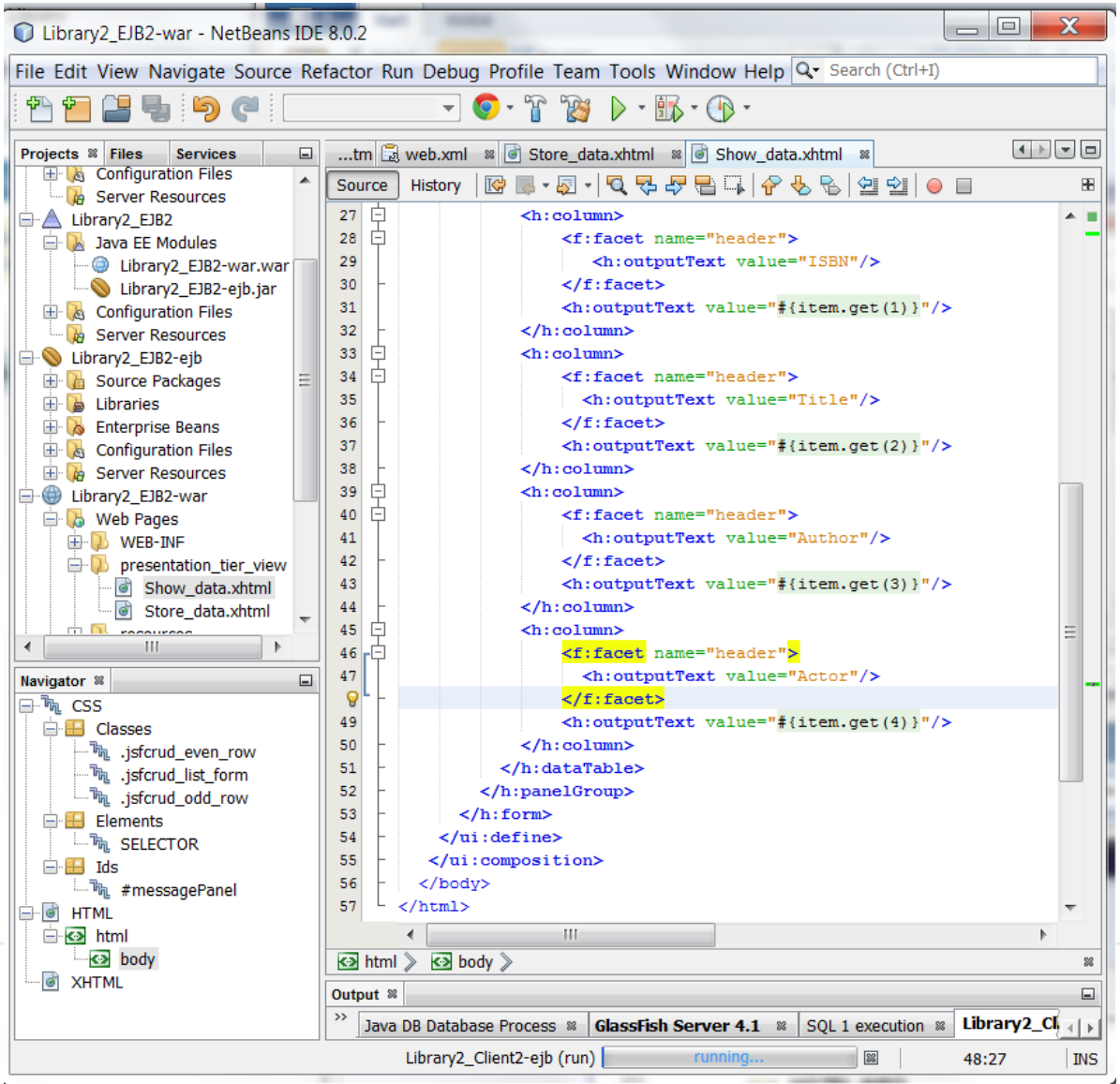
6.5.3. The definition of code of the Store_data.xhtml JSF page.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:ui="http://java.sun.com/jsf/facelets"
     xmlns:h="http://java.sun.com/jsf/html">

<body>
  <ui:composition template="./../template.xhtml">
   <ui:define name="content">
    <h:form>
       <h:commandButton action="#{managed_Bean1.store_data}"
                                value="Store data"/><br/>
    </h:form>
   </ui:define>
  </ui:composition>
</body>
</html>
```

6.5.4. The code of the second JSF page for displaying data from the database , using the h: dataTable JSF component – the Show_data.xhtml page



6.5.5. The code of the second JSF page for displaying data from the database - continuation



6.5.6. The code of the second JSF page for displaying data from the database - continuation

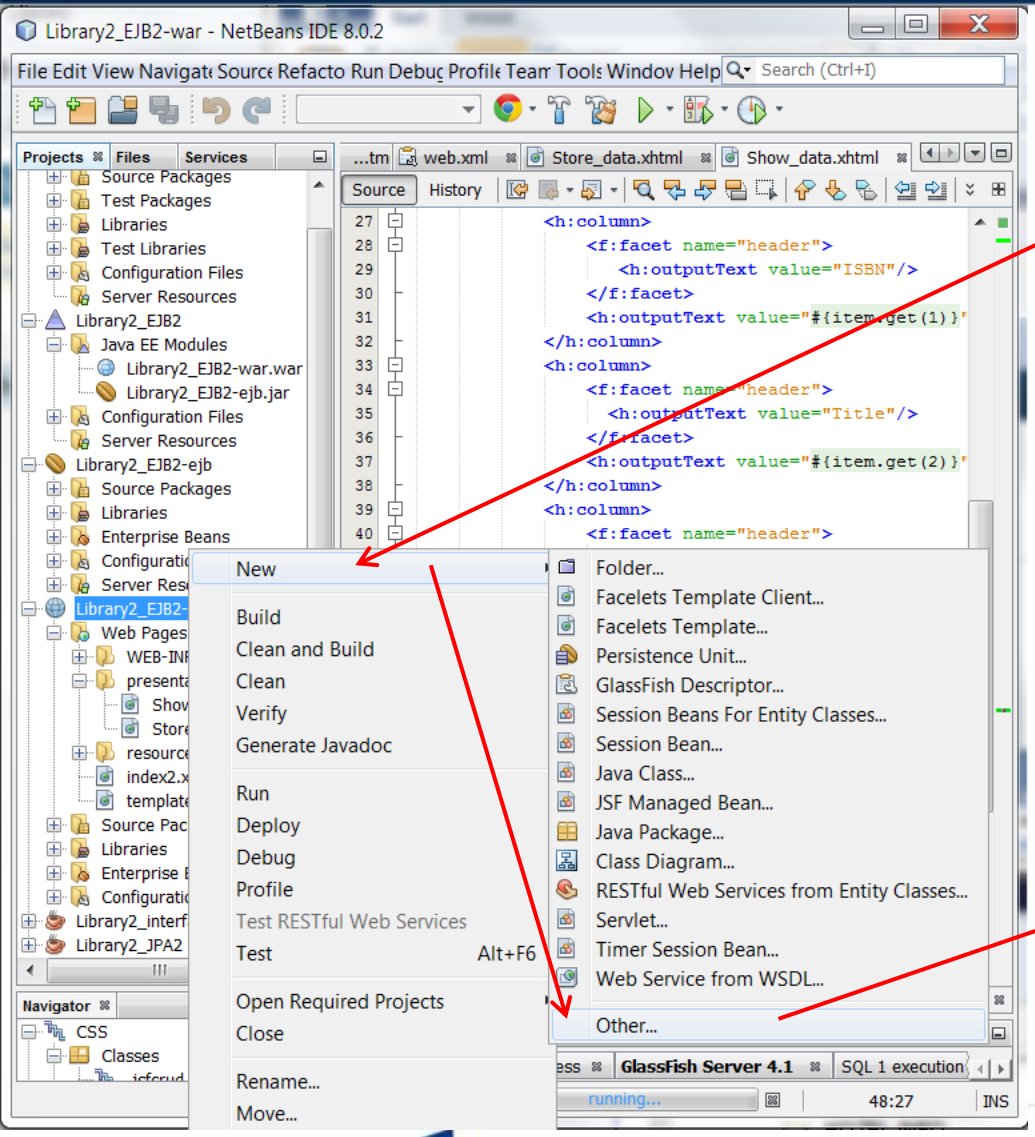
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core">
<body>
<ui:composition template="./../template.xhtml">
 <ui:define name="content">
  <h:form styleClass="jsfcrud_list_form">
   <h:panelGroup id="messagePanel" layout="block">
       <h:messages errorStyle="color: red" infoStyle="color: green" layout="table"/>
   </h:panelGroup>
   <h:outputText escape="false" value="Lista_produktow_pusta" rendered="#{managed_Bean1.items.rowCount == 0}"/>
   <h:panelGroup rendered="#{managed_Bean1.items.rowCount > 0}">
    <h:dataTable value="#{managed_Bean1.items}" var="item" border="0"
            cellpadding="2" cellspacing="0" rowClasses="jsfcrud_odd_row,jsfcrud_even_row " rules="all" style="border:solid 1px">
      <h:column>
        <f:facet name="header">    <h:outputText value="Publisher"/>    </f:facet>
        <h:outputText value="#{item.get(0)}"/>
      </h:column>
      <h:column>
        <f:facet name="header">    <h:outputText value="ISBN"/>          </f:facet>
        <h:outputText value="#{item.get(1)}"/>
      </h:column>
```

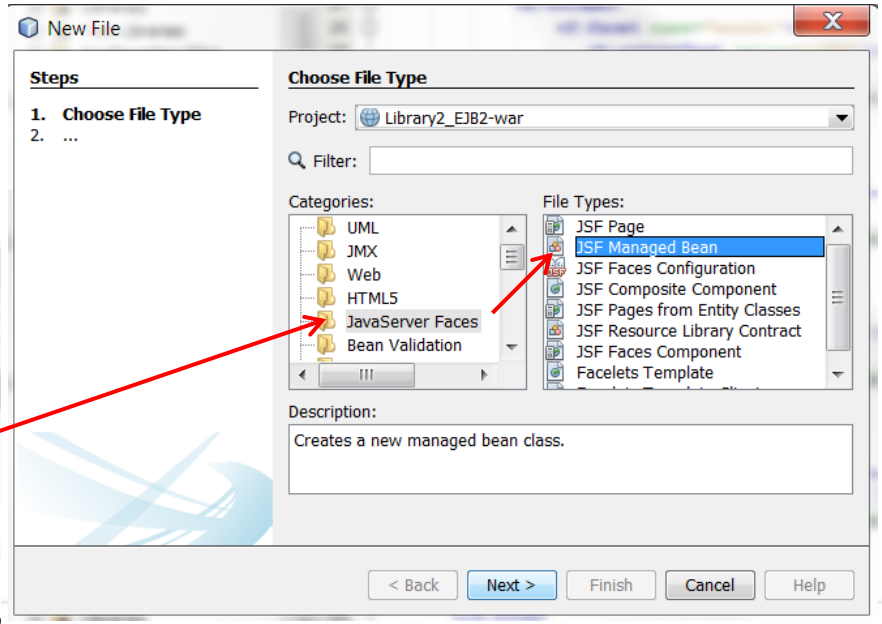
6.5.7. The code of the second JSF page for displaying data from the database - continuation

```
<h:column>
    <f:facet name="header">
      <h:outputText value="Title"/>
    </f:facet>
    <h:outputText value="#{item.get(2)}"/>
</h:column>
<h:column>
     <f:facet name="header">
        <h:outputText value="Author"/>
     </f:facet>
     <h:outputText value="#{item.get(3)}"/>
</h:column>
<h:column>
     <f:facet name="header">
        <h:outputText value="Actor"/>
     </f:facet>
     <h:outputText value="#{item.get(4)}"/>
</h:column>
</h:dataTable>
</h:panelGroup>
</h:form>
</ui:define>
</ui:composition>
</body>
</html>
```

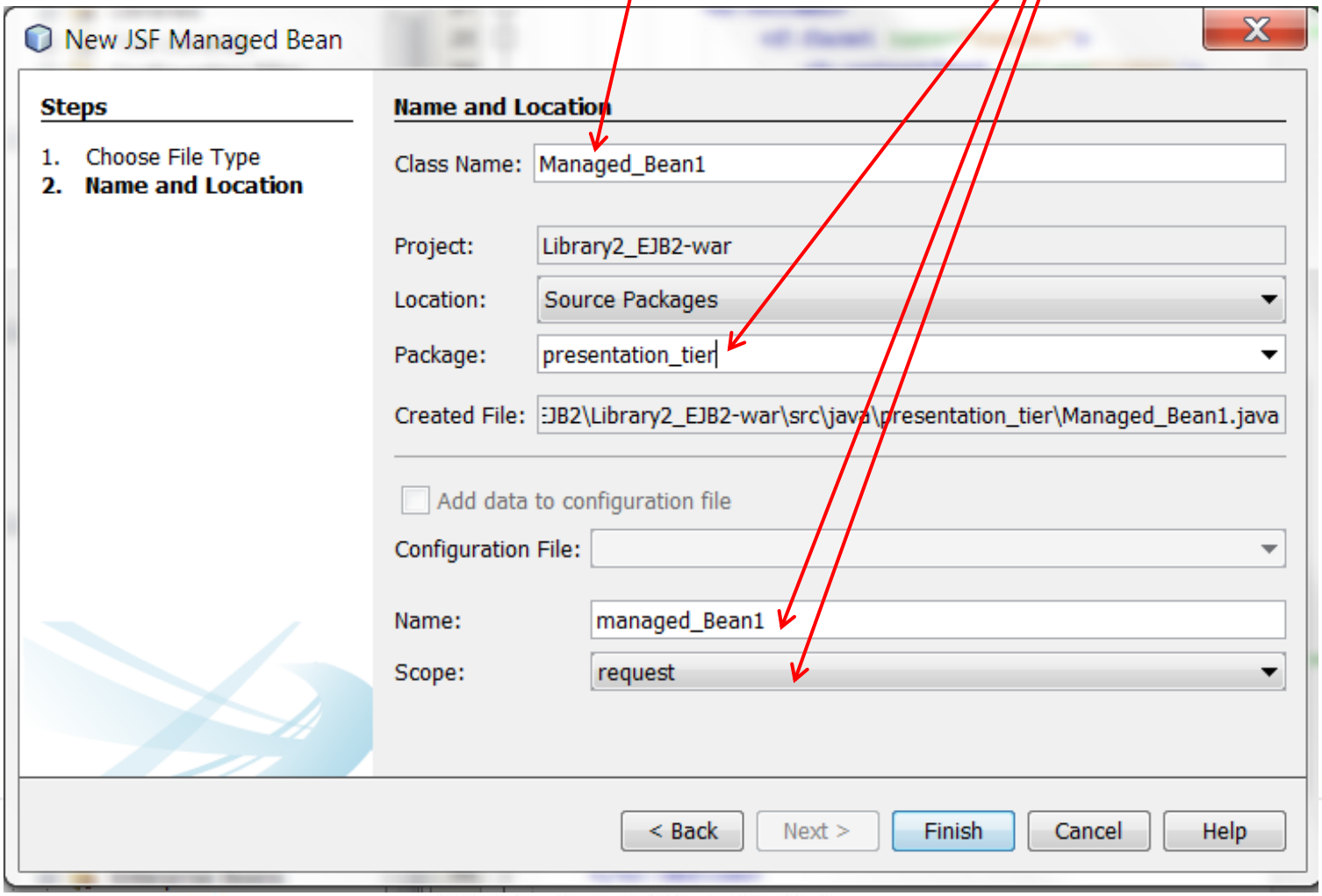
Project co-financed from the EU European Social Fund

6.6. /6.6.1. Creation of the Managed Bean type of object as the Presentation Tier object based on JSF technology – right click the Library2_EJBB2-war item in the Projects item, select the JavaServer Faces/JSF Managed Bean File Types. Click Next.

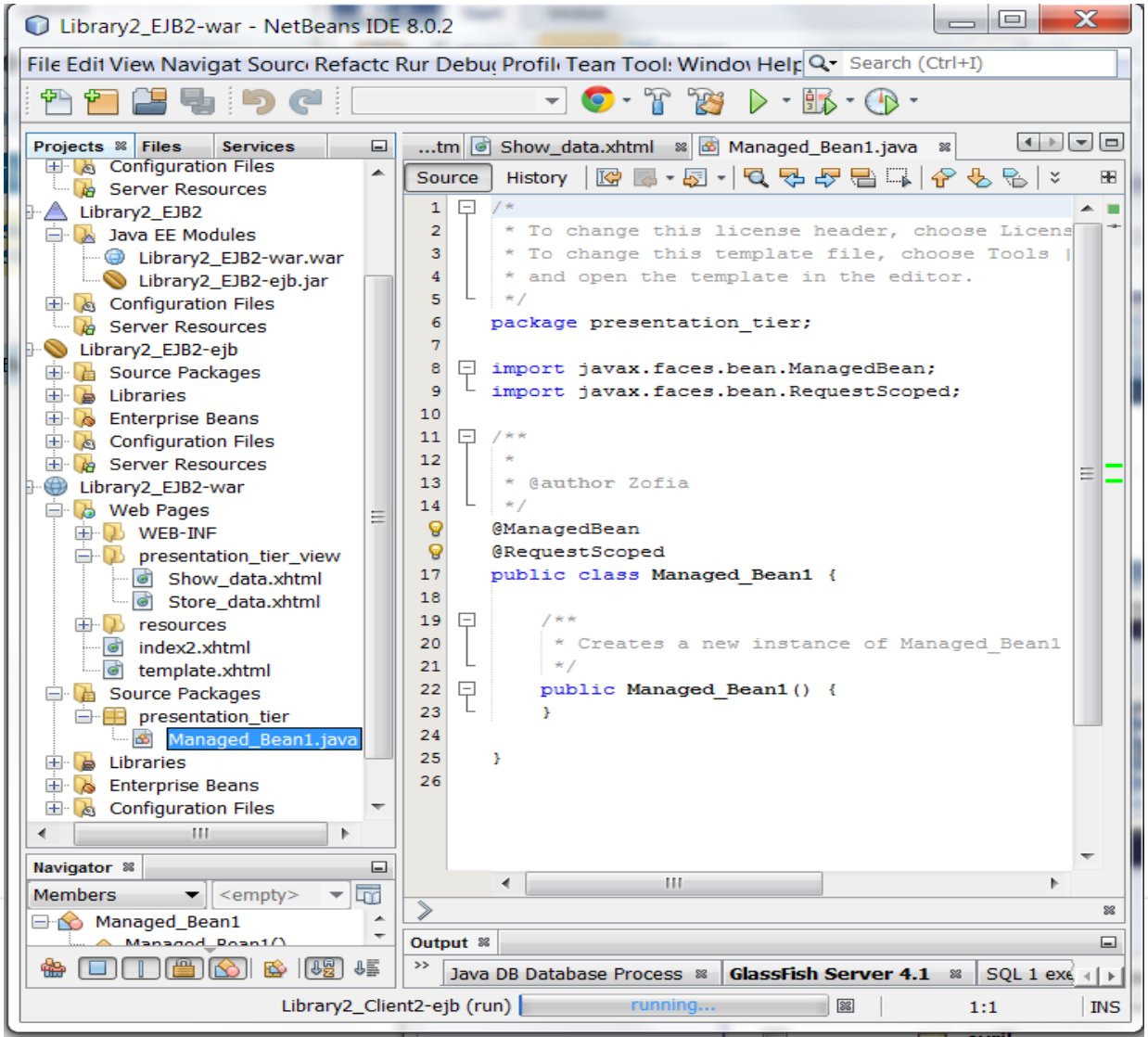




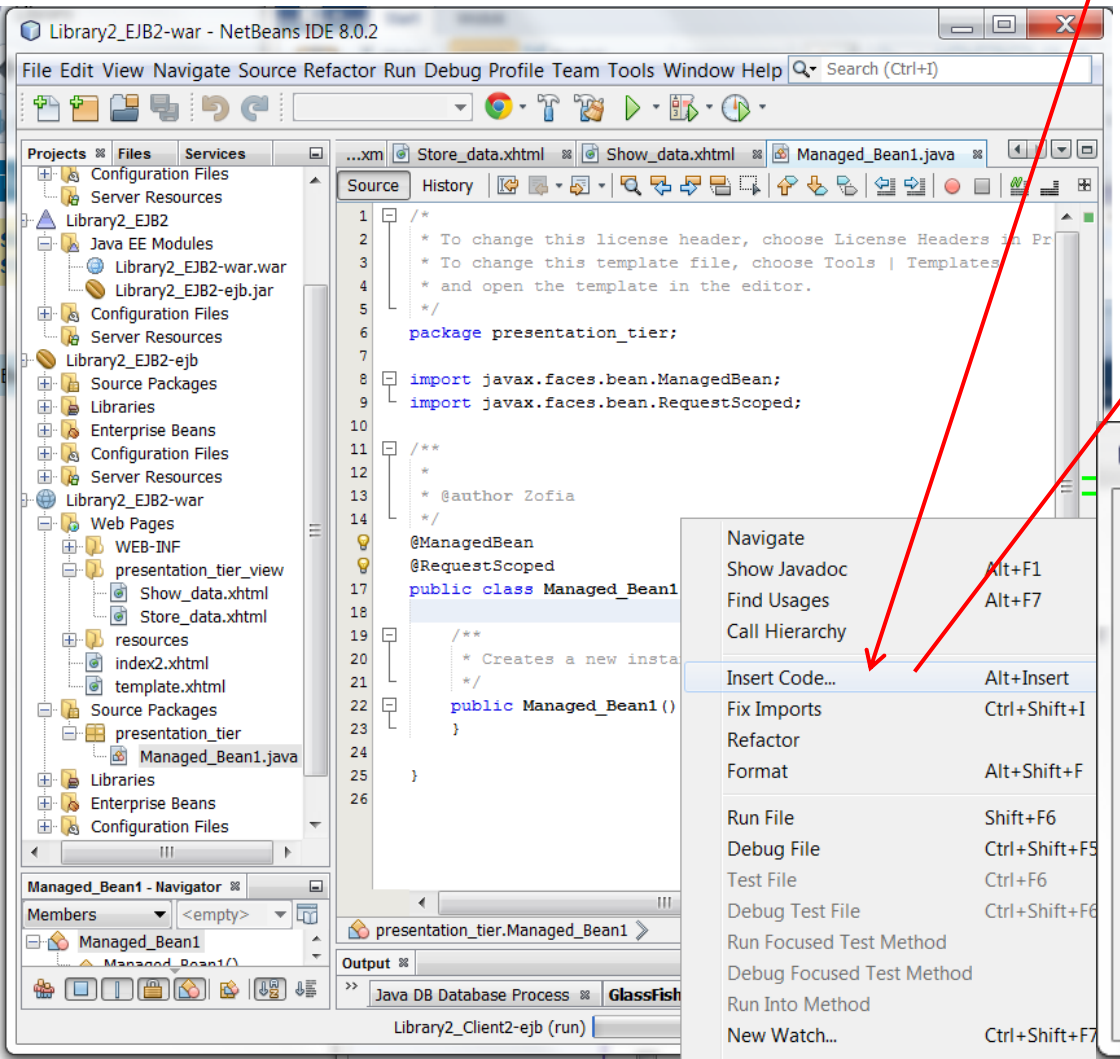
6.6.2. Creation the Managed Bean type of object as the Presentation Tier object based on JSF technology - continuation. Fill the ClassName field, name of package in the Package field, setup the scope of Managed Bean and fill its name in the name field.

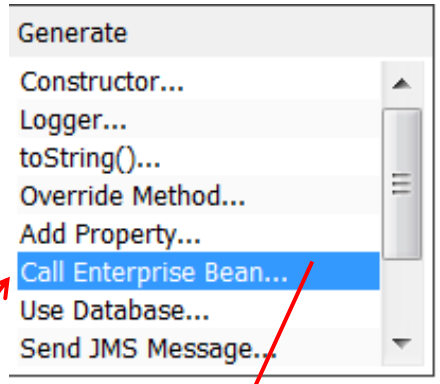


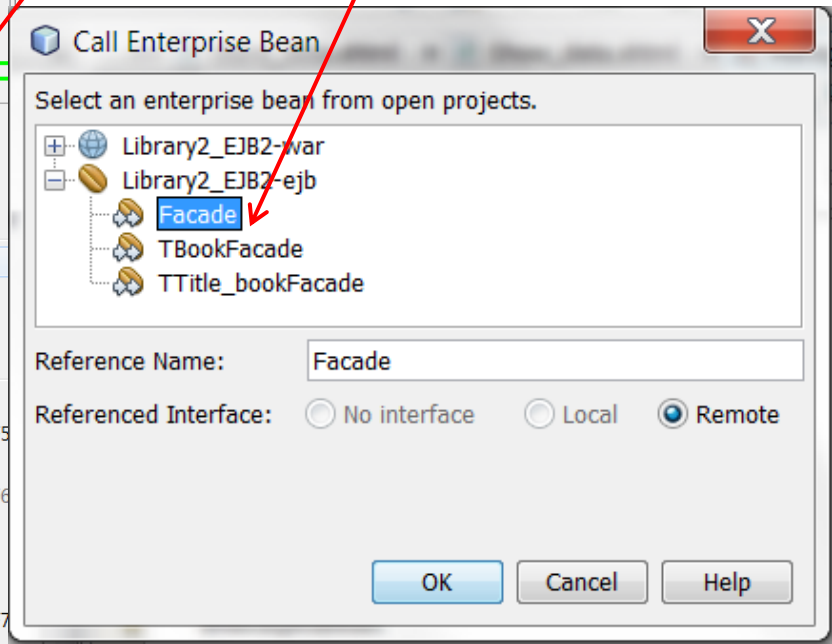
6.6.3. Creation of the Managed Bean type of object as the Presentation Tier object based on JSF technology - the generated code of this class



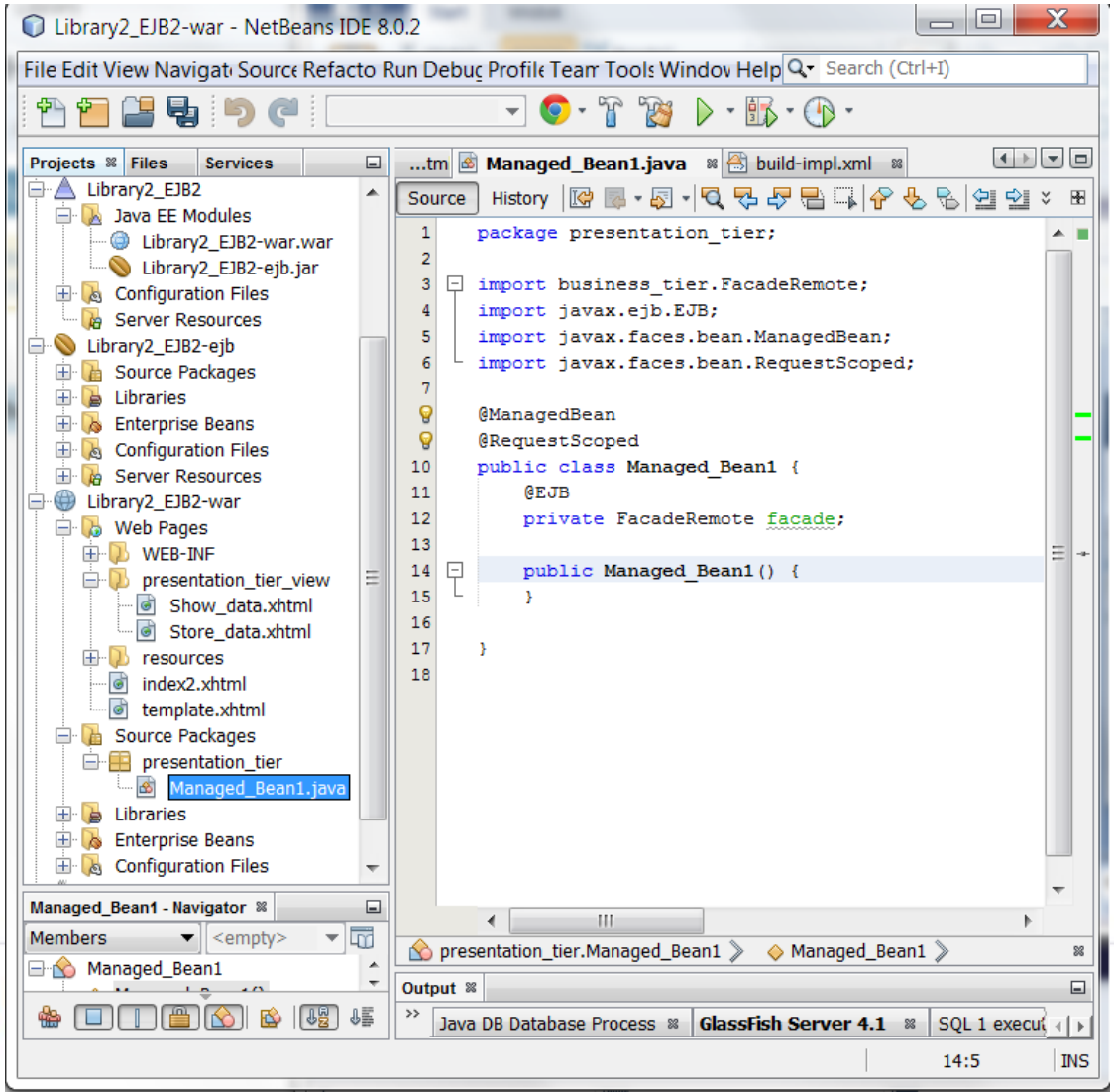
6.6.4. Creation the connection with the Facade remote session bean from the Library2_EJB2-ejb module by using the mechanism of Insert code (right click the code in the editor window, select the the Insert Code… item, and then select the Call Enterprise Bean, and at last select the Facade component from the list in the Call Enterprise dialog). Click OK..







6.6.5. The code of the Managed_Bean1 class with added reference of the Facade EJB by using the annotation and injecting mechanism.



6.6.6. The code of the Managed_Bean1 class

```
package presentation_tier;

import business_tier.FacadeRemote;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;

@ManagedBean
@RequestScoped
public class Managed_Bean1 {
   @EJB
   private FacadeRemote facade;

   private DataModel items;

   public Managed_Bean1() {   }

   public FacadeRemote getFacade()                { return façade;  }
   public void setFacade(FacadeRemote facade)     { this.facade = façade;   }

   public String store_data() {
      try {
         facade.add_titles();
         facade.add_books();
      } catch (Exception e) {    }
      return "/faces/index2";
   }
```

The model of the dataTable component used on the Show_data.xhtml JSF page for displaying the data from the database

The **store_data** method for handling event of the h:commandButton component, used by the Store_data.xhtml page. This method calls two methods from the Facade session bean, which inserting data into the database by using ORM (JPA 2.1 controllers from Library2_EJB2-ejb module). As the response, it returns to the main index2.xhtml JSF page (p. 3-7, 24-25)

6.6.7. The code of the Managed_Bean1 class - continuation

```
public DataModel create_DataModel() {
   try{
   return new ListDataModel( facade.titles());
   }
   catch(Exception e)
   {
      System.out.println("Blad");
      return null;   }
}
```

The **create_DataModel** method for creation the new data model for h:DataTable component – this based on data returned from the titles method of the Facade session bean and they are read from database by using ORM mechanism (p. 24-25, 3-7)

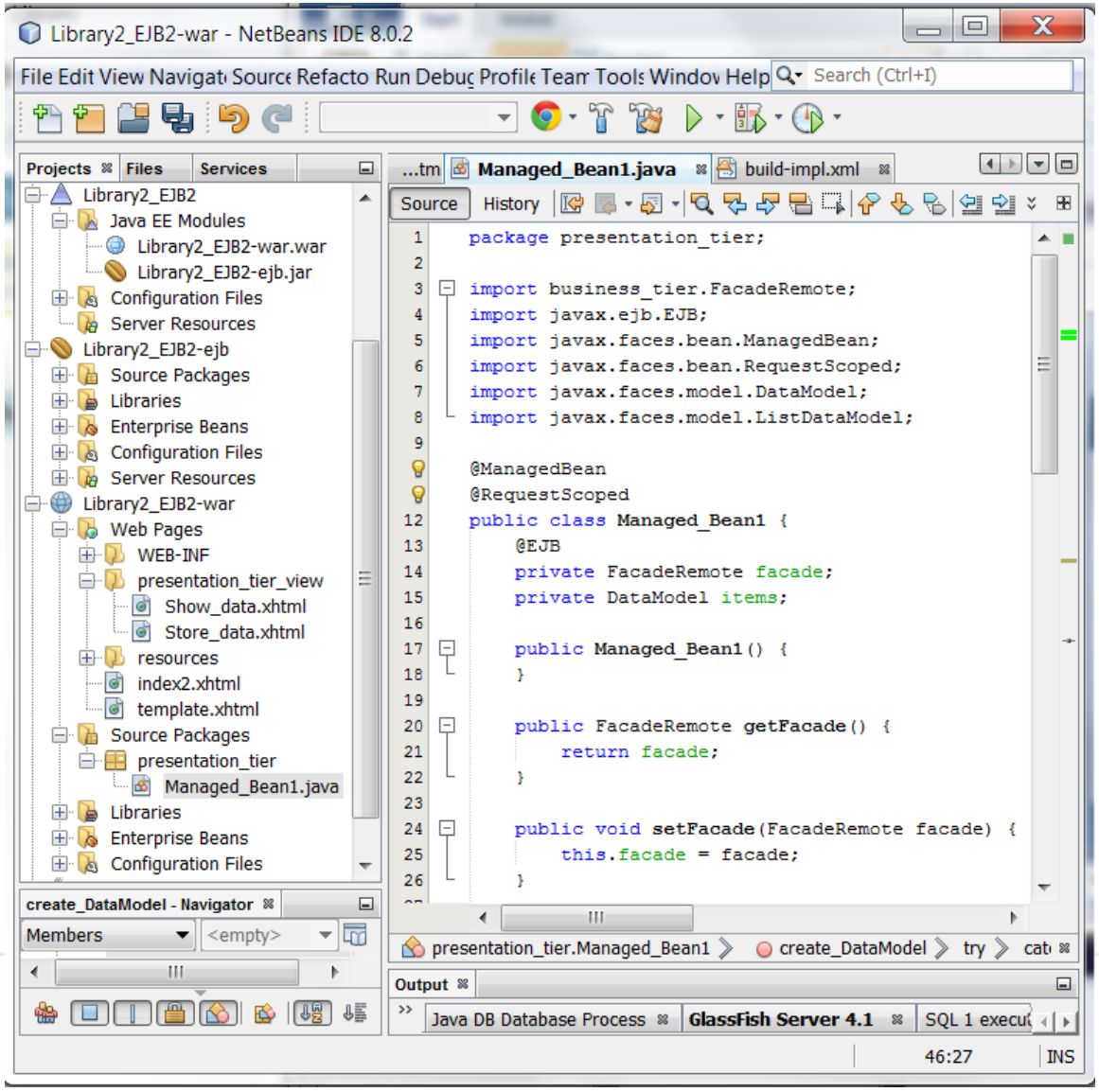
```
public DataModel getItems() {
   if (items == null) {
       items = create_DataModel();     }
   return items;
}
```

The **getItems** method for creation the new data model for h:DataTable component by using binding mechanism – this based on data returned from the create_DataModel (at the previous slide).

```
public void setItems(DataModel items) {
   this.items = items;   }
}
```

Project co-financed from the EU European Social Fund

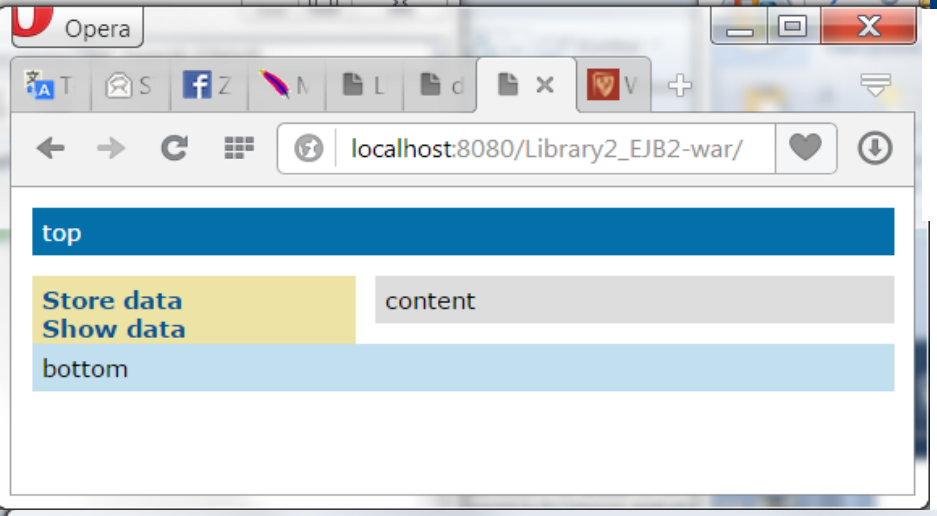
6.6.9. The code of the Managed_Bean1 class – the result



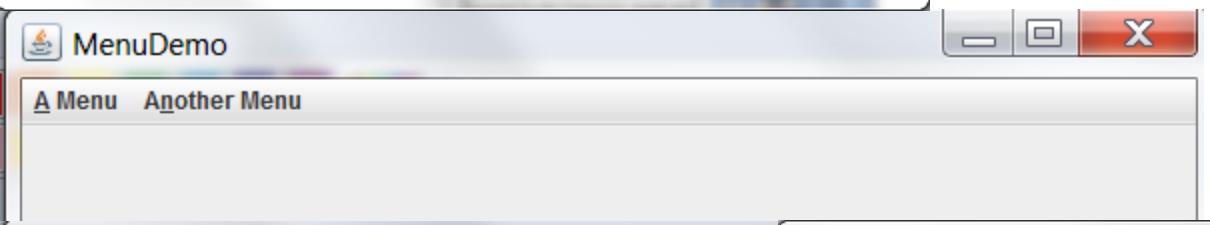
7.0. Running the program

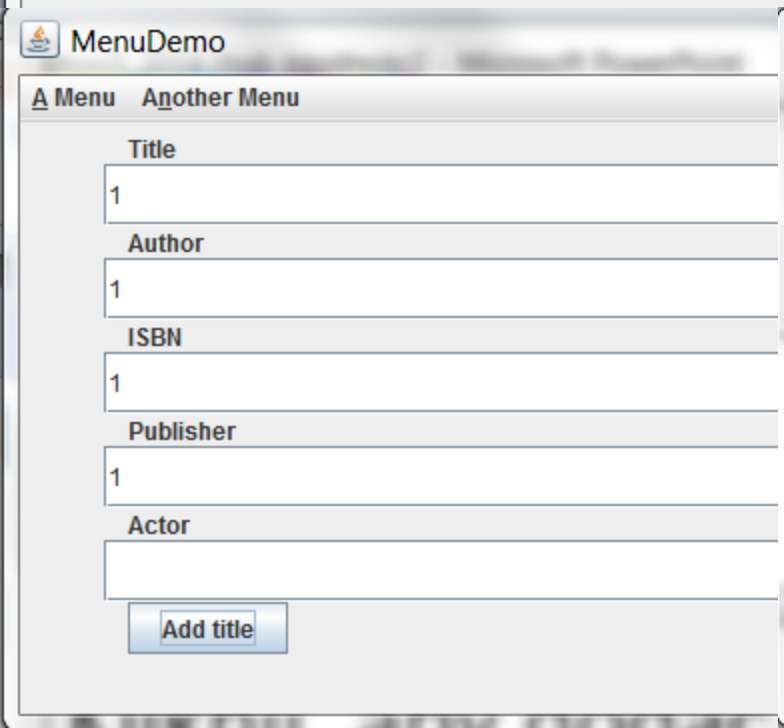
1. Connect with the Library2015 database (5 slide), if you start first one.
2. If you start after introduction of some changes in your programs, you must undeploy all deployed programs: Library2_EJB2 , Library2_Client2-ejb and Library2_EJB2-war, accordingly to the 81 slide. After this development, your program will be executed properly, if you clean and build the following programs (necessary, if you make some changes or start first time):
    1. Library2_JPA2
    2. Library2_interface2-ejb
    3. Library2_EJB2-ejb
    4. Library2_Client2-ejb
    5. Library2_EJB2-war
3. Then you must **deploy** the Library2_EJB2 program if start first one.
4. Finally, you may **run** a few instances of Library2_Client2-ejb programs.
5. At last, insert the following url address: http://localhost:8080/Library2_EJB2-war/ in any browser and run a few web clients .
6. These programs (p.3 i 4) share the common data as titles and books.
7. In the **Service Tab** you may see, if your EE program deploy properly (Server item). The other useful information you may get from the **GlassFish output window tab**.
8. If you stop your above programs, you must undeploy them, accordingly to the 81 slide.

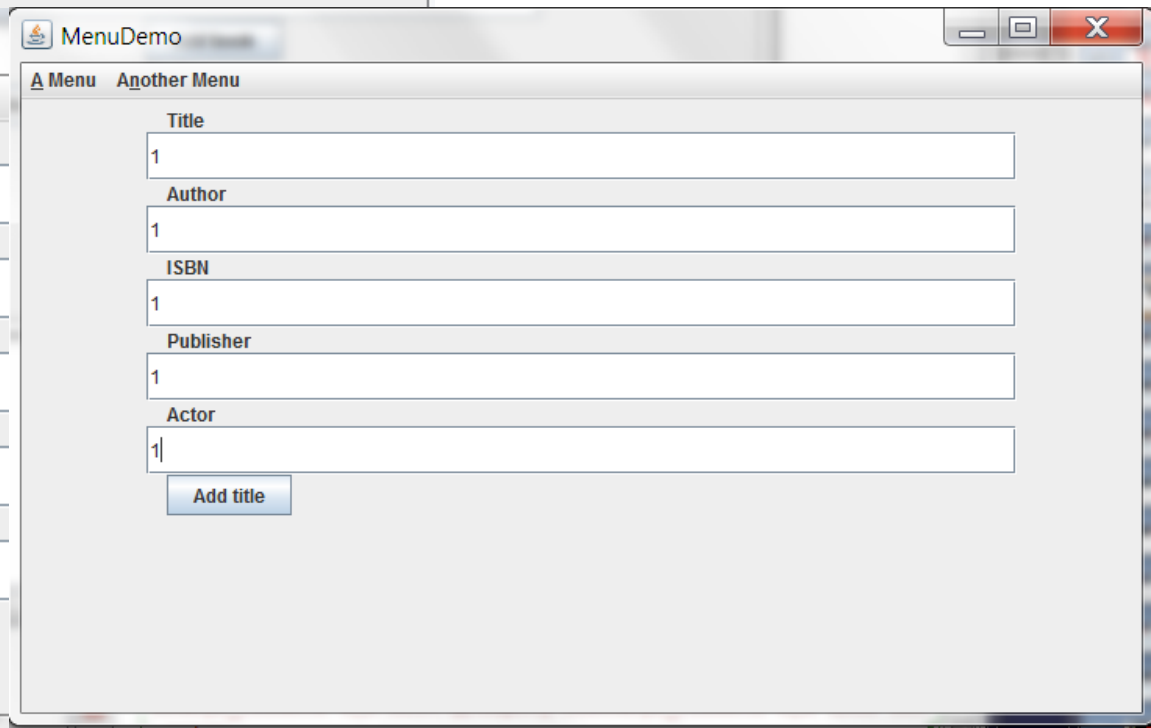
7.1. The view of the main web page (rendered by using the index2.xhtml JSF page) of the **Library2_EJB2-war project**



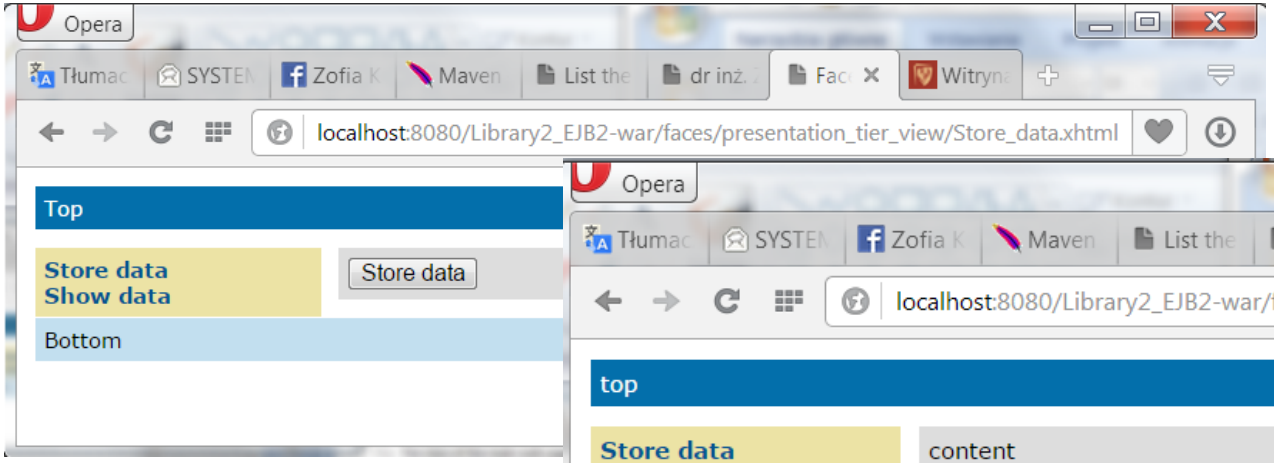
7.2. The view of the Enterprise Application Client (**Library2_Client2-ejb**) for processing of application data - with the same responsibilities as of the version of third laboratory.



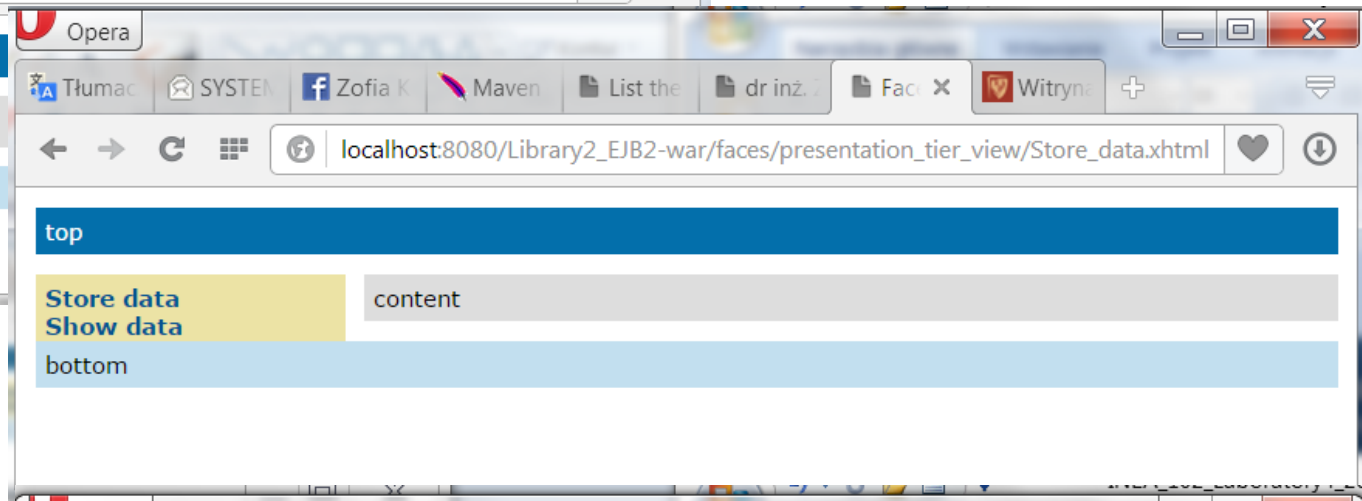




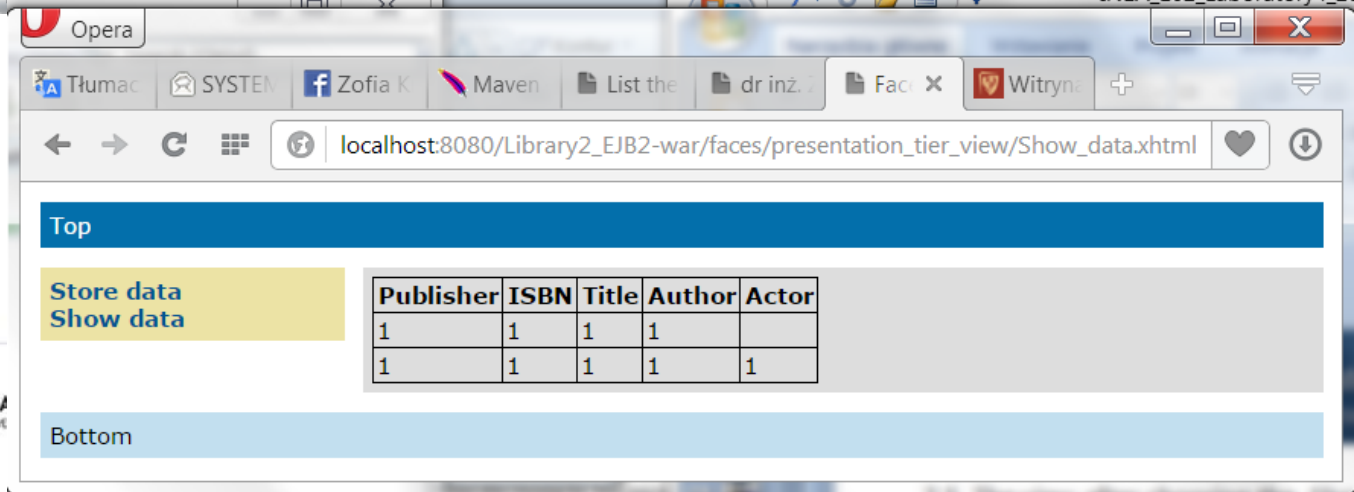
7.3. The view after choose the Store data link (the left part of page) of the **Library2_EJB2-war** client



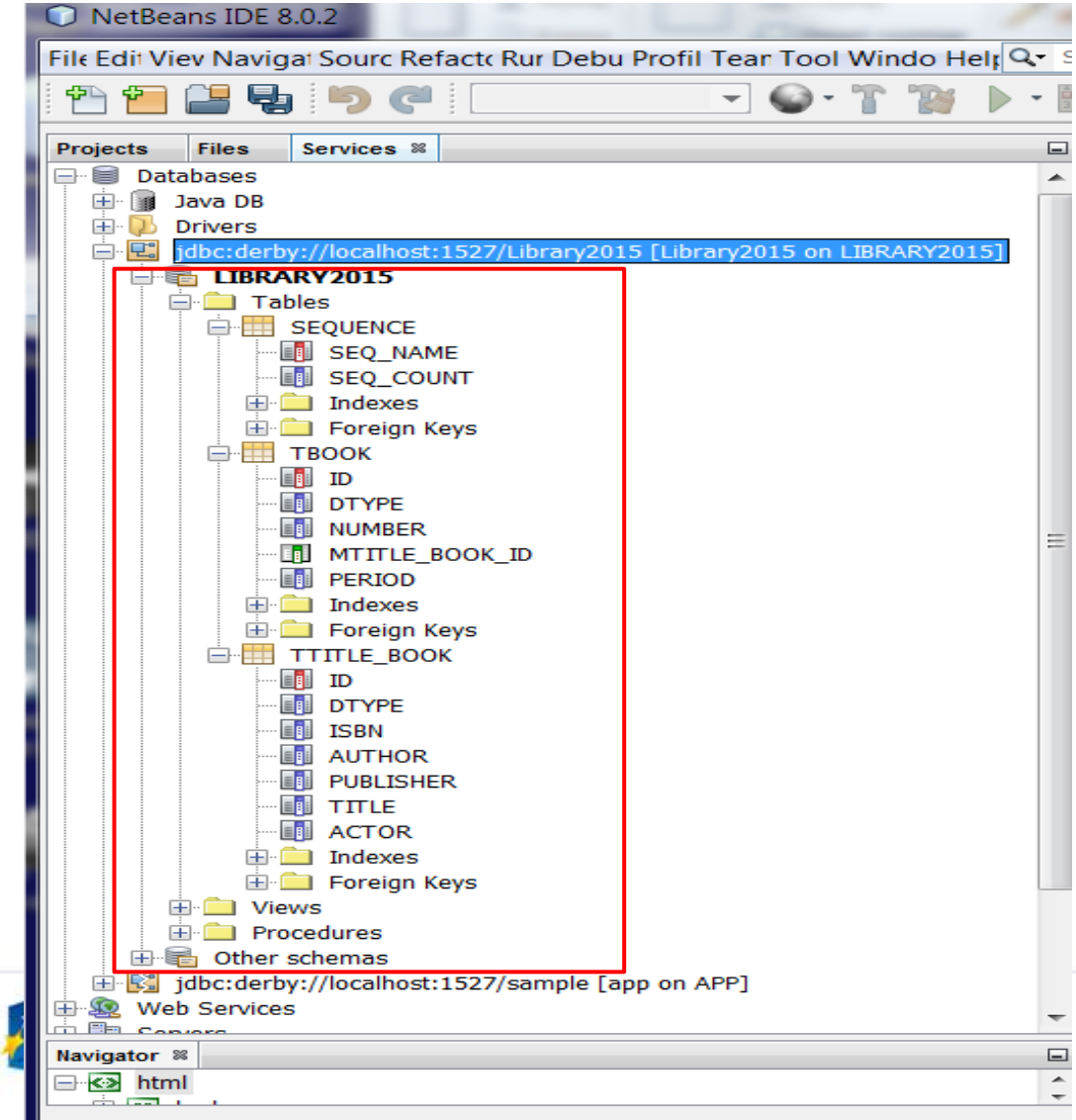
7.4. The view after choosing the Store data button (the right part of page) – i.e. after inserting the application data into the database (p.7.2) by the **Library2_EJB2-war** client



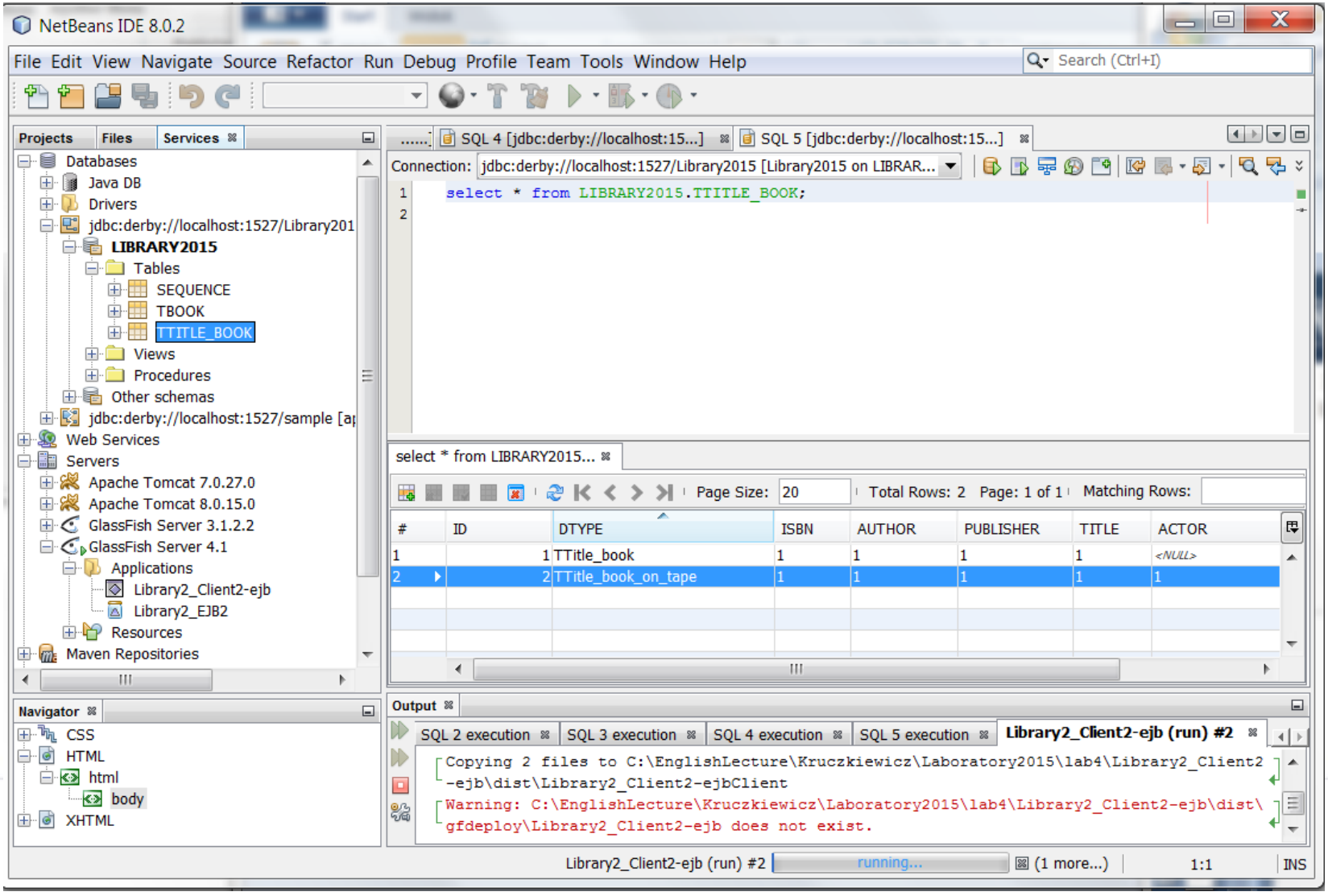
7.5. The view after choosing the Show data button (the left part of page) – i.e. after getting data from the database by the **Library2_EJB2-war** client

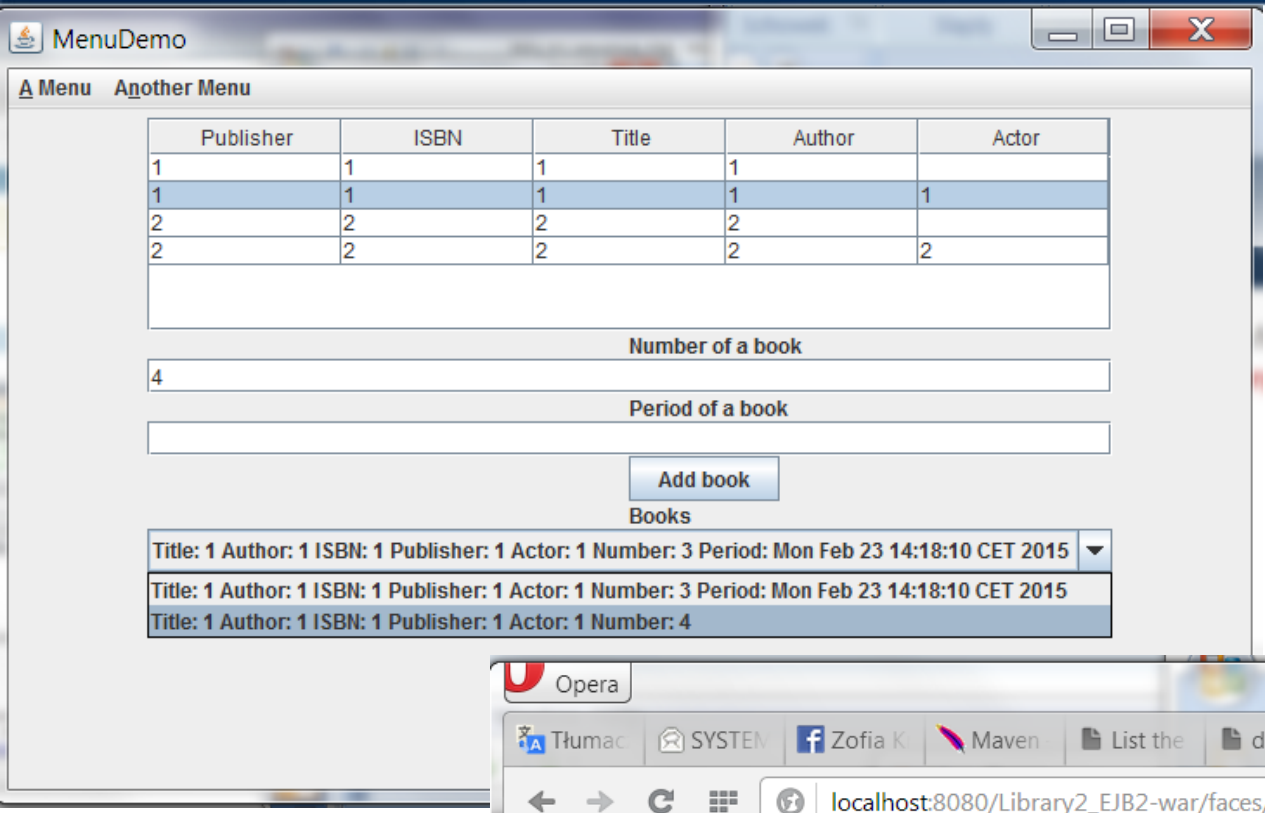


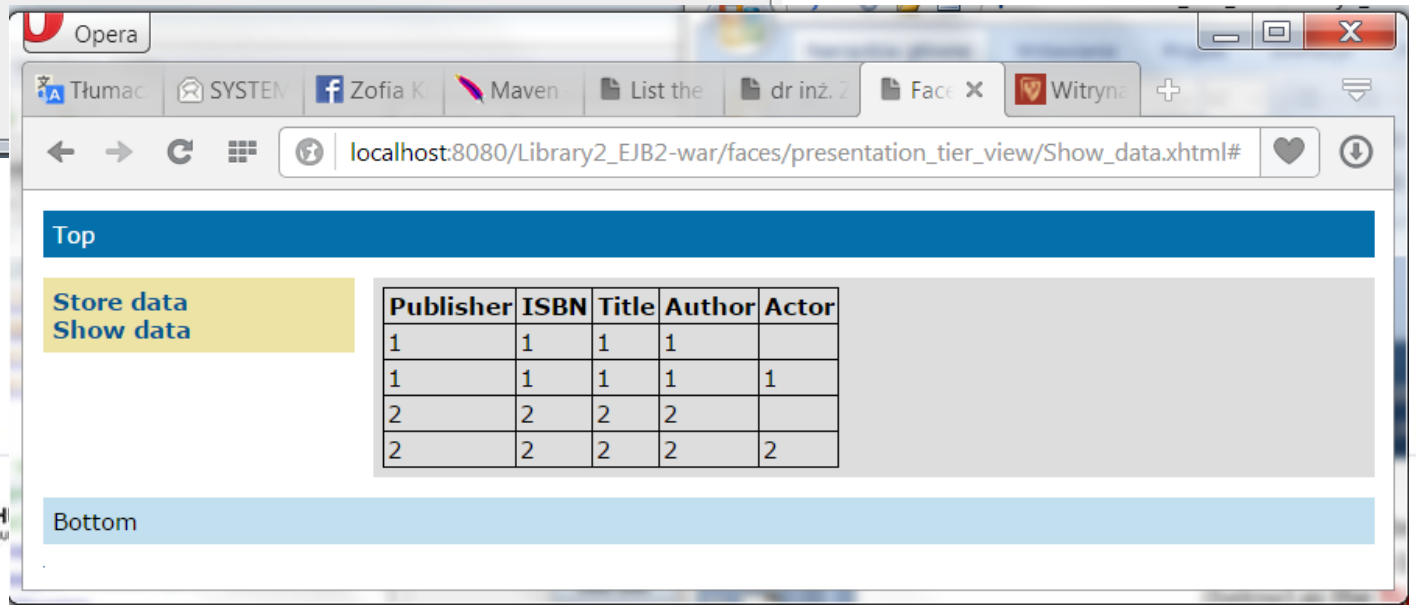
7.6. The generated tables by using the ORM mechanism, accordingly to the annotation placed in the Entity classes (the instruction of the fourth laboratory)



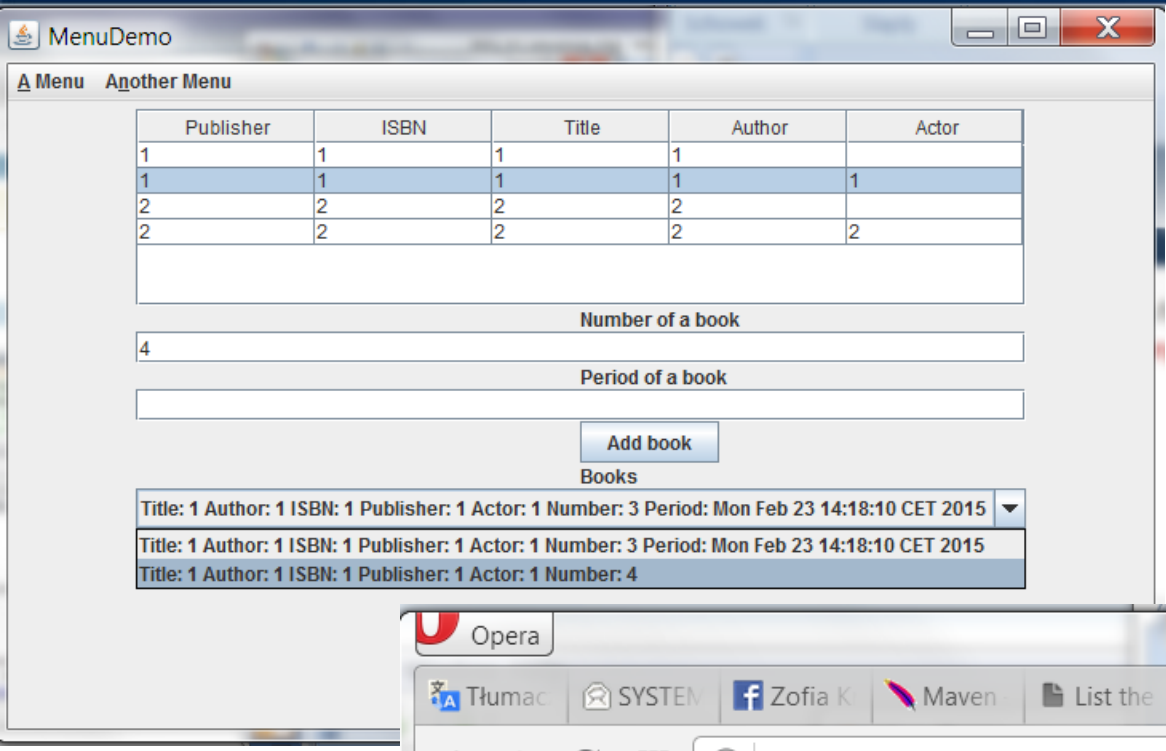
7.7. The data stored in the database by Store data web page- at previous time they have been inserted by the Enterprise Application Client program (**Library2_Client2-ejb**) as the application data (p.7.2)

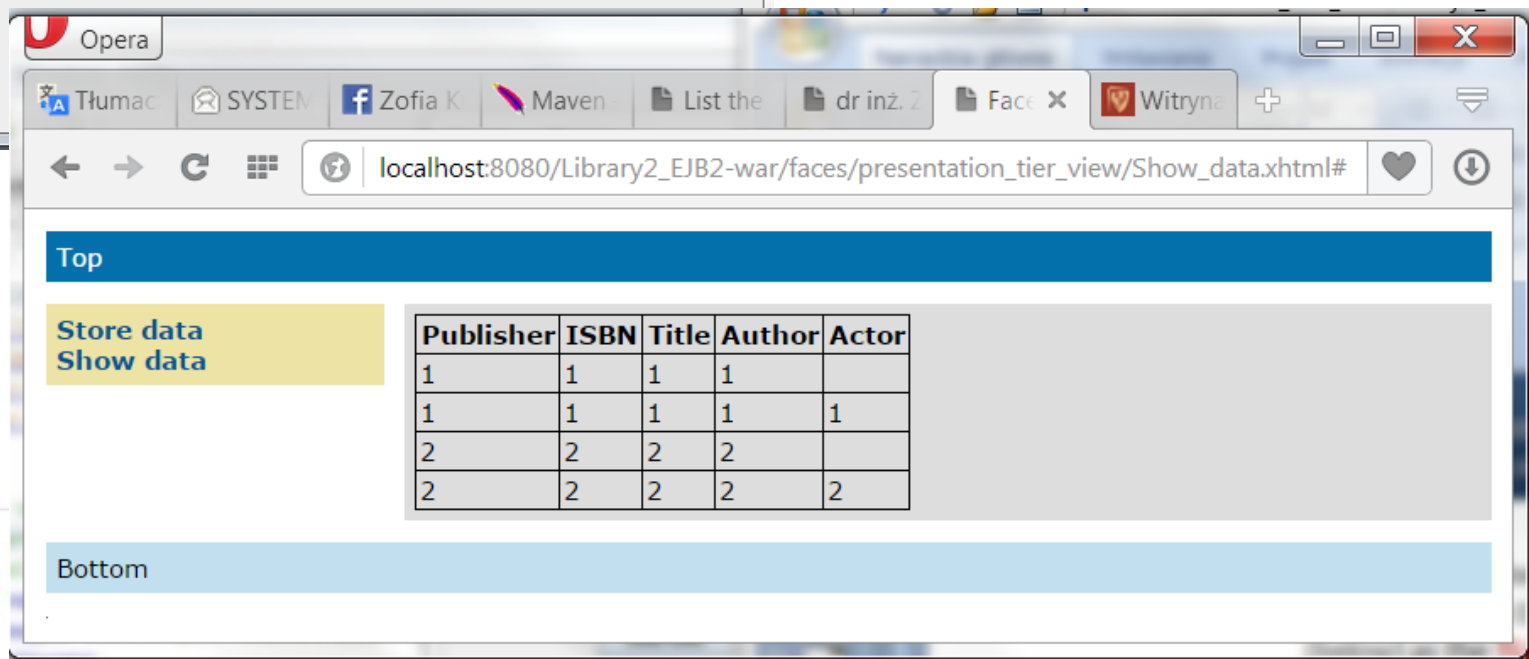






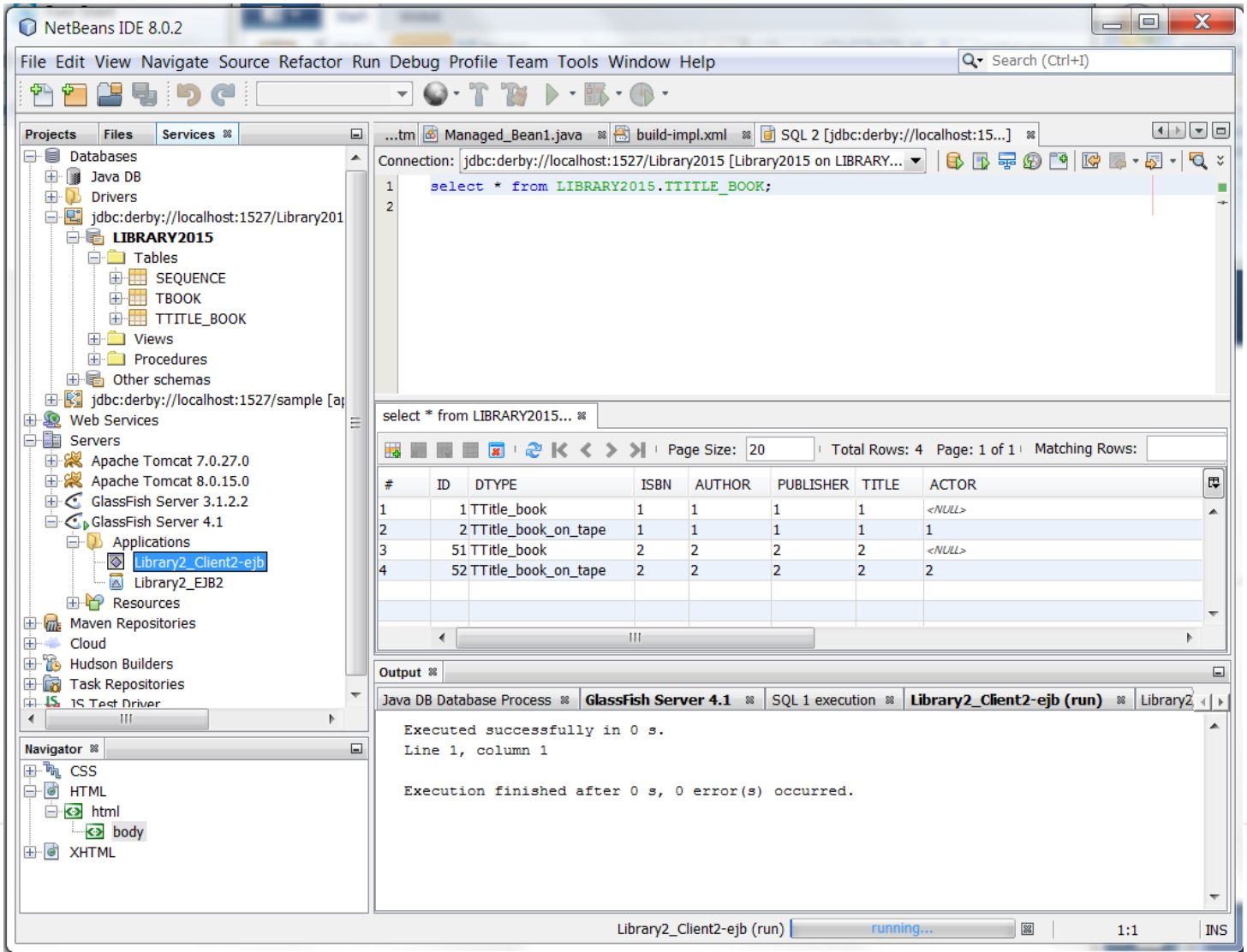
7.8. The **Library2_Client2-ejb** form to adding the new books of the selected title, as the application data.



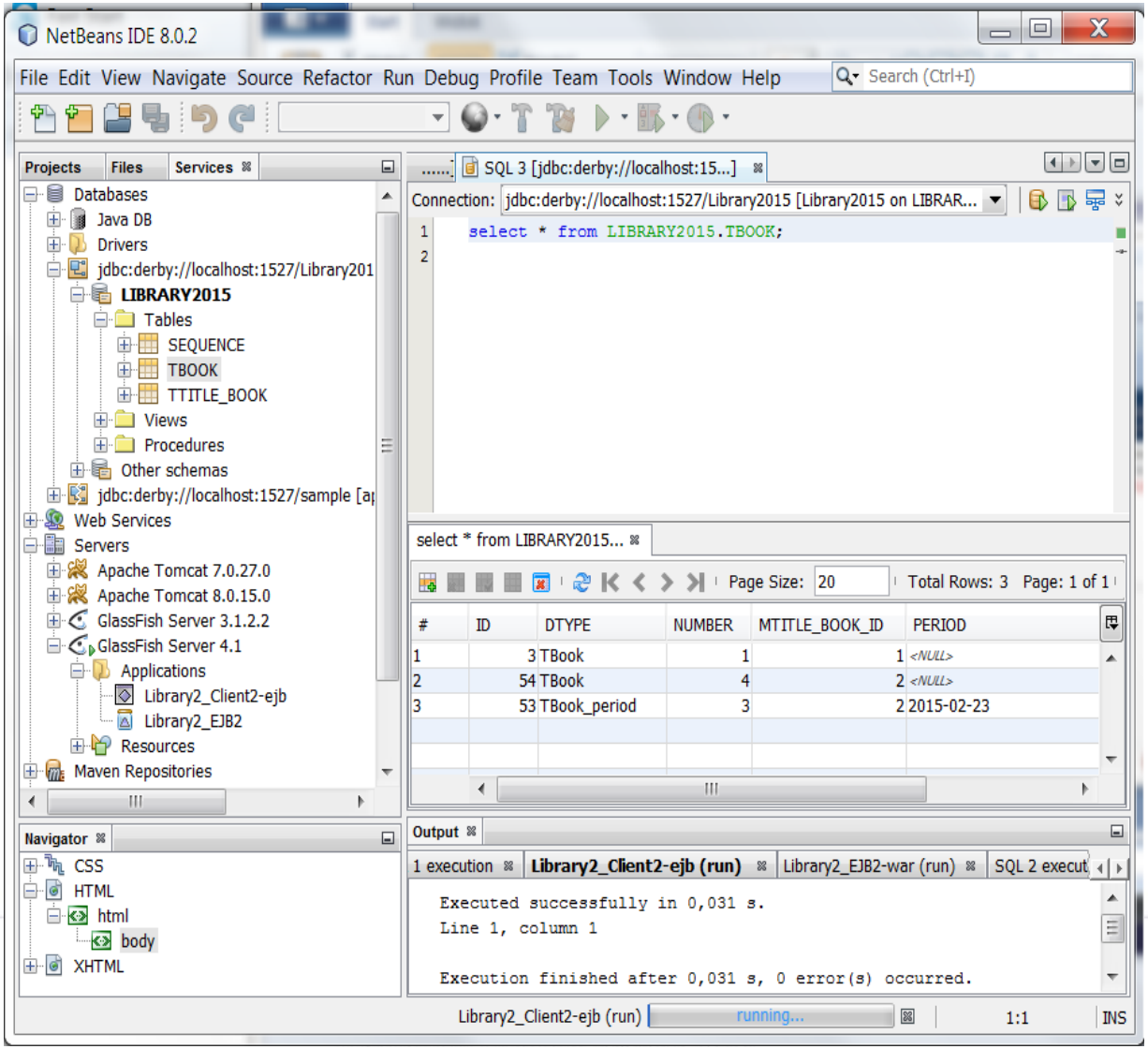


7.9. Restored data from database, after again opening EE application with two kinds of clients: **Library2_Client2-ejb** as the Enterprise Application client (below) and **Library2_EJB2-war** as the web client

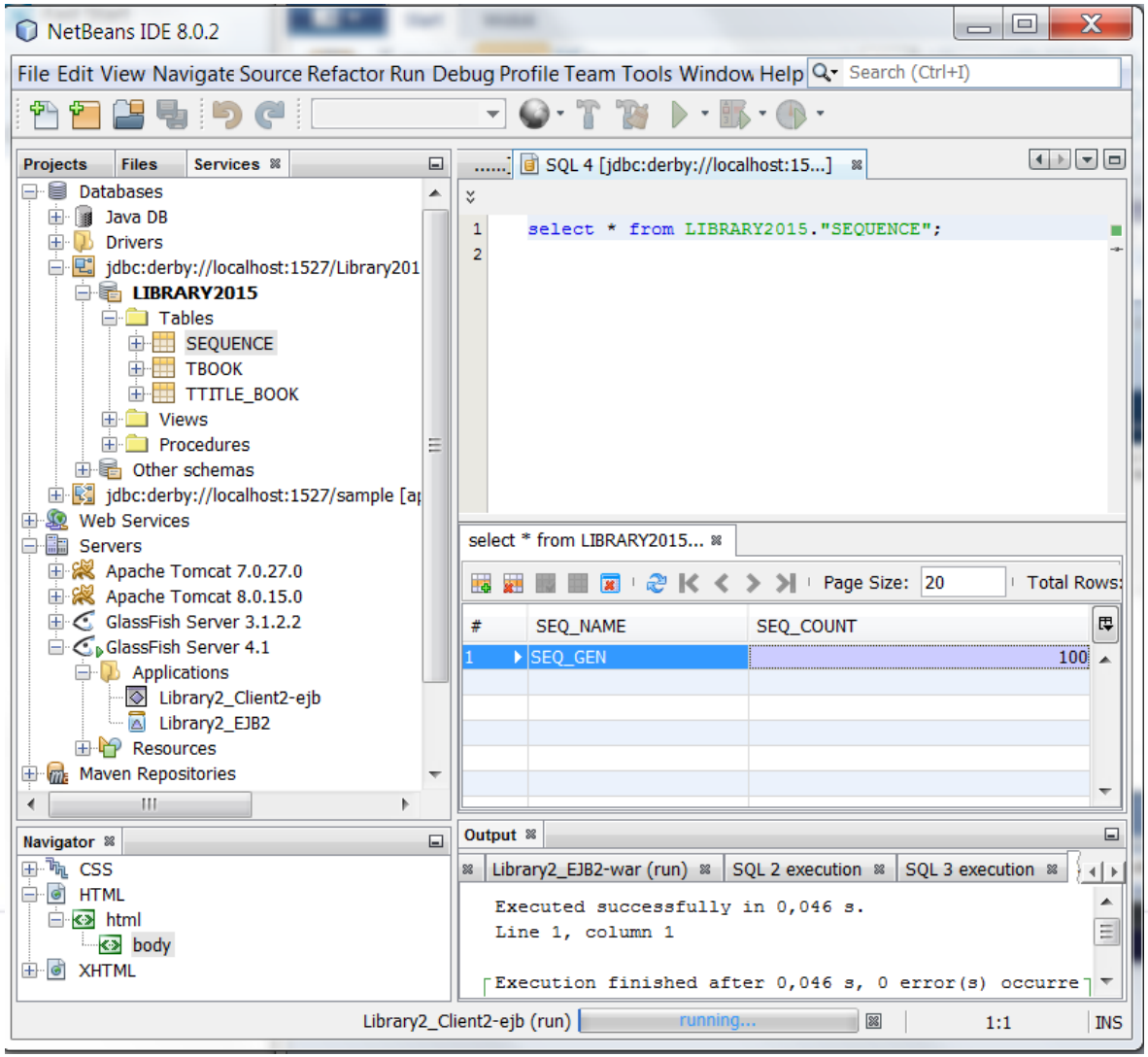
7.10. The data stored in the database (titles) by Store data web page- at previous time they have been inserted by the Enterprise Application Client program (**Library2_Client2-ejb**) as the application data (p. 7.8)



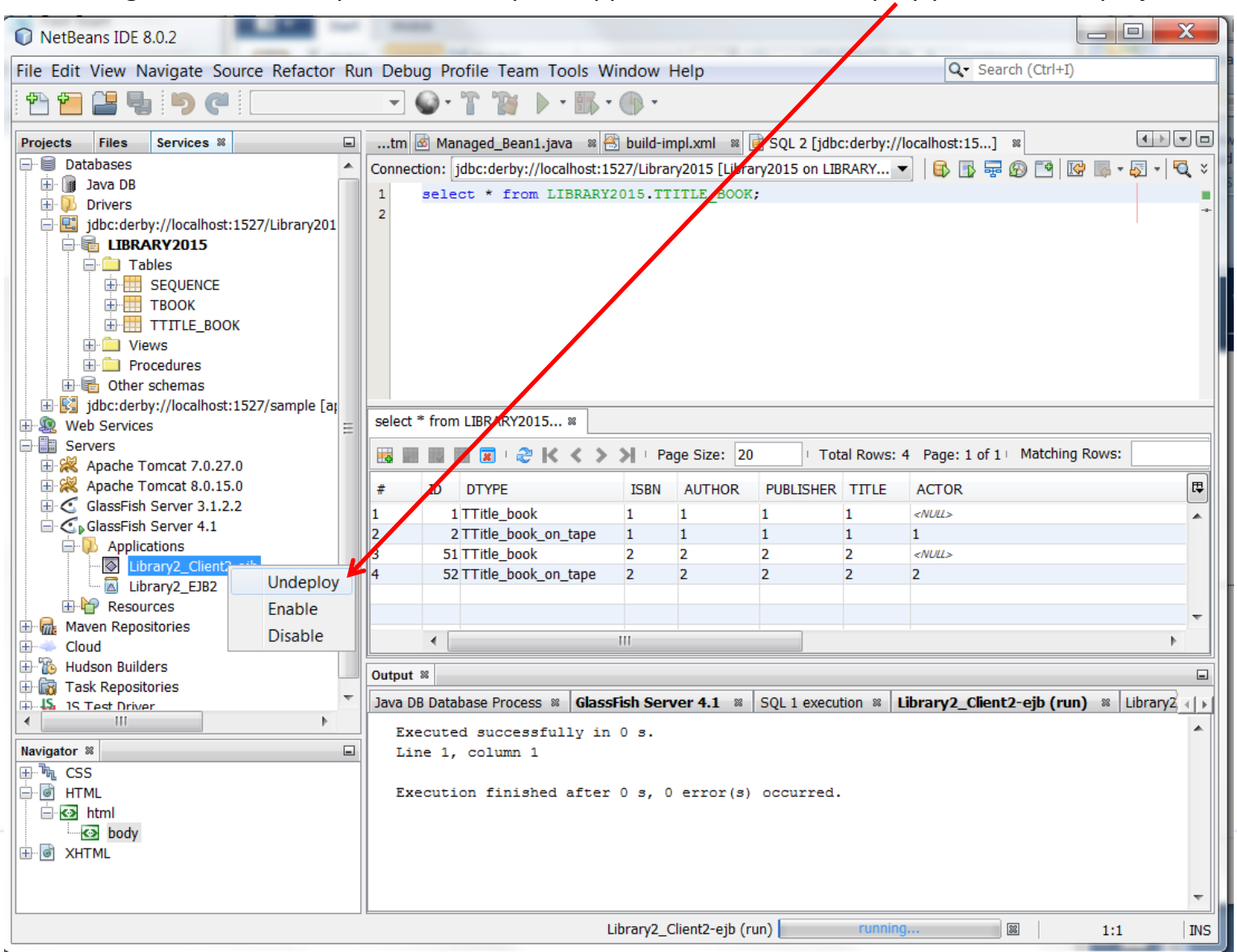
7.11. The data stored in the database (books) by Store data web page- at previous time they have been inserted by the Enterprise Application Client program (**Library2_Client2-ejb**) as the application data (p. 7.8)



7.12. The data stored in the database by the Store data web page - at previous time they have been inserted by the Enterprise Application Client program (**Library2_Client2-ejb**) as the application data (p.7.8) - the view of auxiliary sequence table (to support the AUTO mechanism of generating the keys of persisted data during ORM mechanism)



7.13. The closing or before the update of Enterprise application – after undeploy process of EE project components



5. You may add annotation to your own new classes and create the proper controllers – for higher assessment (5.0 or 5.5)