

# 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

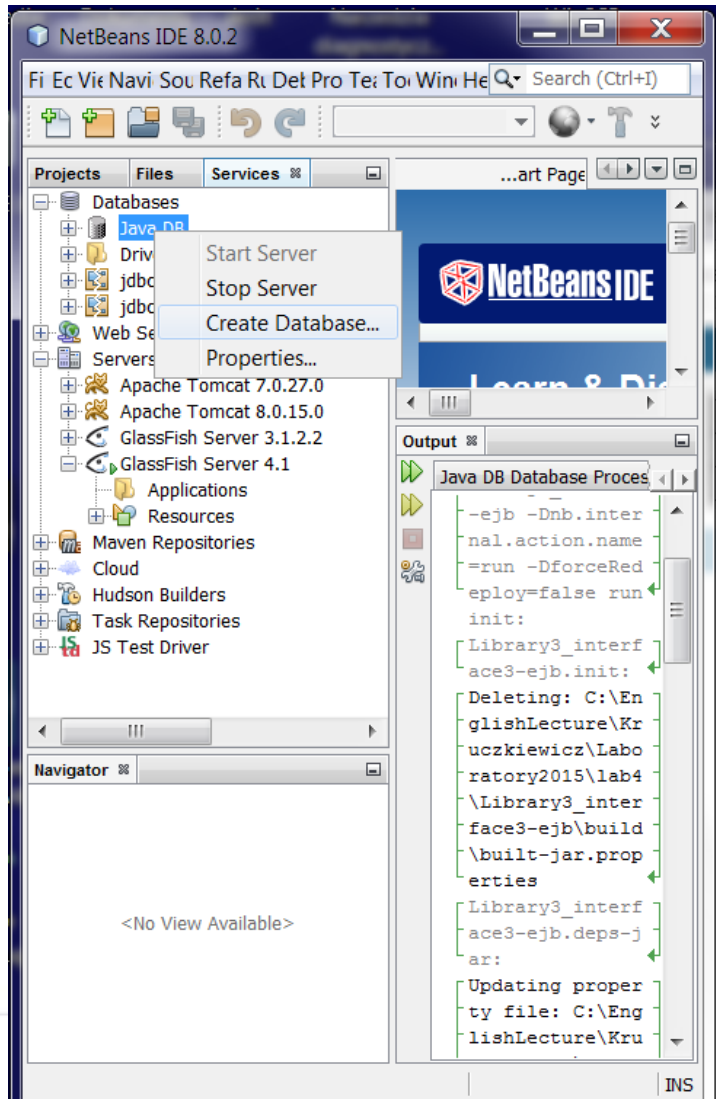


## 1. Create a database in the Derby database system

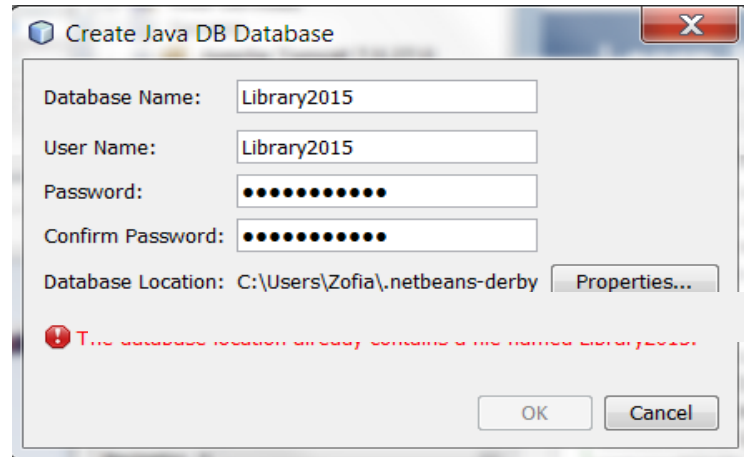
1. Select the Services tab and then expand the Databases node. Then right-click on Java DB item and select Create Database – (slide 2)
2. In the form for creating a database (slide 3), enter the data as follows: Database Name, User, Password and the Database Location (default or selected by the user). 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 catalogue 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.



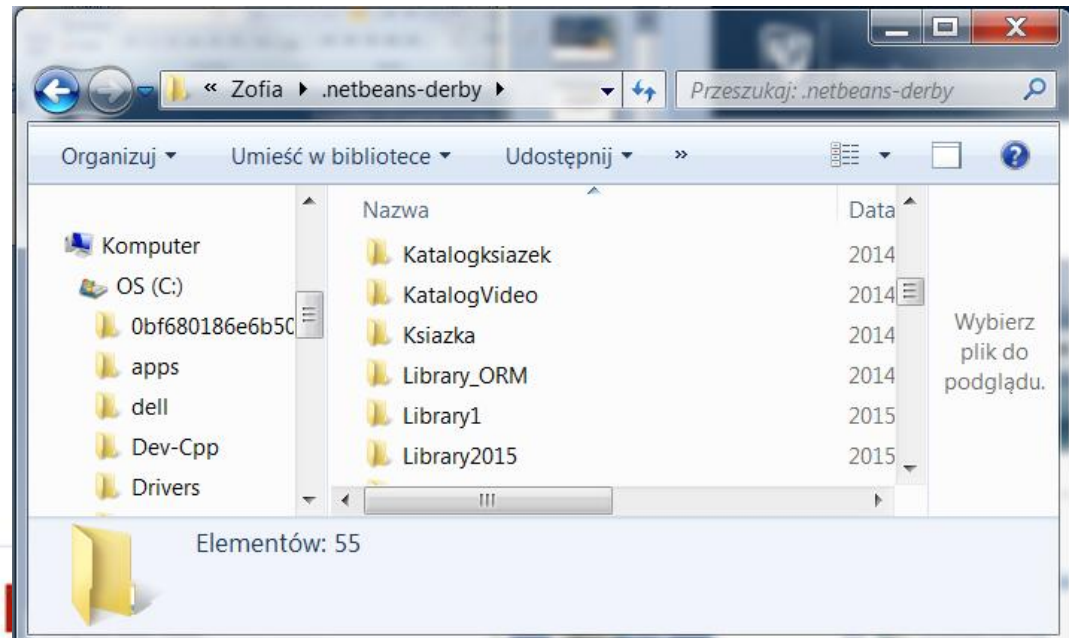
1)



2)



3)

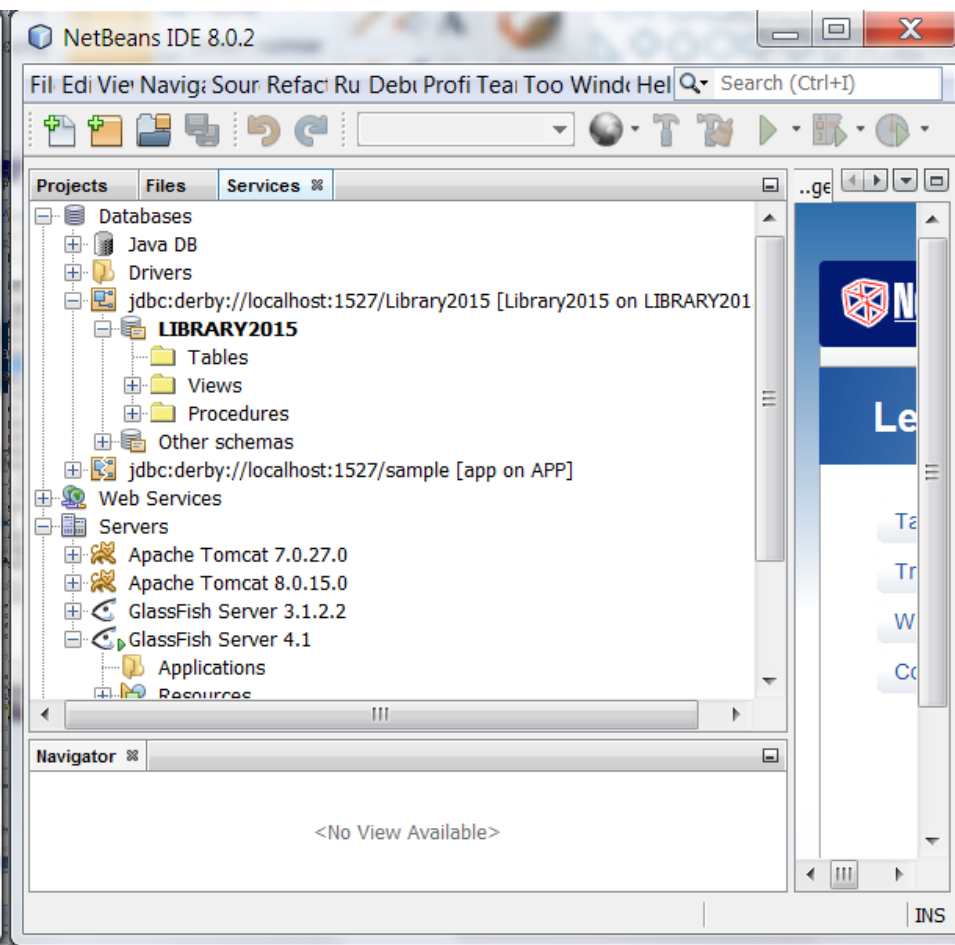
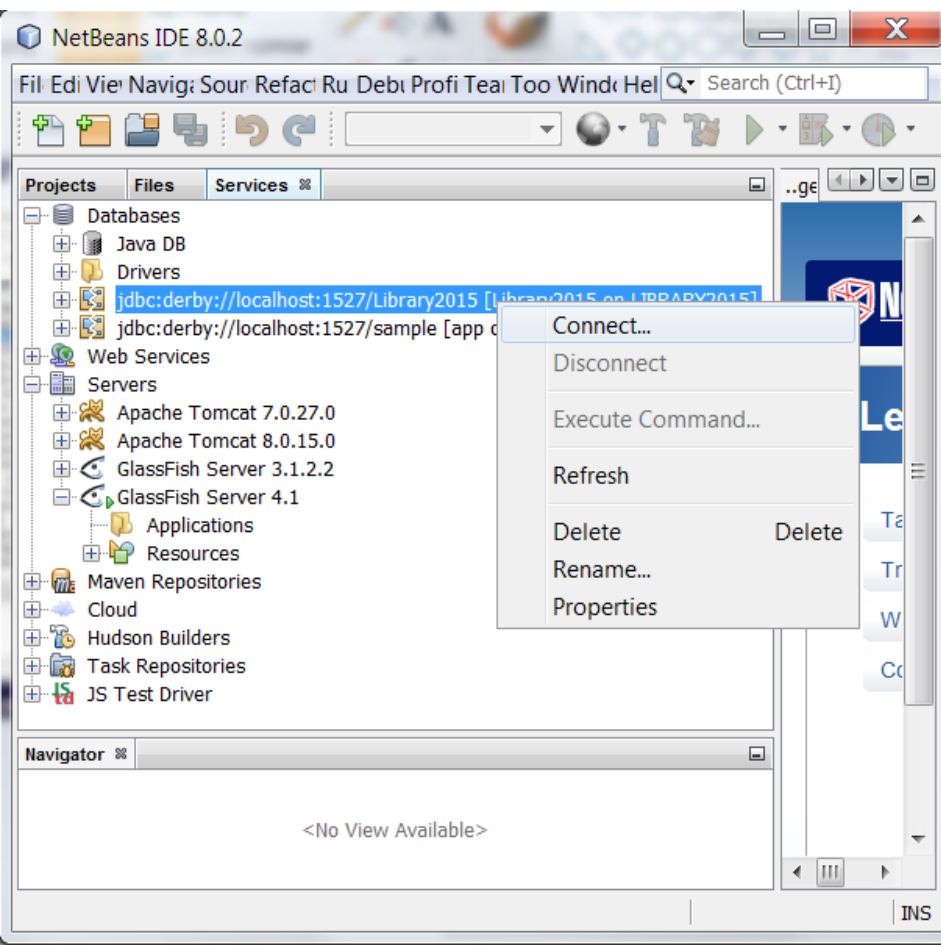






4)

5)





### 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)





### 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

The screenshot illustrates the process of creating a copy of a project in NetBeans IDE 8.0.2. It is divided into three main sections:

- Top Left:** The NetBeans IDE interface with the 'Projects' tab selected. The 'Copy...' option is highlighted in the context menu.
- Top Right:** The 'Copy Project' dialog box. The 'Project Name' is 'Library2\_JPA', the 'Project Location' is 'C:\EnglishLecture\Kruczkiewicz\Laboratory2015', and the 'Project Folder' is 'EnglishLecture\Kruczkiewicz\Laboratory2015\Library2\_JPA'. A red arrow points from the 'Copy' menu option to the 'Copy' button in the dialog. A warning message is displayed: "WARNING: This operation will not copy hidden files. If this project is under version control, ...".
- Bottom Right:** A smaller screenshot of the IDE showing the 'Library2\_JPA' project in the 'Files' tab. The 'Library2\_JPA' folder is selected in the project tree. The 'Output' window shows the execution results: "Executed successfully in 0 s. Line 1, column 1" and "Execution finished after 0 s, 0 error(s) occurred." A red arrow points from the 'Copy' button in the dialog to the 'Library2\_JPA' folder in the project tree.



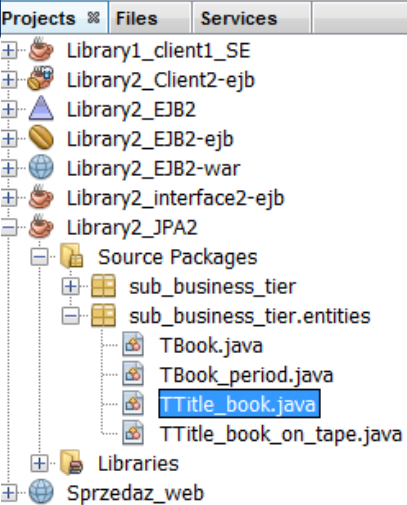


### 3.3. Add libraries of EclipseLink (JPA 2.1) technologies to the Library2\_JPA project – right click the Libraries folder, select the Add Library... item; choose the EclipseLink from GlassFish and click AddLibrary

The screenshot illustrates the process of adding external libraries to a project in NetBeans IDE 8.0.2. It is divided into three main parts:

- Left Panel:** Shows the 'Library2\_JPA' project in the 'Projects' view. The 'Libraries' folder is selected, and a context menu is open with 'Add Library...' highlighted. A red arrow points from this menu item to the 'Add Library' dialog.
- Middle Panel:** The 'Add Library' dialog is open, displaying a list of 'Available Libraries'. 'EclipseLink from GlassFish' is selected and highlighted in blue. A red arrow points from this selection to the 'Add Library' button at the bottom of the dialog.
- Right Panel:** Shows the 'Library2\_JPA2' project in the 'Projects' view. The 'Libraries' folder is expanded, showing a list of added libraries, including 'EclipseLink from GlassFish - org.eclipse.persistence.antlr.jar', 'EclipseLink from GlassFish - org.eclipse.persistence.asm.jar', 'EclipseLink from GlassFish - org.eclipse.persistence.core.jar', 'EclipseLink from GlassFish - org.eclipse.persistence.jpa.jar', 'EclipseLink from GlassFish - org.eclipse.persistence.jpa.modelgen.jar', 'EclipseLink from GlassFish - org.eclipse.persistence.moxy.jar', 'EclipseLink from GlassFish - org.eclipse.persistence.oracle.jar', 'EclipseLink from GlassFish - javax.persistence.jar', and 'JDK 1.8 (Default)'. A red arrow points from the 'Add Library' button in the middle panel to this list.



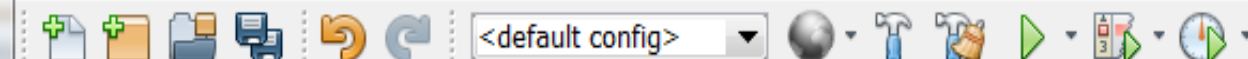


...ava TTitle\_book\_on\_tape.java TTitle\_book.java Book\_form.java Card0.jav...

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
17 @Entity
18 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
25     @Id
26     @GeneratedValue(strategy = GenerationType.AUTO)
27     private Long id;
28     public Long getId() { return id; }
29     public void setId(Long val) { id = val; }
30
31     @OneToMany(mappedBy = "mTitle_book")
32     private Collection<TBook> books;
33     public Collection<TBook> getBooks() { return books; }
34     public void setBooks(Collection<TBook> val) { books = val; }
35
36     @Transient
37     List<TBook> mBooks;
38     public List<TBook> getmBooks() { return mBooks; }
39     public void setmBooks(List<TBook> mBooks) { this.mBooks = mBooks; }
40
41     public TTitle_book() { mBooks = new ArrayList<>(); }
```

### 3.4. /3.4.1. Insert JPA annotation to the TTitle\_book class



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
5 @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
11    public String getActor() { return actor; }
12    @Override
13    public void setActor(String actor) { this.actor = actor; }
14    @Override
15    public String toString() {
16        String help = super.toString();
17        help += " Actor: " + getActor();
18        return help; }
19 }
```

### 3.4.2. Insert JPA annotation to the TTitle\_book\_on\_tape class

toString - Navigator

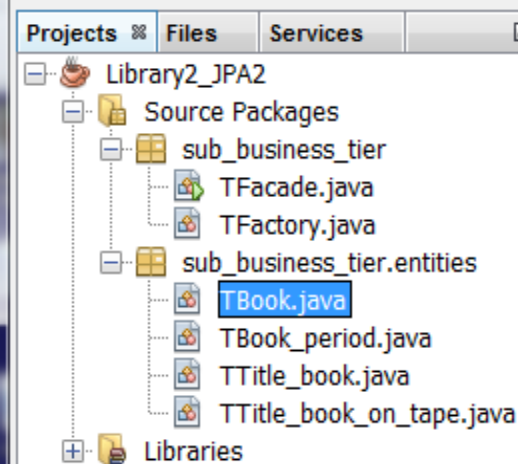
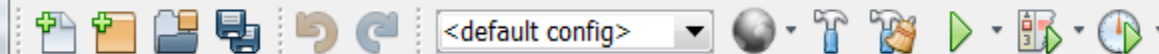
Members &lt;empty&gt;

toString() : String



Output

&gt;&gt; GlassFish Server 4.1 Java DB Database Process Library3\_client3-ejb (run)



...avz TTitle\_book\_on\_tape.java TBook.java

Source History

```

1 package sub_business_tier.entities;
2
3 import java.io.Serializable;
4 import java.util.Date;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9 import javax.persistence.ManyToOne;
10

```

### 3.4.3. Insert JPA annotation to the TBook class

```

11 @Entity
12 public class TBook implements Serializable {
13
14     private static final long serialVersionUID = 1L;
15     private int number;
16
17     @ManyToOne
18     private TTitle_book mTitle_book;
19     public TTitle_book getmTitle_book() {
20         return mTitle_book; }
21     public void setmTitle_book(TTitle_book mTitle_book) {
22         this.mTitle_book = mTitle_book; }
23
24     @Id
25     @GeneratedValue(strategy = GenerationType.AUTO)
26     private Long id;
27     public void setId(Long id) { this.id = id; }
28     public Long getId() { return id; }
29     public TBook () { }
30

```

hashCode - Navigator

Members <empty>

hashCode() : int  
period\_pass(Object data) : boolean

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

period - Navigator

Members <empty>

- TBook\_period :: TBook
  - getPeriod() : Date
  - period\_pass(Object data) : boo
  - setPeriod(Date date)
  - startPeriod(Object data)
  - toString() : String
  - period : Date
  - serialVersionUID : long

```

Source | History
1 package sub_business_tier.entities;
2
3 import java.util.Date;
4 import javax.persistence.Entity;
5 import javax.persistence.Temporal;
6 import sub_business_tier.TFactory;
7
8 @Entity
9 public class TBook_period extends TBook {
10
11     private static final long serialVersionUID = 1L;
12
13     @Temporal(javax.persistence.TemporalType.DATE)
14     private Date period;
15
16     @Override
17     public Date getPeriod() {
18         return period;
19     }
20
21     @Override
22     public void setPeriod(Date date) {
23         period = date;
24     }
  
```

### 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...

The screenshot shows the NetBeans IDE 8.0.2 interface. The 'File' menu is open, and 'New Project...' is selected. The main editor window displays the code for 'TBook\_period.java'. The code includes imports for 'java.util.Date', 'javax.persistence.Entity', and 'javax.persistence.Temporal', and a class definition for 'TBook\_period' extending 'TBook'. The 'Run' button is visible in the toolbar, and the 'Library2\_JPA2 (run)' console window is open at the bottom.

```

sub_business_tier.entities;

ava.util.Date;
avax.persistence.Entity;
avax.persistence.Temporal;
sub_business_tier.TFactory;

class TBook_period extends TBook {

    static final long serialVersionUID = 1L;

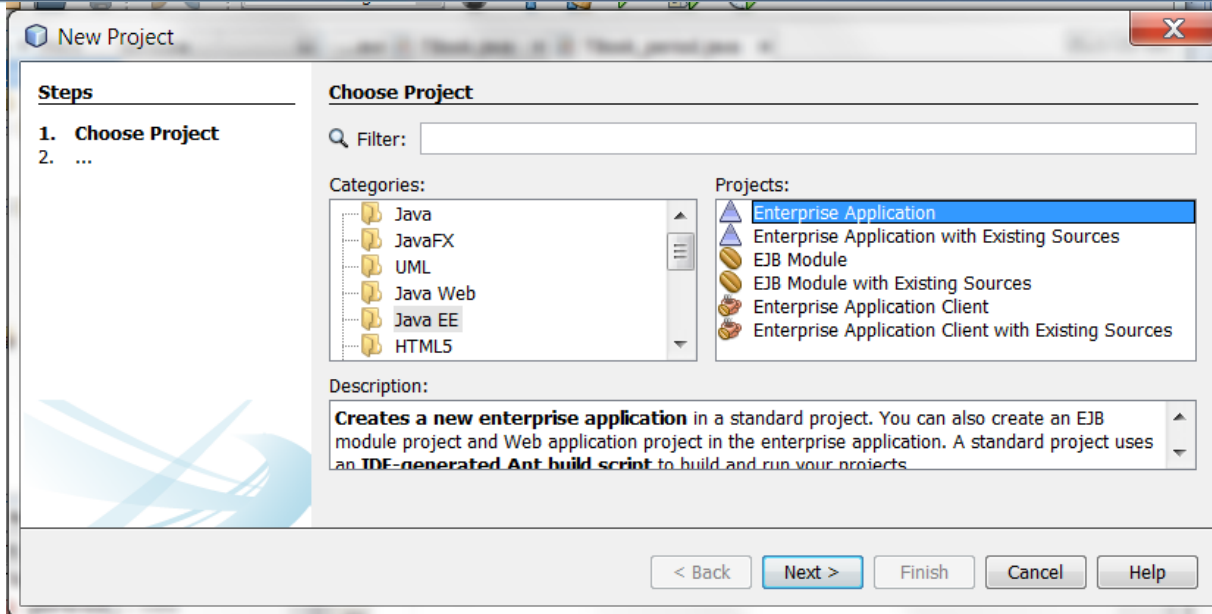
    @Temporal(javax.persistence.TemporalType.DATE)
    private Date period;

    @Override
    public Date getPeriod() {
        return period;
    }

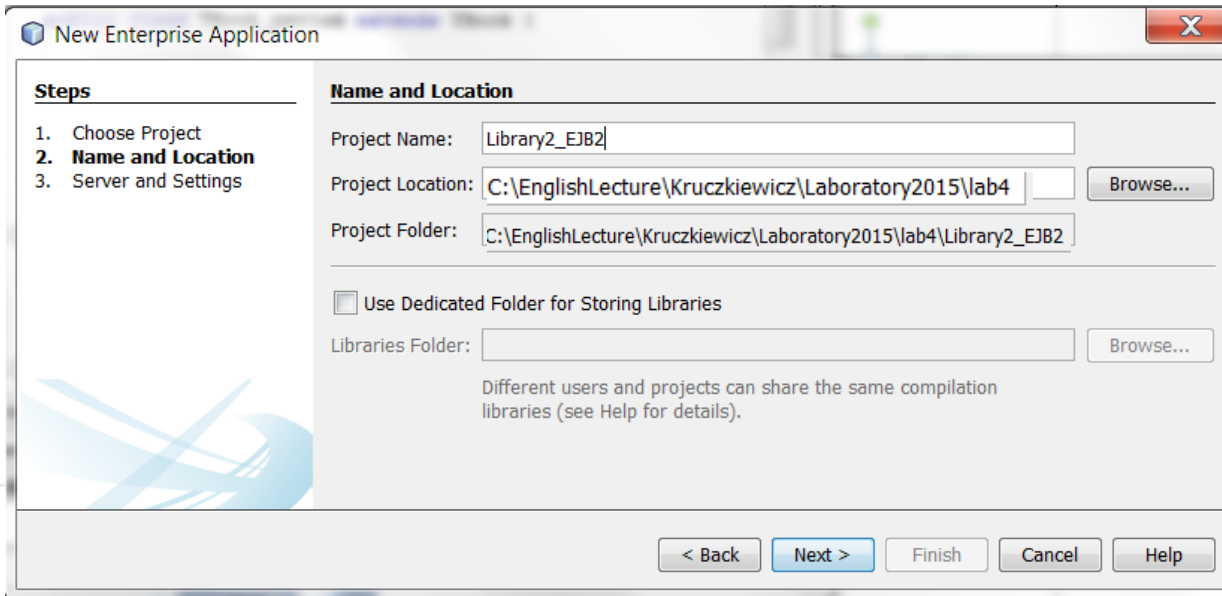
    @Override
    public void setPeriod(Date date) {
        period = date;
    }
}

```





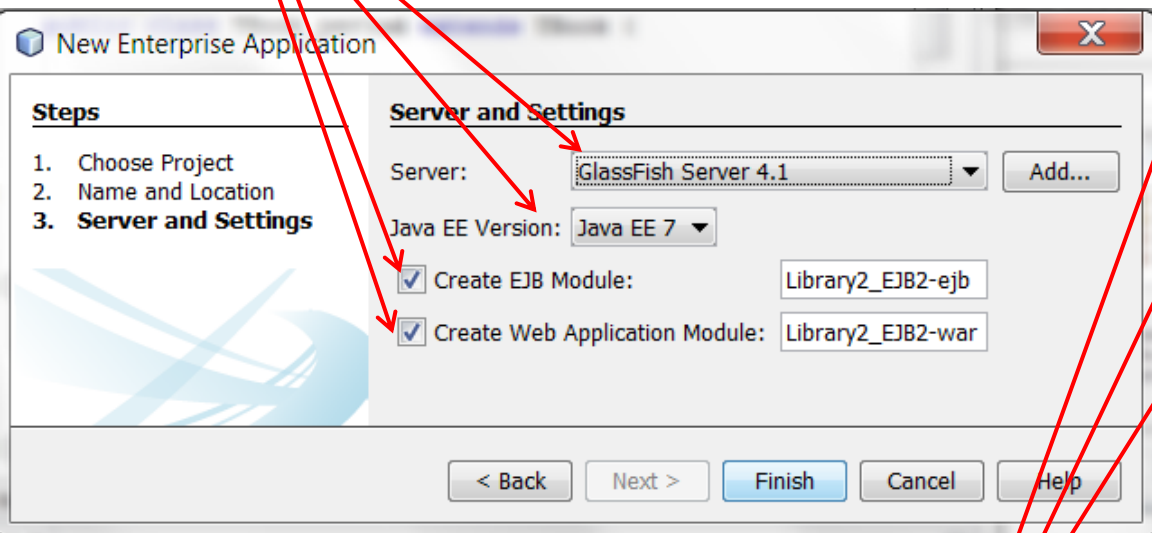
#### 4.1.2. Select the Java EE/ Enterprise Application options



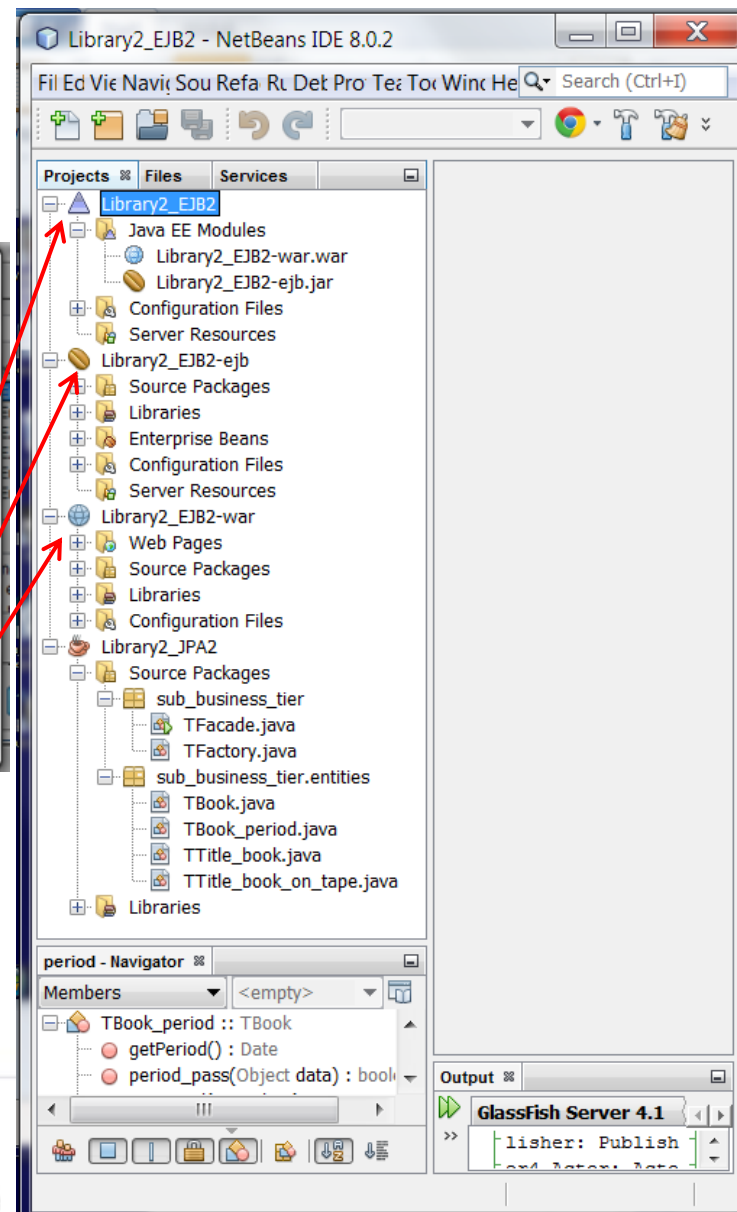
#### 4.1.3. Fill the Library2\_EJB2 name of Java EE project and select location of this project



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

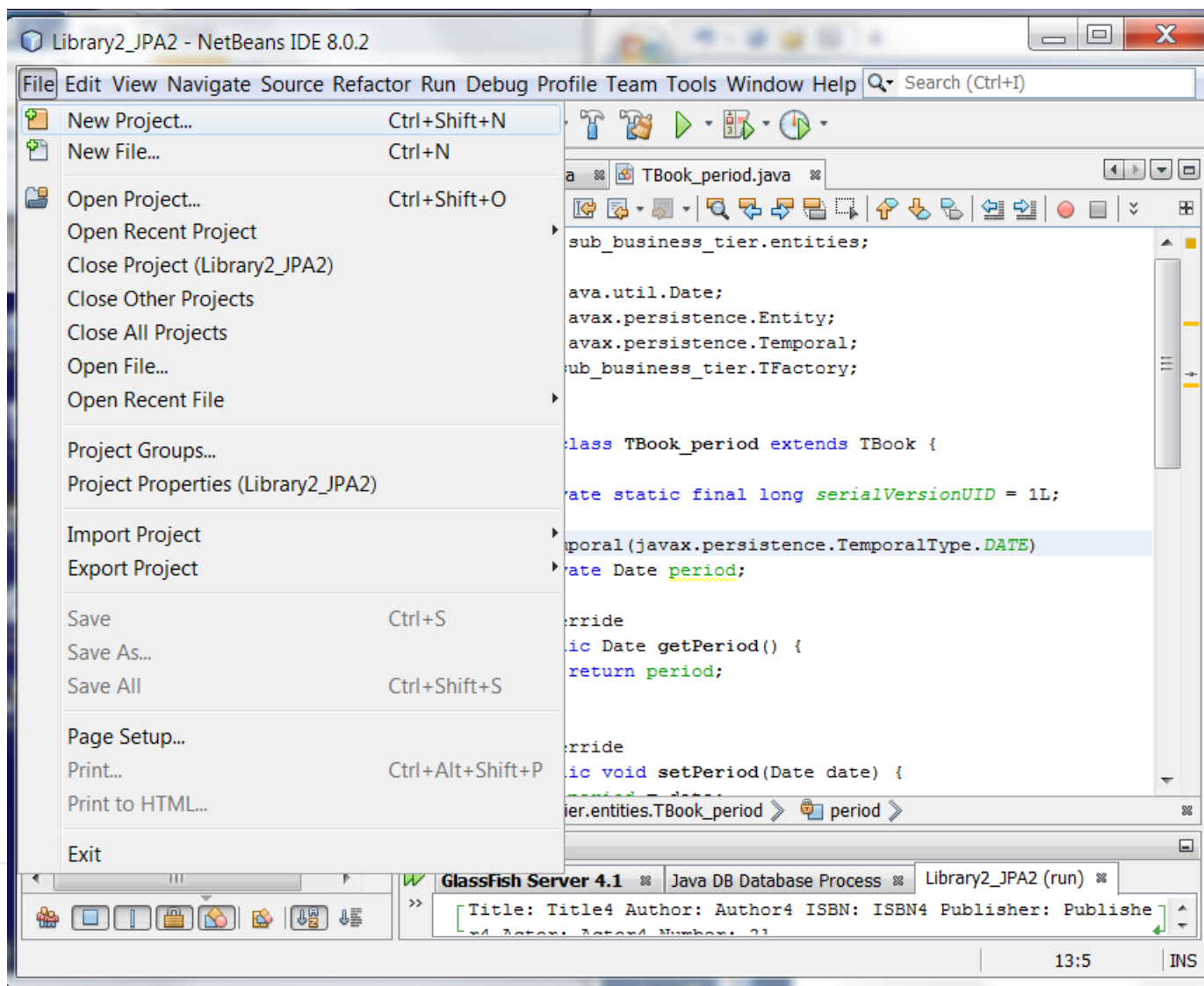


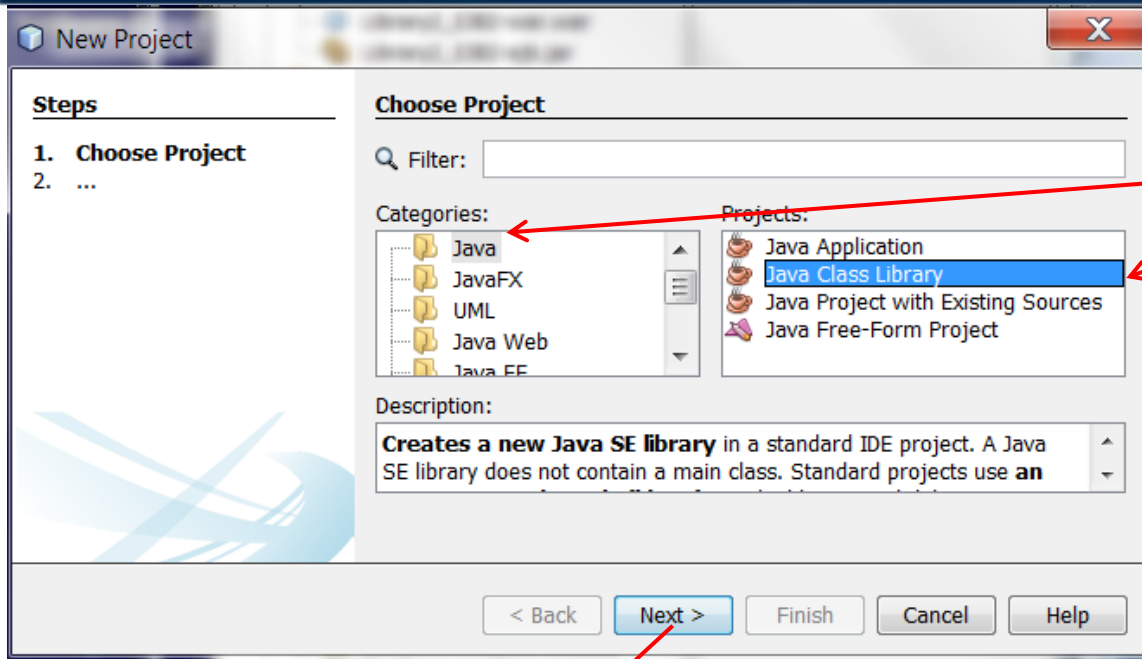
**4.1.5. Result**



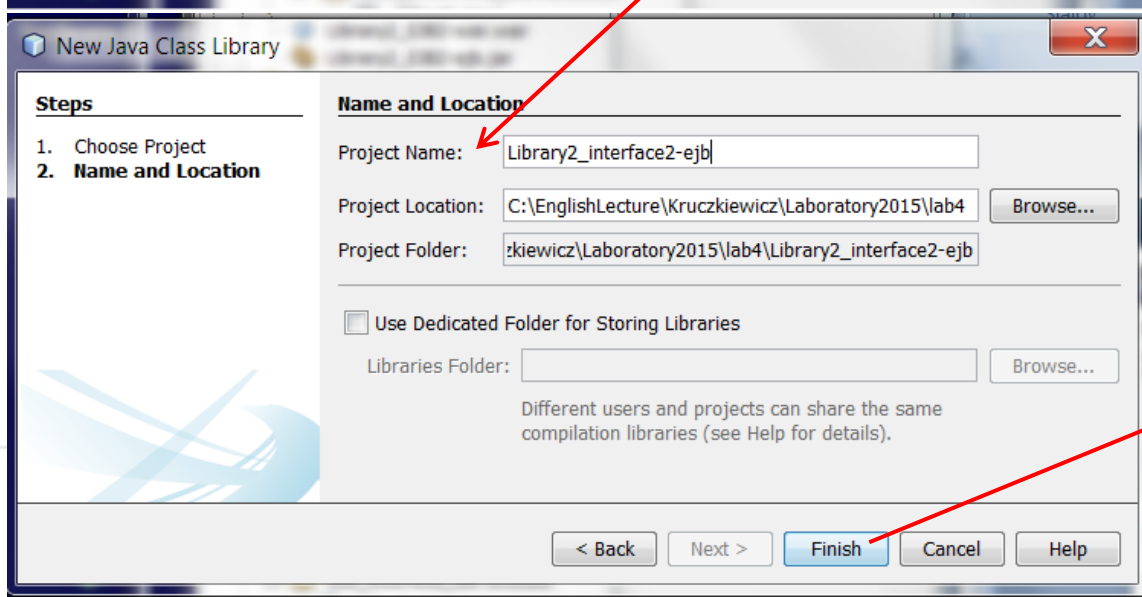


**4.2./4.2.1. Create the Java Class Library project as the interface for remote Session Bean in the Library2\_EJB2-ejb module of the Library2\_EJB2 Enterprise Application project – click File/New Project...**

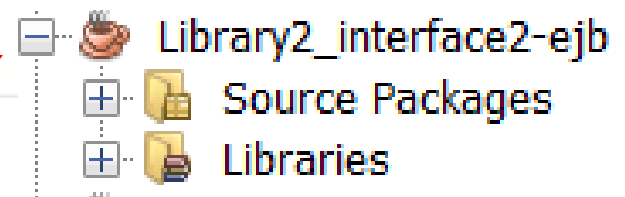




4.2.2. Select the Java/ Java Class Library options and click Next



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**

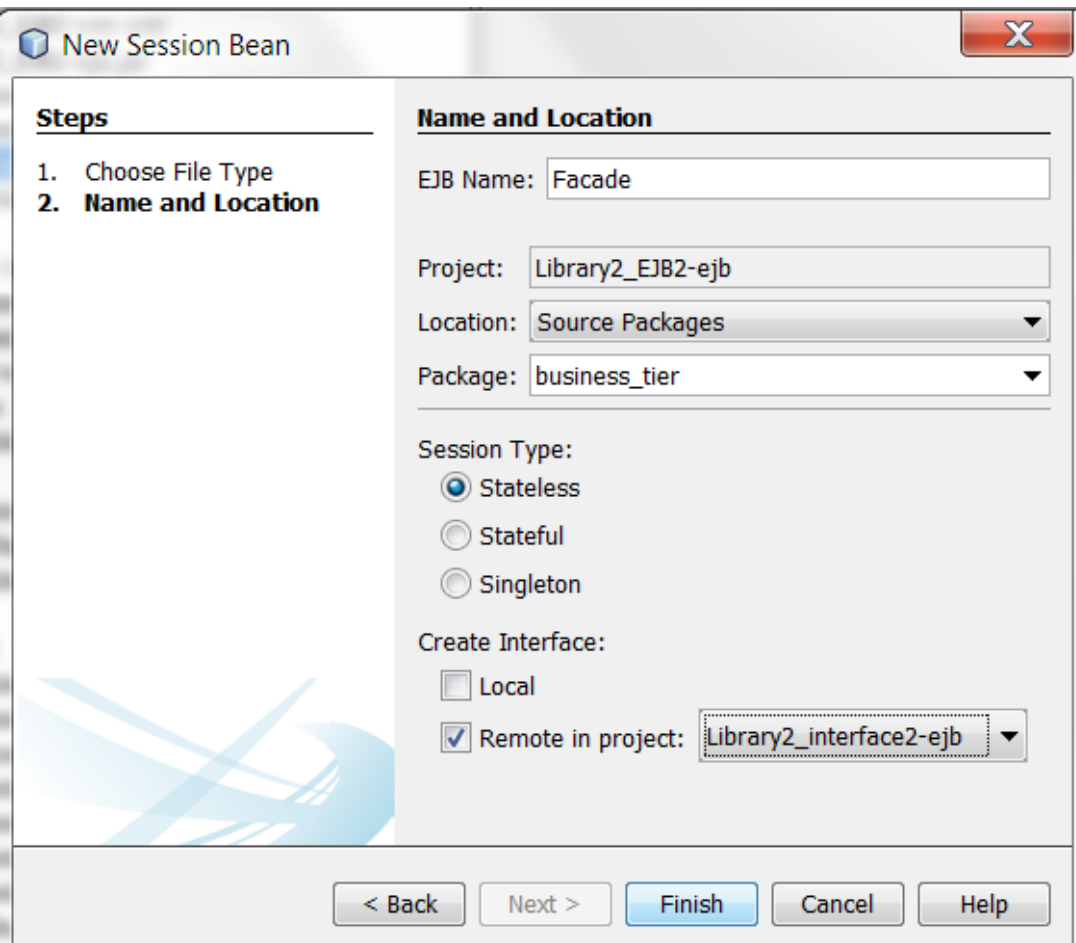
The screenshot shows the NetBeans IDE 8.0.2 interface. On the left, the 'Projects' tab is active, showing a tree view of the 'Library2\_EJB2' project. The 'Library2\_EJB2-ejb' module is selected, and a context menu is open with 'New' and 'Other...' options highlighted. A red arrow points from the 'New' option in the context menu to the 'New File' dialog. The 'New File' dialog is open, showing the 'Choose File Type' step. The 'Project' is set to 'Library2\_EJB2-ejb'. The 'File Types' list is expanded, and 'Session Bean' is selected. A red arrow points from the 'Session Bean' option in the 'File Types' list to the 'Next >' button. The 'Description' field contains the text: 'Creates an empty Session Enterprise JavaBean (EJB) component. A session bean is typically used to encapsulate business logic or enterprise resources. This template creates the Java classes for a single session bean and...'. The 'Next >' button is highlighted.





**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**



**New Session Bean**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

EJB Name: Facade

Project: Library2\_EJB2-ejb

Location: Source Packages

Package: business\_tier

Session Type:

Stateless

Stateful

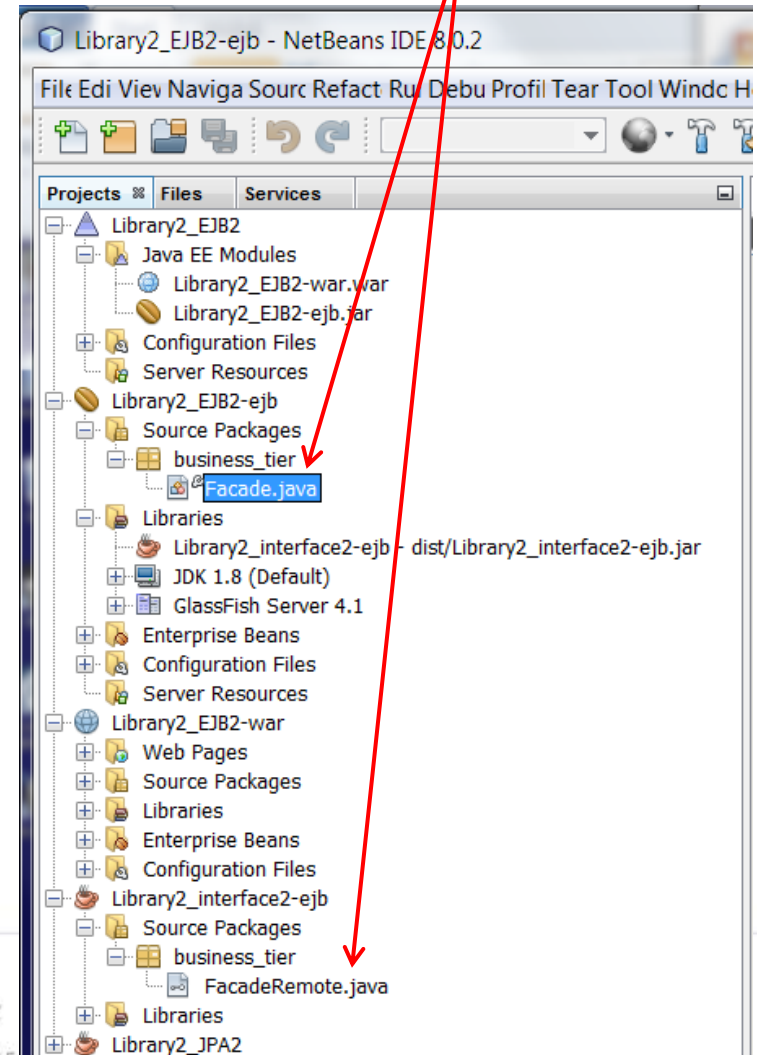
Singleton

Create Interface:

Local

Remote in project: Library2\_interface2-ejb

< Back   Next >   Finish   Cancel   Help



#### 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

The screenshot shows the NetBeans IDE 8.0.2 interface. The main window displays the 'Library2\_EJB2-ejb' project. The 'Libraries' folder is selected, and a context menu is open with 'Add Project...' chosen. The 'Add Project' dialog box is open, showing the 'Laboratory2015' directory. The 'Library2\_JPA2' project is selected in the file list. The 'Project Name' is 'Library2\_JPA2' and the 'Project JAR Files' is 'dist/Library2\_JPA2.jar'. The 'File name' is 'wicz\Laboratory2015\lab4\Lib' and the 'Files of type' is 'Project Folder'. The 'Add Project JAR Files' button is highlighted. A red arrow points from the 'Add Project...' menu item to the 'Add Project JAR Files' button.

The 'Add Project' dialog box shows the following details:

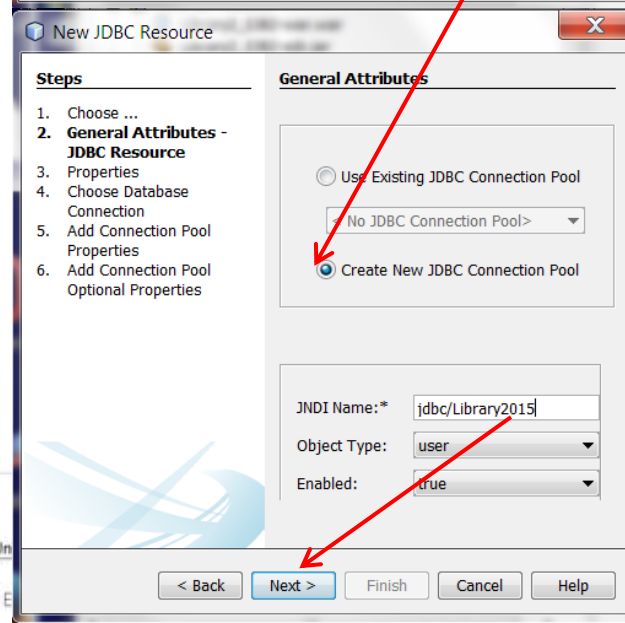
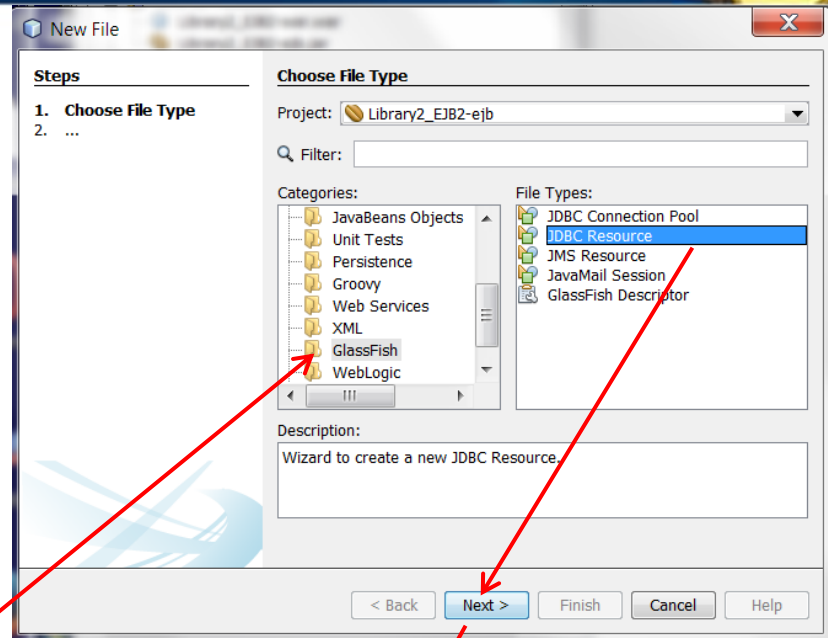
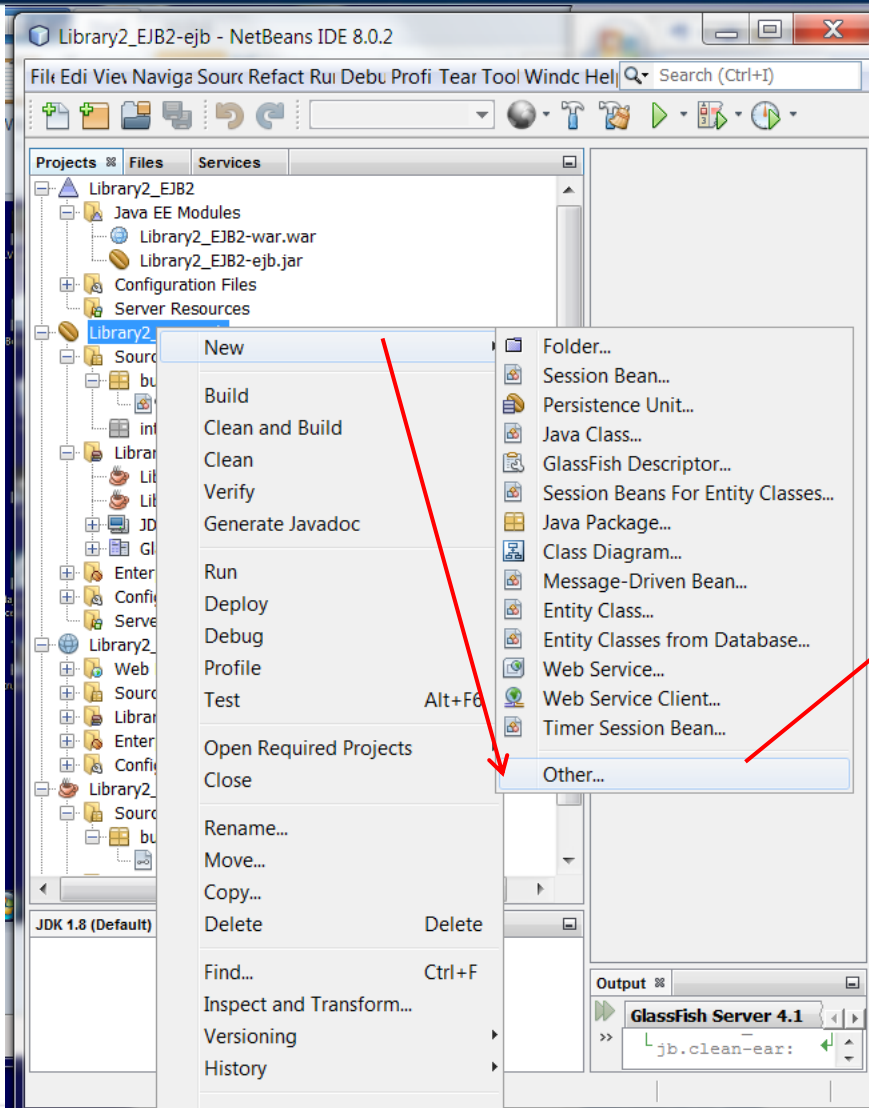
- Look in: Laboratory2015
- Project Name: Library2\_JPA2
- Project JAR Files: dist/Library2\_JPA2.jar
- File name: wicz\Laboratory2015\lab4\Lib
- Files of type: Project Folder
- Buttons: Add Project JAR Files, Cancel

The 'Libraries' folder in the project tree contains the following items:

- Library2\_interface2-ejb - dist/Library2\_interface2-ejb.jar
- Library2\_JPA2 - dist/Library2\_JPA2.jar

```
@Stateless
public class FacadeRemote {
    // Add business logic here
    // "Insert"
}
```

## 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

**Steps**

1. Choose ...
2. General Attributes - JDBC Resource
3. Properties
- 4. Choose Database Connection**
5. Add Connection Pool Properties
6. Add Connection Pool Optional Properties

**Choose Database Connection**

Provide configuration information for the JDBC Connection Pool.  
Either choose an existing database connection to extract information, or enter the configuration information.  
Fields with an \* mark are required.

JDBC Connection Pool Name: \*

Extract from Existing Connection:

New Configuration using Database:

XA (Global Transaction)

**Steps**

1. Choose ...
2. General Attributes - JDBC Resource
3. Properties
4. Choose Database Connection
- 5. Add Connection Pool Properties**
6. Add Connection Pool Optional Properties

**Add Connection Pool Properties**

Enter the Datasource Classname, URL, and User to continue.  
Hit the Enter key to save values in the Properties table.

Datasource Classname:

Resource Type:

Description:

Properties:

Name	Value
URL	jdbc:derby://localhost:1527/Library2015
serverName	localhost
PortNumber	1527
DatabaseName	Library2015

**Steps**

1. Choose ...
2. General Attributes - JDBC Resource
3. Properties
4. Choose Database Connection
5. Add Connection Pool Properties
- 6. Add Connection Pool Optional Properties**

**Specify Optional Properties for Connection Pool**

**Pool Settings**

Steady Pool Size:

Max Pool Size:

Max Wait Time:

Pool Resize Quantity:

Idle Timeout (secs):

**Transaction Isolation**

Transaction Isolation:

Guarantee Isolation Level:

**Connection Validation**

Connection Validation Required:

Validation Method:

Table Name:

Fail All Connections:

Non Transactional Connections:

Allow Non Component Callers:





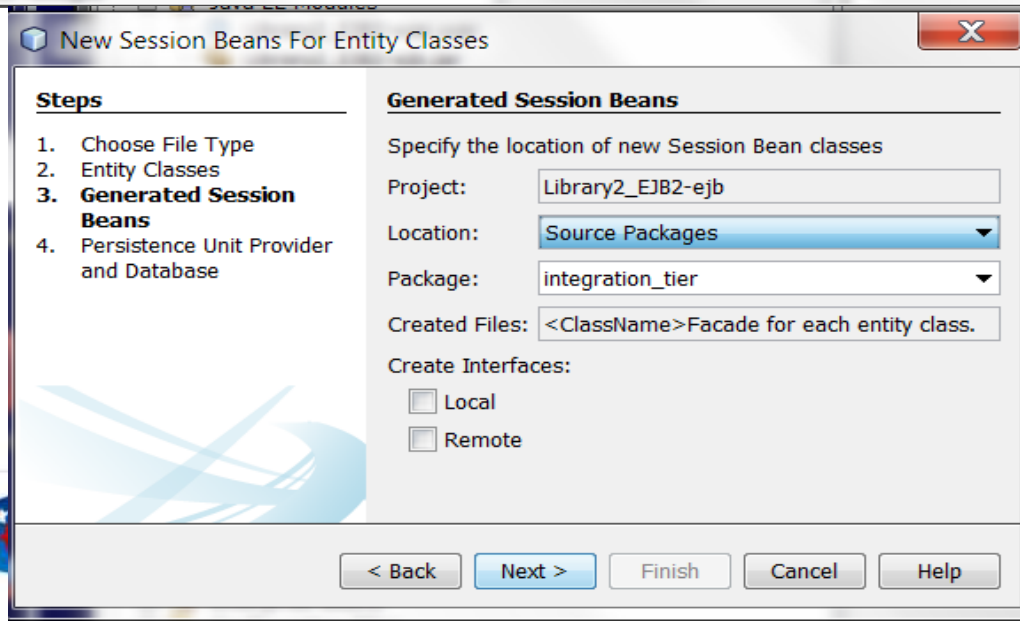
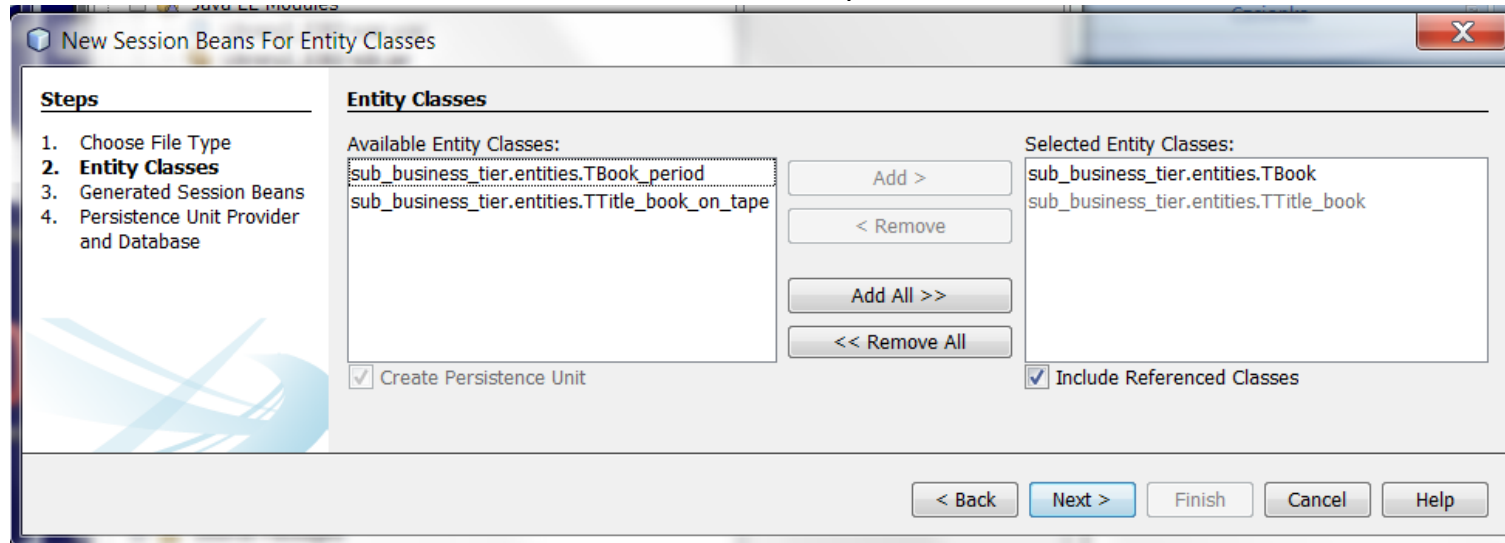
**4.6./4.6.1. Create the local Session Bean for Entity Classes in the Library2\_EJB2-ejb Enterprise Application project – right click the Library2\_EJB2-ejb module in the Projects tab and select the New/Other items; select the Enterprise JavaBeans/Session Bean For Entity Classes options and click Next**

The screenshot shows the NetBeans IDE interface. On the left, the 'Projects' tab displays the project structure for 'Library2\_EJB2'. The 'Library2\_EJB2-ejb' module is selected, and a context menu is open with 'New' > 'Other...' selected. The 'New File' dialog is open, showing the 'Choose File Type' step. The 'Enterprise JavaBeans' category is expanded, and 'Session Beans For Entity Classes' is selected. The 'Description' field contains the text: 'Creates Session EJB Facade based on Entity class. This template creates Session EJB for every Entity class with basic access methods.' The 'Next >' button is highlighted.



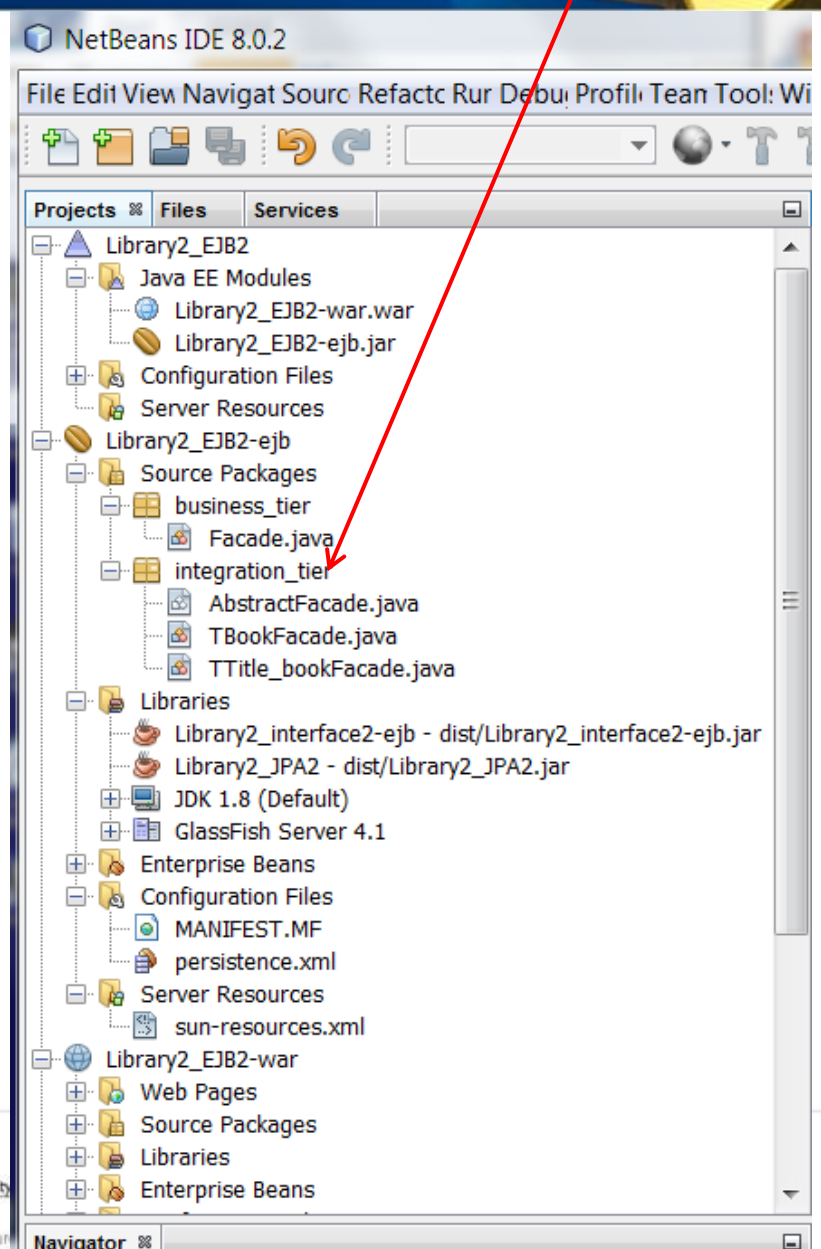
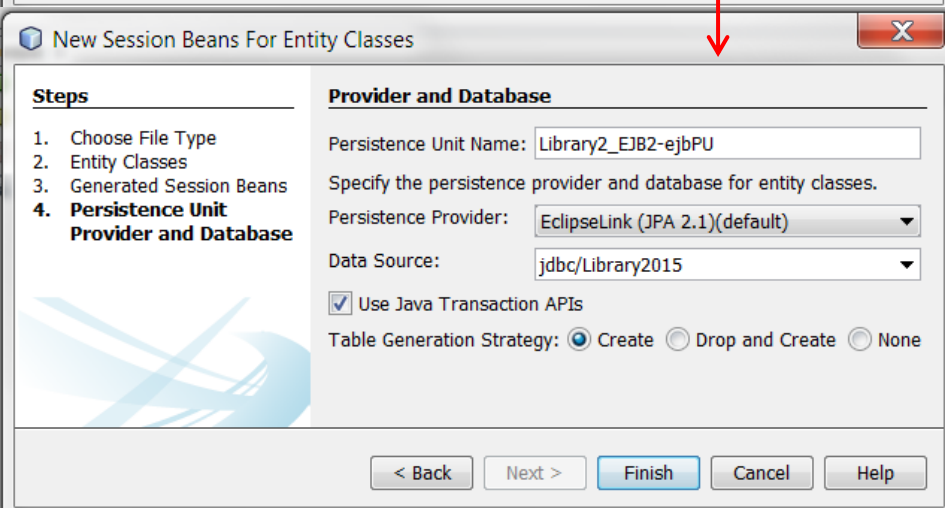
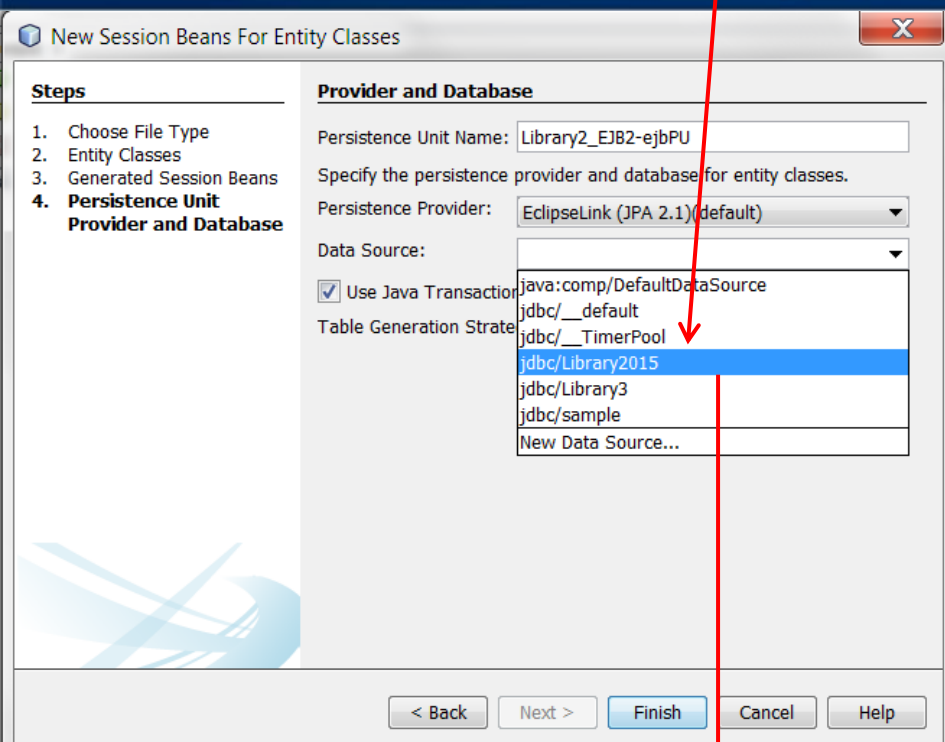


**4.6.2. Continuation** – select the TTitle\_book and TBook entity classes and add to the right with the selected Include Referenced Classes option and click Next



**4.6.3. Continuation** –fill the field Package with the integration\_tier name of the new package for created beans and click Next

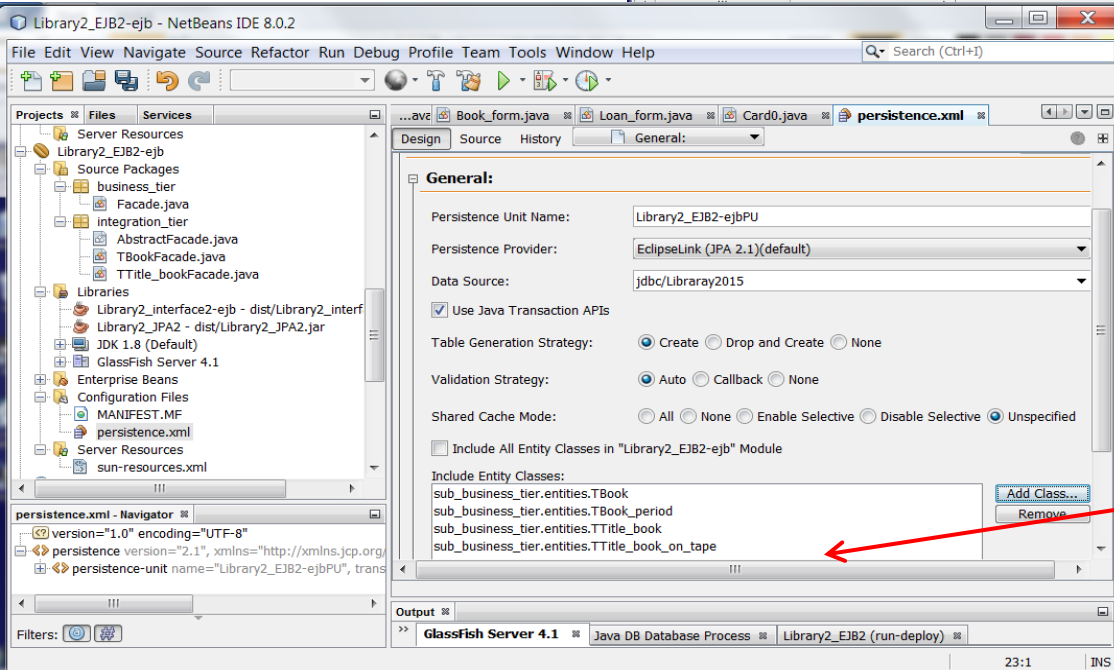
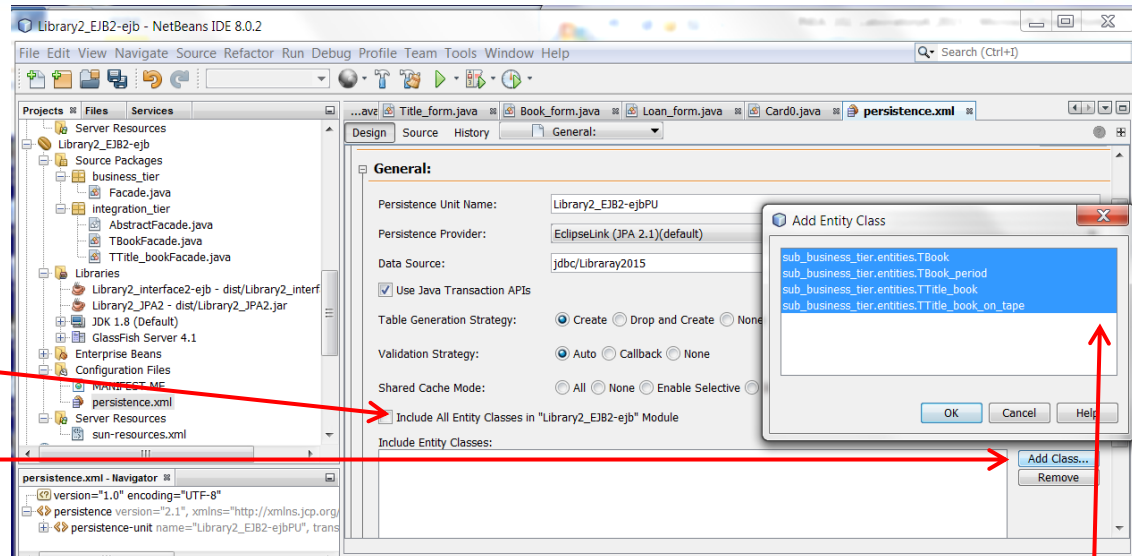
#### 4.6.4. Continuation (Persistence Unit Provider and Database) – select the DataSource created during the 4.5 step and click Finish. The result is shown on the right.





### 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





```

package integration_tier;

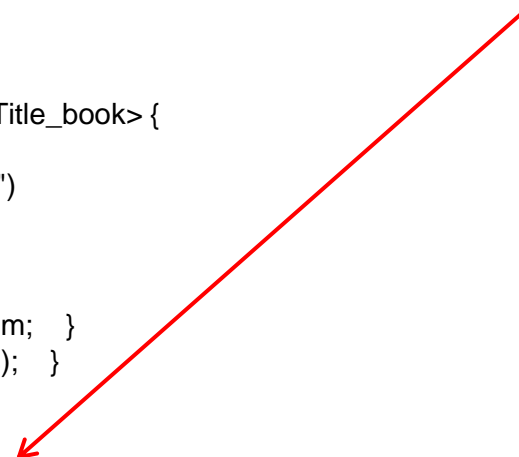
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import sub_business_tier.entities.TTitle_book;
@Stateless
public class TTitle_bookFacade extends AbstractFacade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

@Stateless
public class TBookFacade extends AbstractFacade<TBook> {

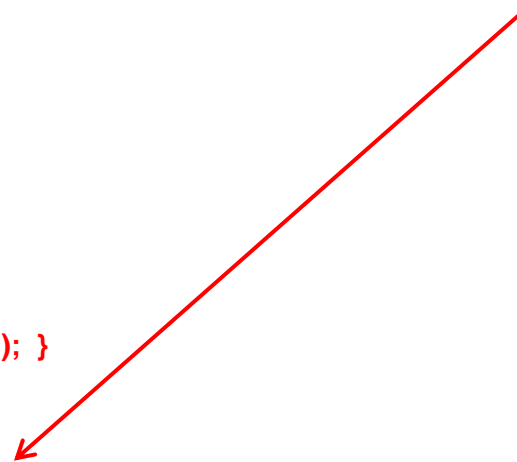
    @PersistenceContext(unitName = "Library2_EJB2-ejbPU")
    private EntityManager em;

    @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);    }
            }
        }
    }
}

```

#### 4.8.2. Update generated code of TBookFacade Session Bean for TBook and TBook\_period entity classes, generated in the 4.6 steps







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

Library2\_EJB2-ejb - NetBeans IDE 8.0.2

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

Search (Ctrl+I)

Projects Files Services

Library2\_EJB2

Java EE Modules

Library2\_EJB2-war.war

Library2\_EJB2-ejb.jar

Configuration Files

Server Resources

Library2\_EJB2-ejb

Source Packages

business\_tier

Facade.java

integration\_tier

AbstractFacade.java

TBookFacade.java

TTitle\_bookFacade.java

Libraries

Library2\_interface2-ejb - dist/Library2\_interface2-ejb.jar

Library2\_JPA2 - dist/Library2\_JPA2.jar

JDK 1.8 (Default)

GlassFish Server 4.1

Enterprise Beans

Configuration Files

MANIFEST.MF

persistence.xml

Server Resources

sun-resources.xml

Library2\_EJB2-war

Web Pages

Source Packages

Libraries

Enterprise Beans

getTBooks\_ - Navigator

Members

<empty>

getEntityManager(): EntityManager

getTBooks(): TBook[]

Source

```

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);
            }
        }
    }
}

```

integration\_tier.TBookFacade > getTBooks\_ >

Output

GlassFish Server 4.1 Java DB Database Process Library2\_EJB2 (run-deploy)

35:36 INS

Select the fully qualified name to use in the import statement.

Import Statements:

TTitle\_book sub\_business\_tier.entities.TTitle\_book

Iterator java.util.Iterator

List java.util.List

Remove unused imports

Library2\_EJB2-ejb - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run De

OK Cancel

### 4.8.4. The result

Projects Files Services

Library2\_EJB2

Java EE Modules

Library2\_EJB2-war.war

Library2\_EJB2-ejb.jar

Configuration Files

Server Resources

Library2\_EJB2-ejb

Source Packages

business\_tier

Facade.java

integration\_tier

AbstractFacade.java

TBookFacade.java

TTitle\_bookFacade.java

Libraries

Library2\_interface2-ejb - dist/Library2\_inte

Library2\_JPA2 - dist/Library2\_JPA2.jar

JDK 1.8 (Default)

GlassFish Server 4.1

Enterprise Beans

Configuration Files

MANIFEST.MF

persistence.xml

Server Resources

sun-resources.xml



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

The screenshot shows the NetBeans IDE interface. The main editor displays the source code of `TTtitle_bookFacade.java`. A `Fix All Imports` dialog box is open, showing a list of import statements: `java.util.Iterator` and `java.util.List`. The `Remove unused imports` checkbox is checked. The `OK` button is highlighted. A red arrow points from the text '4.8.5. Right click and select the Fix Imports.. item. In the form of Fix Imports you must select the shown imports.' to the dialog box. Another red arrow points from the text '4.8.6. The result' to the `TTtitle_bookFacade.java` file in the project tree on the right.

```
19 @PersistenceContext(unitName = "Library2_EJB2-ejbPU")
20 private EntityManager em;
21
22 @Override
23 protected EntityManager getEntityManager() {
24     return em;
25 }
26
27 public TTtitle_bookFacade() {
28     super(TTtitle_book.class);
29 }
30 //-----
31
32 public TTtitle_book[] getTTtitle_books_() {
33     return findAll().toArray(new TTtitle_book[0]);
34 }
35
36 public void addTTtitle_books(List<TTtitle_book> titles) {
37     TTtitle_book newTTtitle_book=null;
38     Iterator<TTtitle_book> it = titles.iterator();
39     while (it.hasNext()) {
40         newTTtitle_book = it.next();
41         if (newTTtitle_book.getId() == null) {
42             getEntityManager().persist(newTTtitle_book);
43         }
44     }
45 }
46 }
```

### 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;

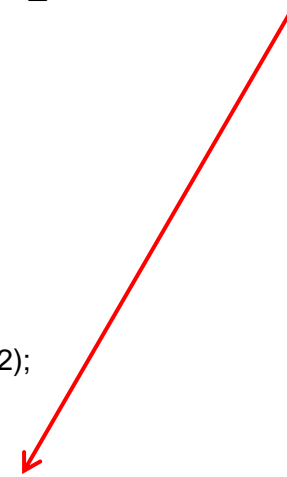
    public void update_books() throws Exception;

    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;
}
```





The screenshot shows the NetBeans IDE interface. The main editor window displays the code for `FacadeRemote.java`. The code is as follows:

```
1 package business_tier;
2
3 import java.util.ArrayList;
4 import javax.ejb.Remote;
5
6 @Remote
7 public interface FacadeRemote {
8
9     public Object[][] gettitle_books();
10    public String add_title_book(String data[]);
11    public ArrayList<String> add_book(String data1[], String data2[]);
12    public ArrayList<String> Search_title_book(String data[]);
13    public String Search_accessible_book(String data1[], Object data2);
14    //.....
15
16    public void init();
17    public void update_titles() throws Exception;
18    public void update_books() throws Exception;
19    public void update_data() throws Exception;
20    public void add_titles() throws Exception;
21    public void add_books() throws Exception;
22    public ArrayList<ArrayList<String>> titles() throws Exception;
23 }
```

A red arrow points from the text on the right to the `import java.util.ArrayList;` line in the code editor.

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.**

The screenshot shows the NetBeans IDE 8.0.2 interface. The main editor displays the code for `Facade.java` in the `business_tier` package. A right-click context menu is open over the code, with `Insert Code...` selected. A secondary menu is open over `Insert Code...`, with `Call Enterprise Bean...` selected. A third dialog box, `Call Enterprise Bean`, is open, showing a tree view of the project structure. The `TBookFacade` entity is selected in the tree. The `Reference Name` field is set to `TBookFacade`, and the `Referenced Interface` is set to `No interface`. The `OK` button is highlighted.

```
package business_tier;

import javax.ejb.Stateless;

@Stateless
public class Facade implements FacadeRemote {

    // Add business logic below (Right-click in editor)
}
```

Generate

- Add Business Method...
- Constructor...
- Logger...
- toString()...
- Implement Method...
- Override Method...
- Add Property...
- Use Entity Manager...
- Call Enterprise Bean...
- Use Database...
- Send JMS Message...
- Send E-mail...
- Call Web Service Operation...
- Generate REST Client...

Insert Code... Alt+Insert

Fix Imports Ctrl+Shift+I

Refactor

Format Alt+Shift+F

Run File Shift+F6

Debug File Ctrl+Shift+F5

Test File Ctrl+F6

Debug Test File Ctrl+Shift+F6

Run Focused Test Method

Debug Focused Test Method

Run Into Method

New Watch... Ctrl+Shift+F7

Call Enterprise Bean

Select an enterprise bean from open projects.

- Library2\_EJB2-war
- Library2\_EJB2-ejb
  - Facade
    - TBookFacade
    - TTitle\_bookFacade

Reference Name: TBookFacade

Referenced Interface:  No interface  Local  Remote

OK Cancel Help





**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.

The screenshot shows the NetBeans IDE interface with the following components:

- Project Explorer:** Shows the project structure for 'Library2\_EJB2-ejb', including source packages like 'business\_tier' and 'integration\_tier', and libraries.
- Code Editor:** Displays the code for 'Facade.java'. The code includes imports for 'TBookFacade', 'EJB', and 'Stateless', and defines a 'Facade' class that implements 'FacadeRemote'. A red arrow points to the code editor area.
- Context Menu:** A right-click context menu is open over the code editor. The 'Insert Code...' option is selected, and a sub-menu is visible showing 'Call Enterprise Bean...' as the chosen option. A red arrow points from the 'Call Enterprise Bean...' option in the sub-menu to the 'Call Enterprise Bean...' option in the main context menu.
- Call Enterprise Bean Dialog:** A dialog box titled 'Call Enterprise Bean' is open. It prompts the user to 'Select an enterprise bean from open projects.' The 'TTitle\_bookFacade' option is selected in the list. The 'Reference Name' field is filled with 'TTitle\_bookFacade'. The 'Referenced Interface' is set to 'No interface'. A red arrow points from the 'Call Enterprise Bean...' option in the context menu to this dialog box.



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 initialization the content of the mTitle\_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);  
                }  
            }  
        }  
    }  
}
```

```
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) { }  
    }
```

**The new method for initial activities of Session Bean**

```
    @Override  
    public Object[][] gettitle_books()  
    @Override  
    public String add_title_book(String data[])  
    @Override  
    public ArrayList<String> add_book(String data1[], String data2[])  
    @Override  
    public ArrayList<String> Search_title_book(String data[])  
    @Override  
    public String Search_accessible_book(String data1[], Object data2)
```

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

```
{ return facade.gettitle_books(); }  
{ return facade.add_title_book(data); }  
{ return facade.add_book(data1, data2); }  
{ return facade.Search_title_book(data); }  
{ return facade.Search_accessible_book(data1, data2); }
```




#### 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. 

Fix All Imports

Select the fully qualified name to use in the import statement.

Import Statements:

TBook	sub_business_tier.entities.TBook
TTitle_book	sub_business_tier.entities.TTitle_book
ArrayList	java.util.ArrayList
TFacade	sub_business_tier.TFacade
List	java.util.List
PostConstruct	javax.annotation.PostConstruct

Remove unused imports

OK Cancel

Library2\_EJB2-ejb - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Search (Ctrl+I)

Projects Files Services

- Library2\_EJB2-war.war
- Library2\_EJB2-ejb.jar
- Configuration Files
- Server Resources
- Library2\_EJB2-ejb
  - Source Packages
    - business\_tier
      - Facade.java
    - integration\_tier
      - AbstractFacade.java
      - TBookFacade.java
      - TTitle\_bookFacade.java
  - Libraries
    - Library2\_interface2-ejb - dist
    - Library2\_JPA2 - dist/Library2\_
    - JDK 1.8 (Default)
    - GlassFish Server 4.1
  - Enterprise Beans
  - Configuration Files
    - MANIFEST.MF
    - persistence.xml
  - Server Resources

init - Navigator

Members

- init()
- titles(): <any>

Source

```
4 import integration_tier.TTitle_bookFacade;
5 import javax.ejb.EJB;
6 import javax.ejb.Stateless;
7
8 @Stateless
9 public class Facade implements FacadeRemote {
10     @EJB
11     private TTitle_bookFacade tTitle_bookFacade;
12     @EJB
13     private TBookFacade tBookFacade;
14
15     // Add business logic below. (Right-click in editor
16     // "Insert Code > Add Business Method")
17     @PostConstruct
18     public void init() {
19         try {
20             System.out.println("update1");
21             update_data();
22         } catch (Exception e) {
23
24         }
25     }
26
27     TFacade facade = new TFacade();
```

business\_tier.Facade > init > try > catch Exception e >

Output

GlassFish Server 4.1 Java DB Database Process Library2\_EJB2 (run)

23:1 INS

Library2\_EJB2-ejb - NetBeans IDE 8

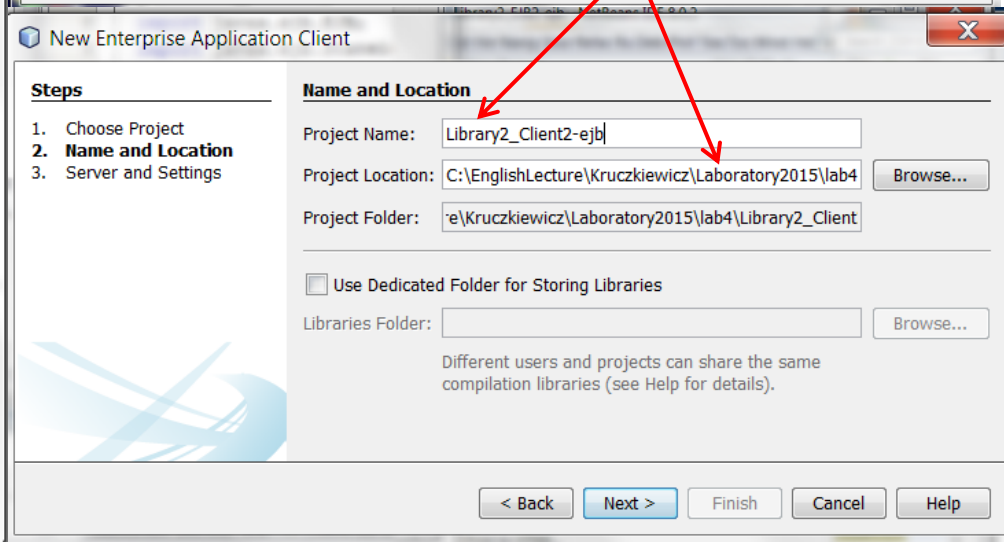
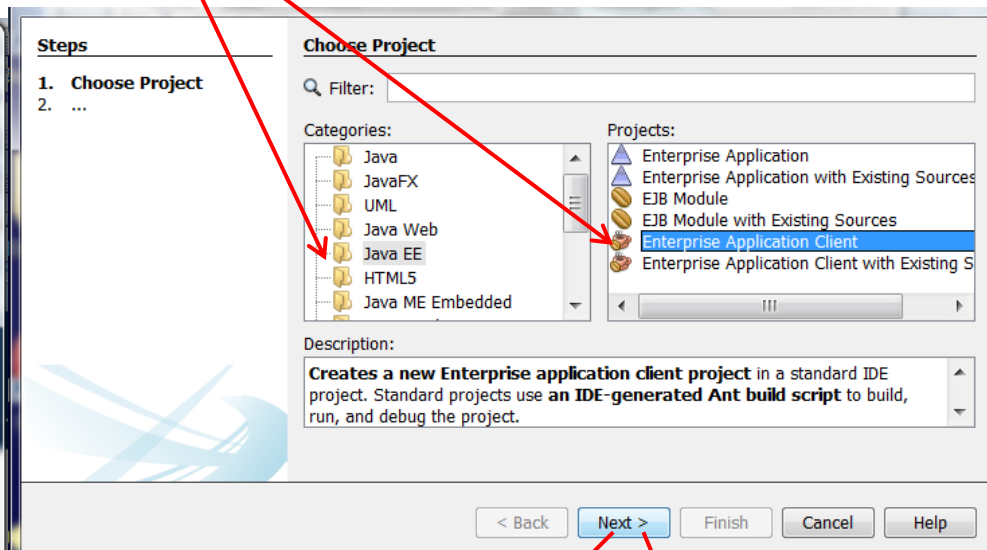
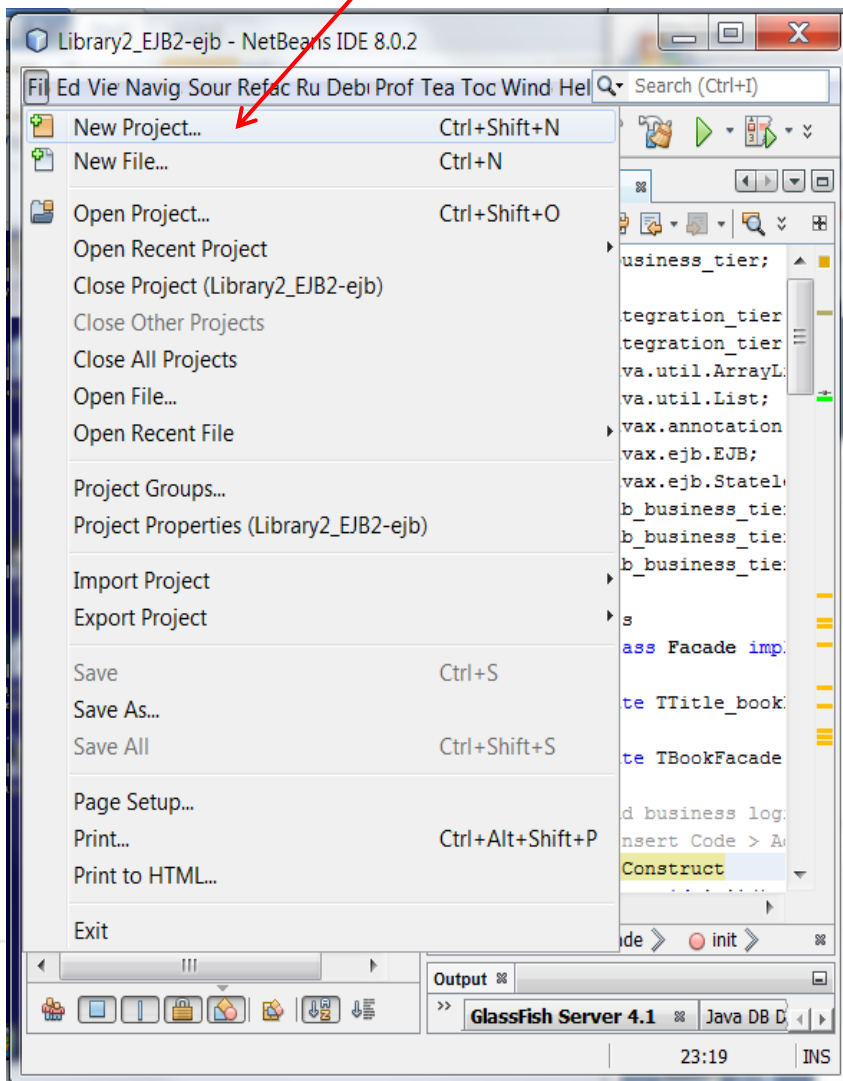
File Edit View Navigate Source Refactor

Projects Files Services

- Library2\_EJB2-ejb
  - Source Packages
    - business\_tier
      - Facade.java
    - integration\_tier
      - AbstractFacade.java
      - TBookFacade.java
      - TTitle\_bookFacade.java
  - Libraries
    - Library2\_interface2-ejb - dist
    - Library2\_JPA2 - dist/Library2\_
    - JDK 1.8 (Default)
    - GlassFish Server 4.1
  - Enterprise Beans
  - Configuration Files
    - MANIFEST.MF
    - persistence.xml
  - Server Resources
    - sun-resources.xml

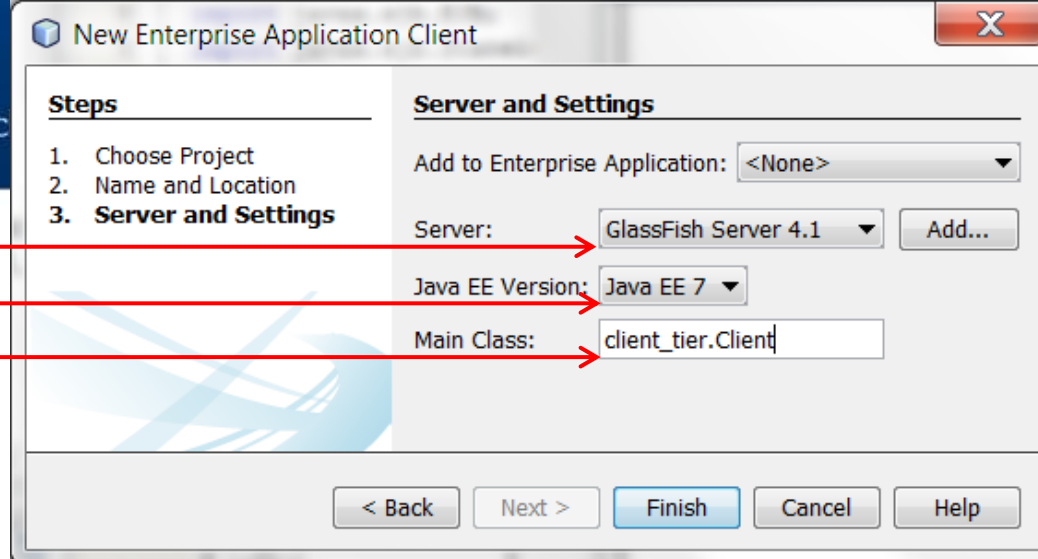


5. /5.1.1. Create the new Enterprise Application Client (the based on code of the Library1\_client1-ejb project of lab3) – click File/New Project.../Java EE/ Enterprise Application Client; fill name of project as Library2\_Client2-ejb and select the location of the project (the same as the location of other projects)

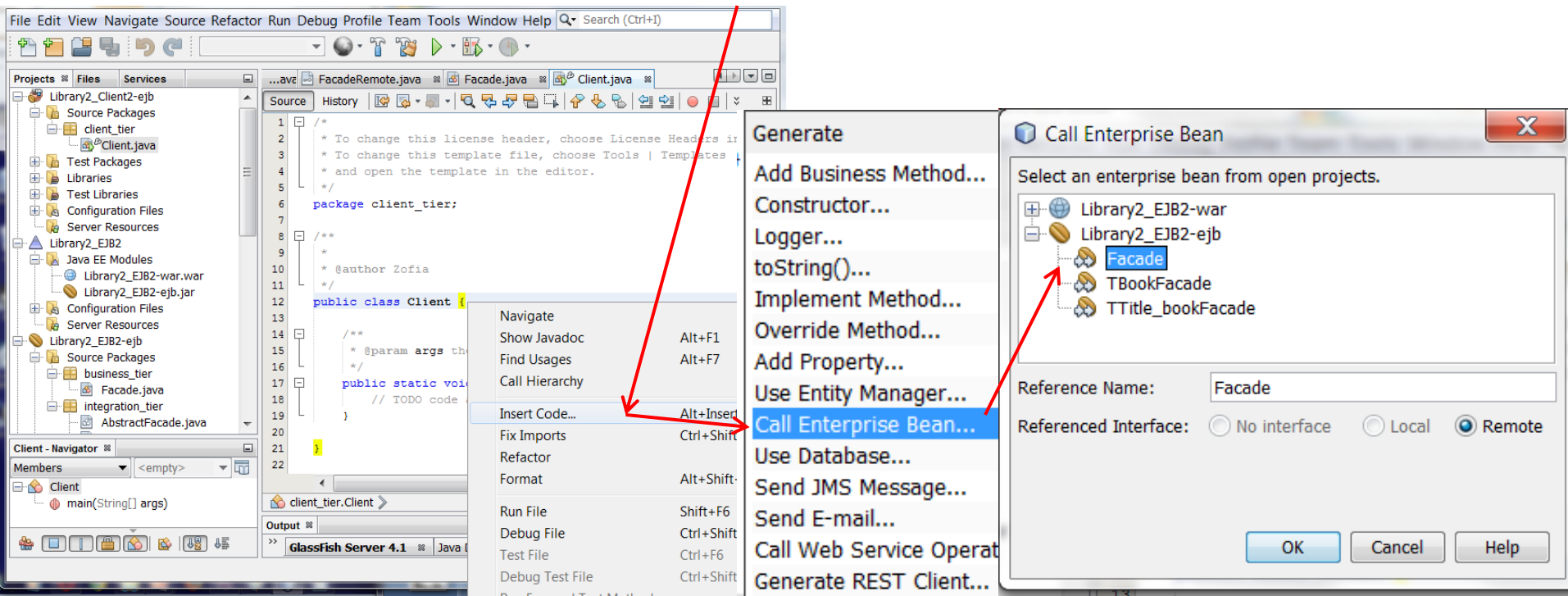




**5.1.2. Continuation – select the application server GlassFish 4.1, Java EE 7 platform and fill the name of main class – the same package and name of class as in the SE client project.**



**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 the Client class. Click Generate.**

The screenshot shows the NetBeans IDE interface with the following components:

- Project Explorer:** Shows the project structure for 'Library2\_Client2-ejb', including source packages like 'client\_tier' and 'Facade.java'.
- Code Editor:** Displays the 'Client.java' file with the following code:

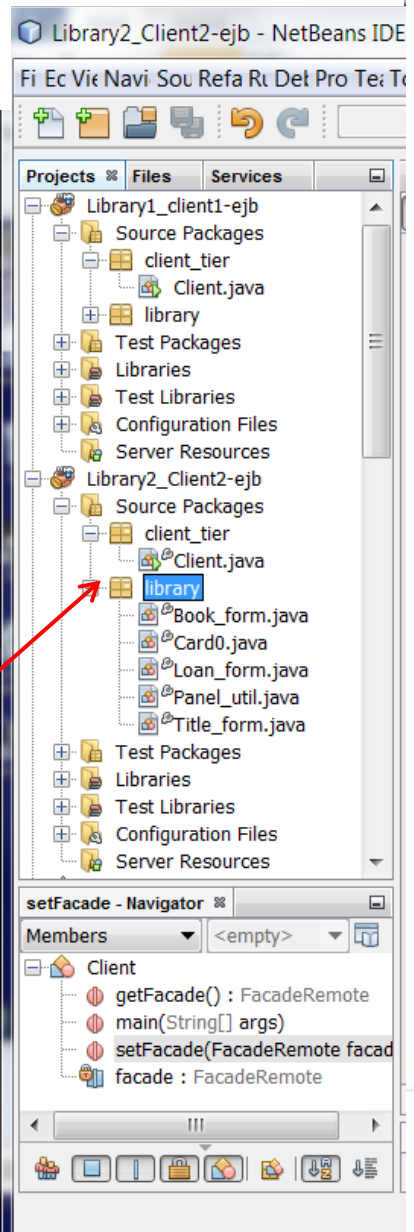
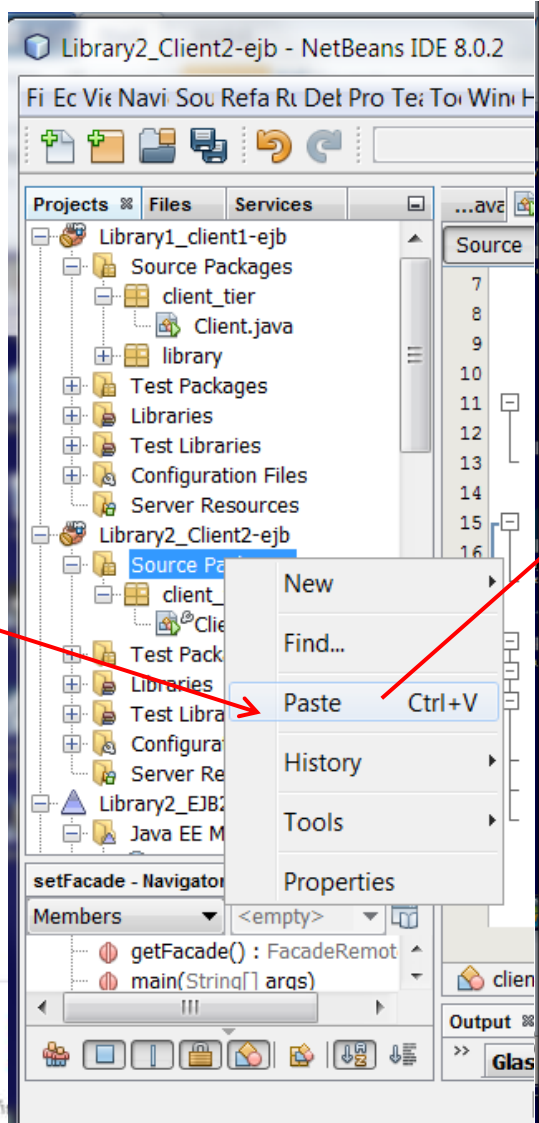
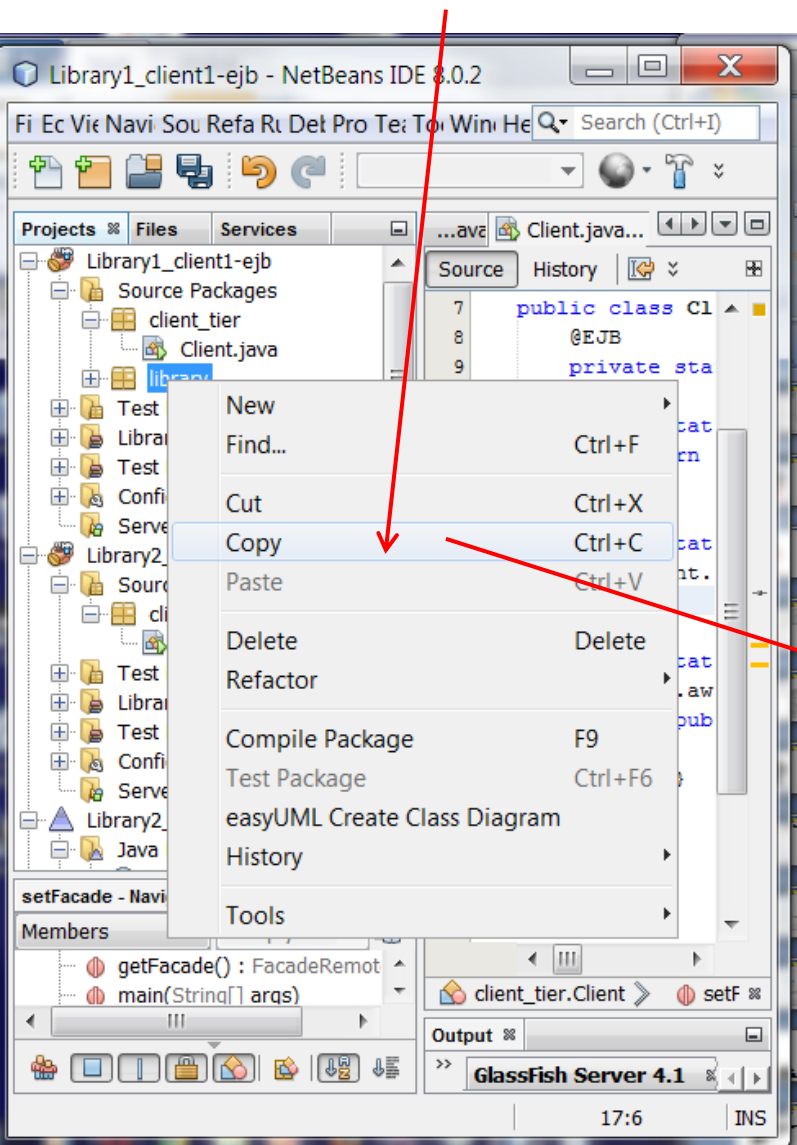
```
10  
11 /**  
12  *  
13  * @author Zofia  
14  */  
15 public class Client {  
16     @EJB  
17     private static FacadeRemote facade;  
18  
19     /**  
20     * @param args the  
21     */  
22     public static void  
23     // TODO code a  
24 }  
25  
26  
27
```
- Context Menu:** A right-click context menu is open over the code editor, with 'Insert Code...' selected. Other options include 'Navigate', 'Show Javadoc', 'Find Usages', 'Refactor', 'Format', 'Run File', etc.
- Generate Dialog:** The 'Generate Getters and Set...' dialog is open. It shows a tree view with 'Client' selected and 'facade : FacadeRemote' highlighted under it. The 'Encapsulate Fields' checkbox is unchecked. 'Generate' and 'Cancel' buttons are at the bottom.



**5.1.5. Create the copy of the library package from the Library1\_client1-ejb project (lab3) – right click the 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();
            }
        });
    }
}
```

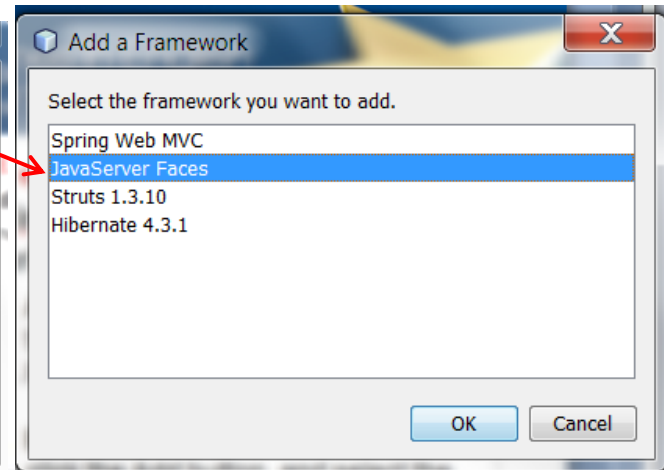
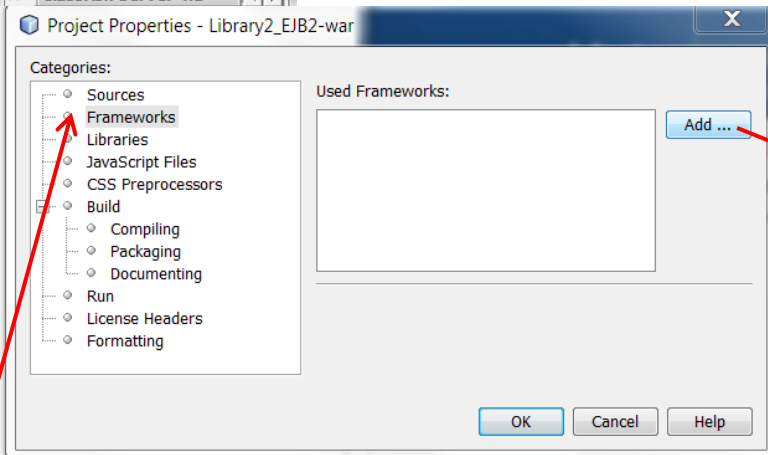
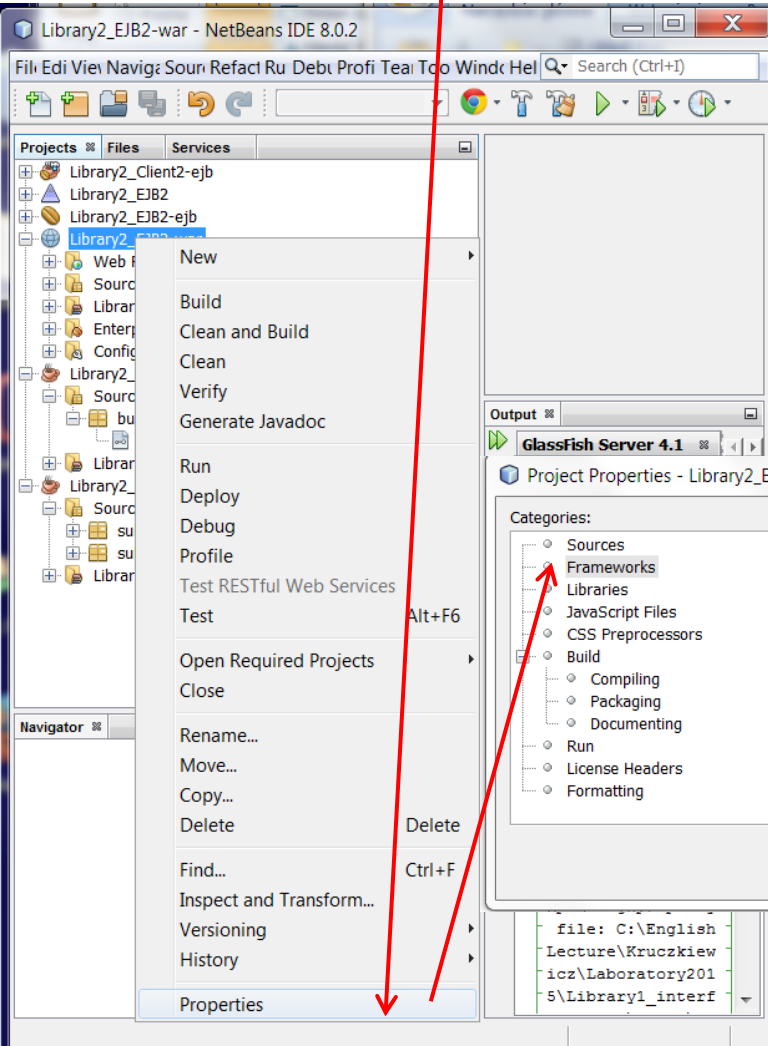
### 5.1.8. Code of the Client class Library2\_Client2-ejb project – the same as the code of lab3 client project



**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.

The image shows a NetBeans IDE window titled 'Library2\_EJB2-war - NetBeans IDE 8.0.2'. The 'Projects' tab is active, showing a tree view of the project structure. A right-click context menu is open over the 'Library2\_EJB2-war' project. The 'New' option is selected, and a sub-menu is displayed. In this sub-menu, the 'Other...' option is highlighted. A red arrow points from the 'New' option in the context menu to the 'Other...' option in the sub-menu. Another red arrow points from the 'Other...' option to the 'New File' dialog box.

The 'New File' dialog box is open, showing the 'Choose File Type' step. The 'Project' is set to 'Library2\_EJB2-war'. The 'Filter' is empty. The 'Categories' list on the left includes 'JMX', 'Web', 'HTML5', 'JavaServer Faces', 'Bean Validation', 'Struts', 'Spring Framework', and 'Enterprise JavaBeans'. The 'File Types' list on the right includes 'JSF Page', 'JSF Managed Bean', 'JSF Faces Configuration', 'JSF Composite Component', 'JSF Pages from Entity Classes', 'JSF Resource Library Contract', 'JSF Faces Component', 'Facelets Template', and 'Facelets Template Client'. The 'Facelets Template' option is selected. A red arrow points from the 'Facelets Template' option in the 'File Types' list to the 'Next >' button at the bottom of the dialog box.

Steps:

1. Choose File Type
2. ...

Choose File Type

Project: Library2\_EJB2-war

Filter:

Categories:

- JMX
- Web
- HTML5
- JavaServer Faces
- Bean Validation
- Struts
- Spring Framework
- Enterprise JavaBeans

File Types:

- JSF Page
- JSF Managed Bean
- JSF Faces Configuration
- JSF Composite Component
- JSF Pages from Entity Classes
- JSF Resource Library Contract
- JSF Faces Component
- Facelets Template
- Facelets Template Client

Description:

Creates a new Facelets template.

< Back Next > Finish Cancel Help



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

The screenshot shows a 'New Facelets Template' dialog box with the following fields and options:

- Steps:**
  1. Choose File Type
  2. **Name and Location**
- Name and Location:**
  - File Name:
  - Project:
  - Location:
  - Folder:
  - Created File:
- Layout Style:**  CSS  Table
- Template Grid:** A 2x4 grid of template icons. The icon in the second row, third column is selected with a blue border and a radio button.
- Buttons:** < Back, Next >, Finish, Cancel, Help





```
<?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://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/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>Facelets Template</title>
  </h:head>
  <h:body>
    <div id="top">
      <ui:insert name="top">Top</ui:insert>
    </div>
    <div>
      <div id="left">
        <ui:insert name="left">Left</ui:insert>
      </div>
      <div id="content" class="left_content">
        <ui:insert name="content">Content</ui:insert>
      </div>
      <div id="bottom">
        <ui:insert name="bottom">Bottom</ui:insert>
      </div>
    </h:body>
</html>
```

xmlns:ui="http://java.sun.com/jsf/facelets"  
xmlns:h="http://java.sun.com/jsf/html"

<h:outputStylesheet name="css/default.css" />  
<h:outputStylesheet name="css/cssLayout.css"/>  
<title>Library</title>

**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">

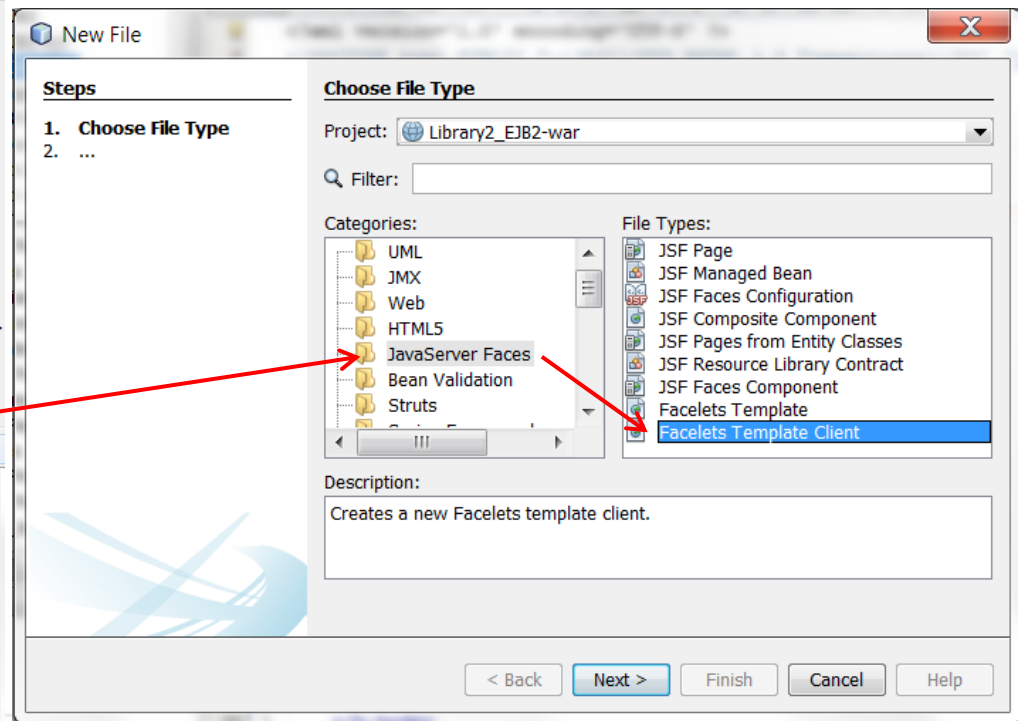
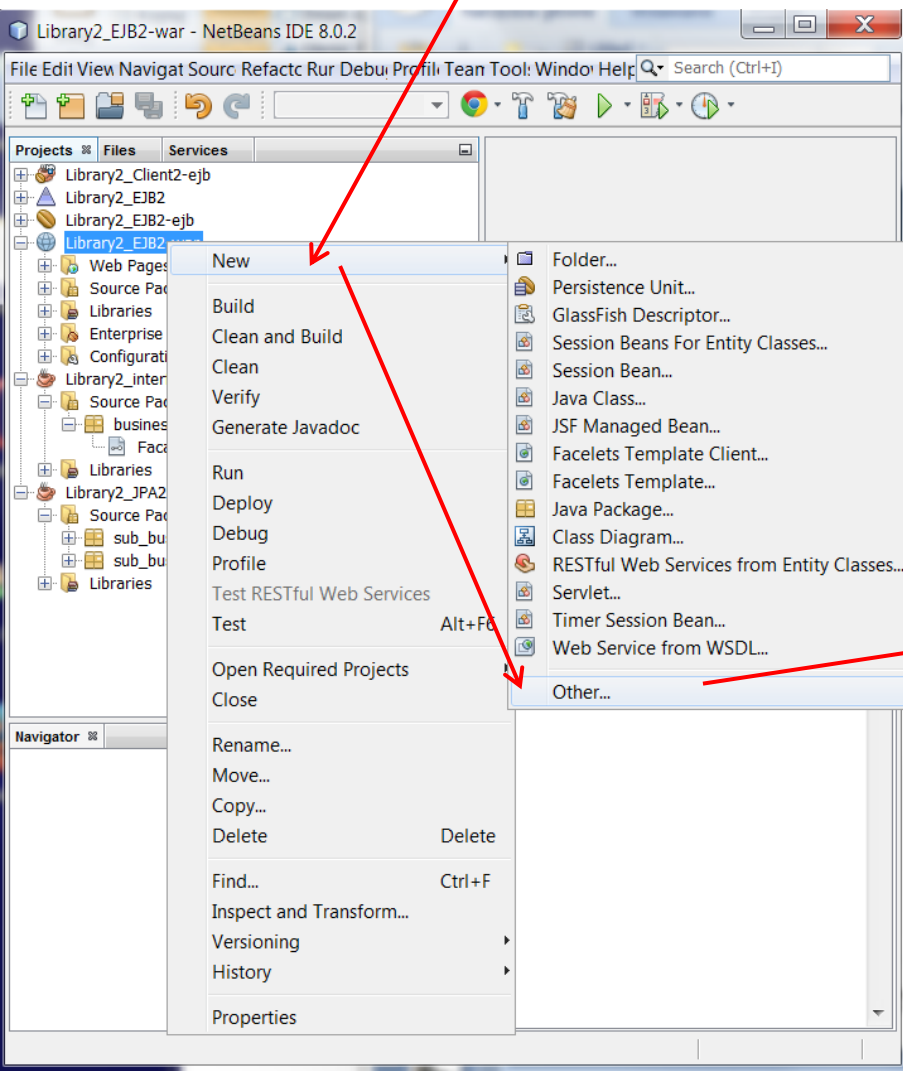
<h:head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <h:outputStylesheet name="css/default.css" />
  <h:outputStylesheet name="css/cssLayout.css"/>
  <title>Library</title>
</h:head>

<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.**





**New Facelets Template Client**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

File Name:

Project:

Folder:

Created File:

Template:

Generated Root Tag:  `<html>`  
 `<ui:composition>`

Sections To Generate:

**Select a template for which the client will be generated.**

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

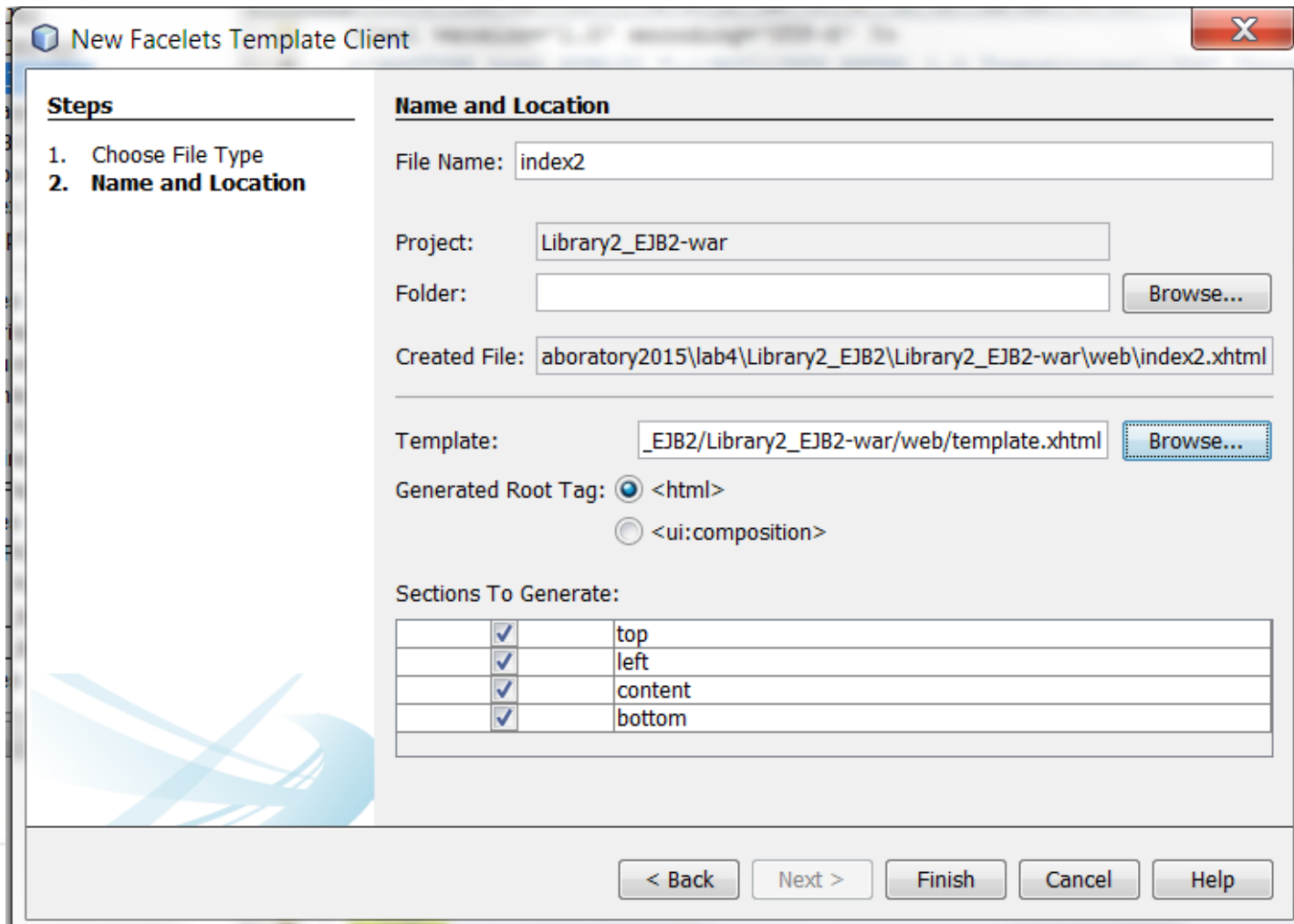
**Browse Files**

Folders:

- Web Pages
  - resources
  - WEB-INF
    - index.html
    - template.xhtml



6.4.3. Creation the main JSF page (index2), based on the previous defined template – continuation  
**Below:** after choosing the proper template



The screenshot shows a 'New Facelets Template Client' dialog box with the following fields and options:

- Steps:**
  1. Choose File Type
  2. **Name and Location**
- Name and Location:**
  - File Name:
  - Project:
  - Folder:
  - Created File:
  - Template:
  - Generated Root Tag:  <html>  <ui:composition>
  - Sections To Generate:

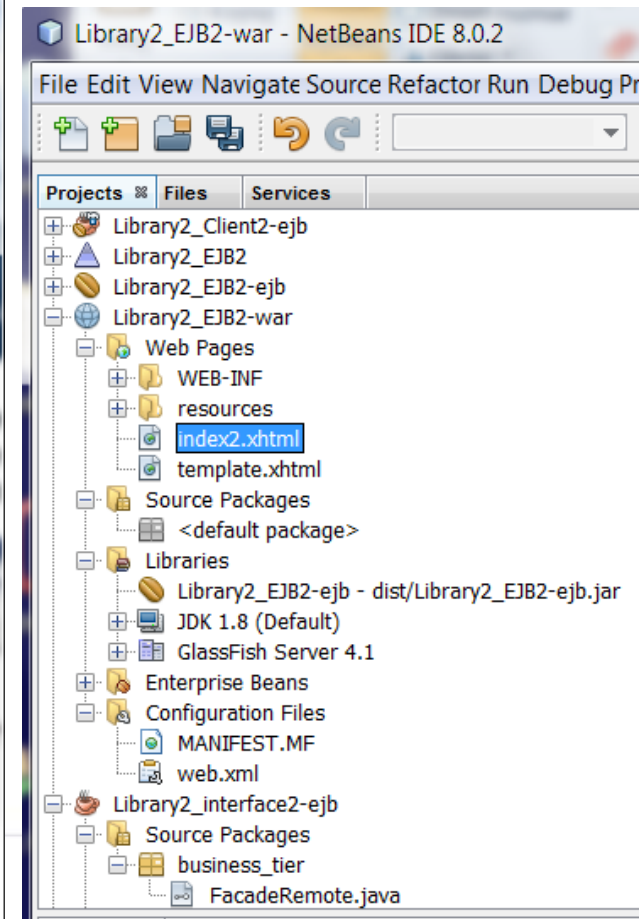
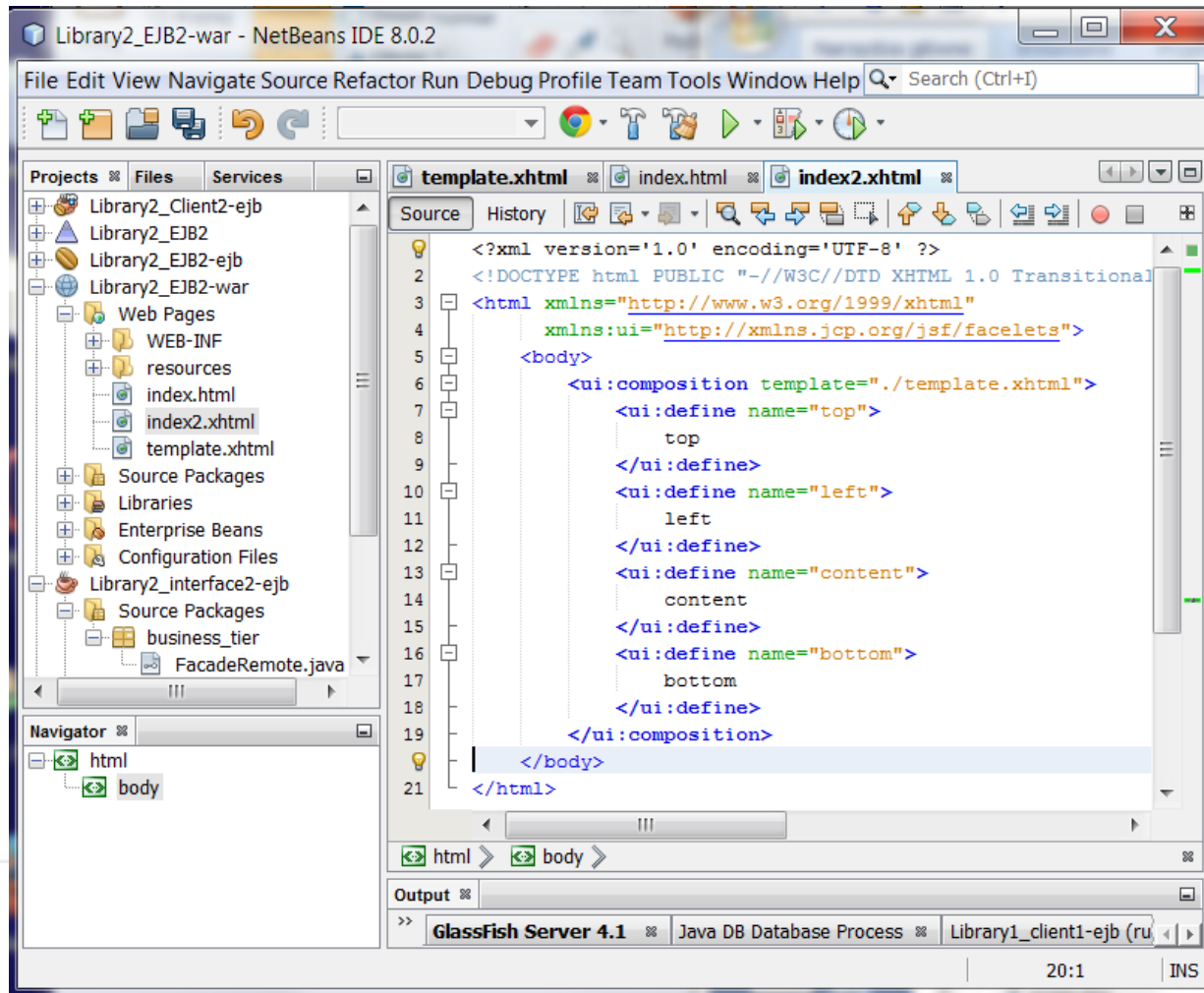
<input checked="" type="checkbox"/>	top
<input checked="" type="checkbox"/>	left
<input checked="" type="checkbox"/>	content
<input checked="" type="checkbox"/>	bottom

Buttons at the bottom: < Back, Next >, Finish, Cancel, Help





**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.xhtml);  
**At right:** after deleting unused pages - the index2.xhtml is the main JSF page



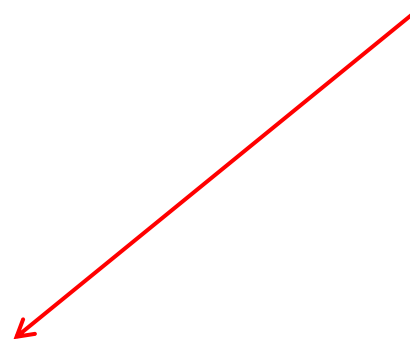


```
<?xml version='1.0' encoding='UTF-8' ?>  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"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">
```

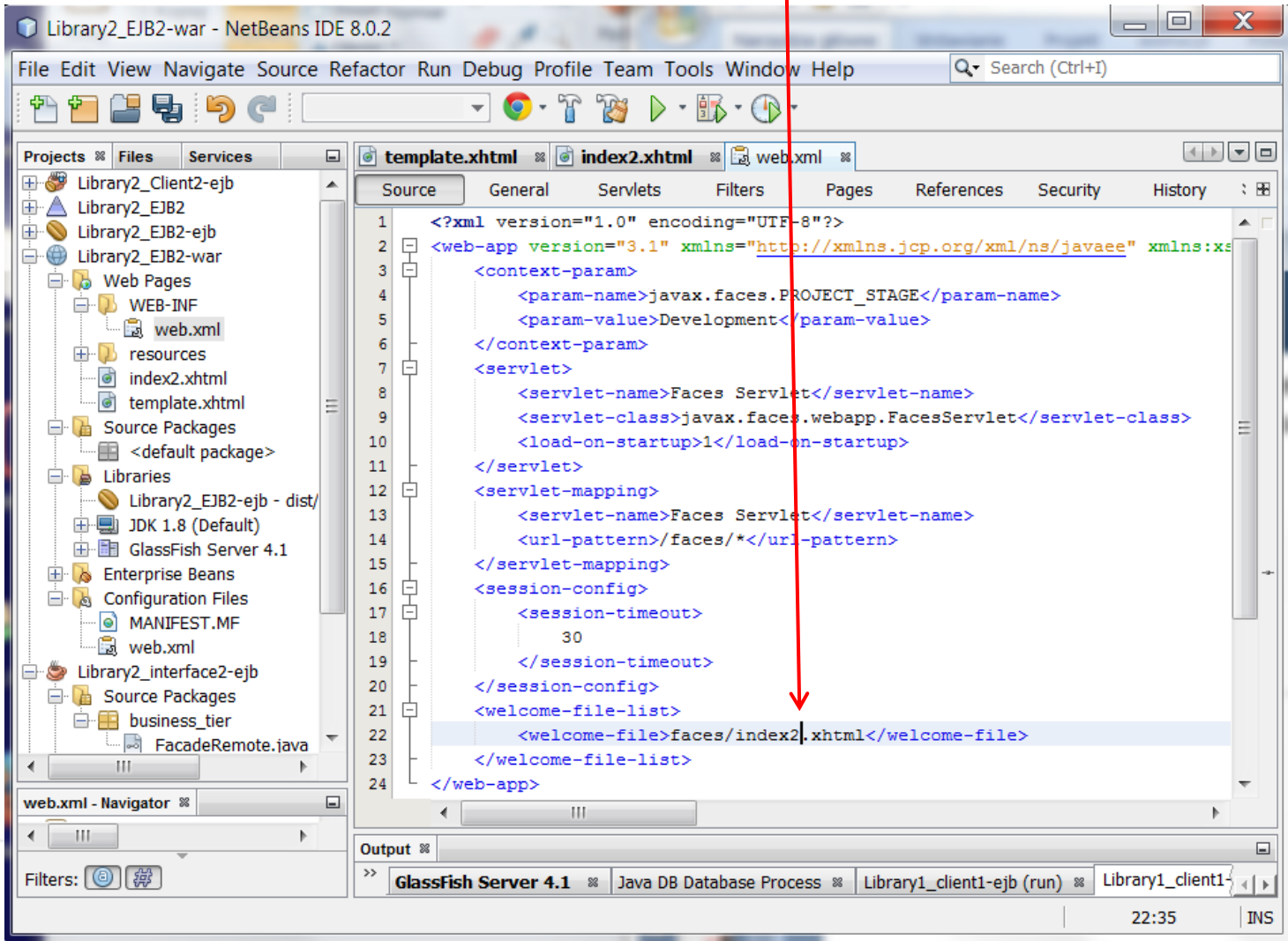
```
<body>  
  <ui:composition template="./template.xhtml">  
    <ui:define name="top">  
      top  
    </ui:define>  
    <ui:define name="left">  
      <h:link outcome="/faces/presentation_tier_view/Store_data" value="Store data"/><br/>  
      <h:link outcome="/faces/presentation_tier_view/Show_data"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

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





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



The screenshot shows the NetBeans IDE interface for a project named 'Library2\_EJB2-war'. The 'Projects' view on the left shows the project structure, including 'Web Pages', 'WEB-INF', and 'web.xml'. The 'Sources' view on the right shows the 'web.xml' file being edited. The XML content is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://xmlns.jcp.org/xml/ns/xsi" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
3   <context-param>
4     <param-name>javax.faces.PROJECT_STAGE</param-name>
5     <param-value>Development</param-value>
6   </context-param>
7   <servlet>
8     <servlet-name>Faces Servlet</servlet-name>
9     <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
10    <load-on-startup>1</load-on-startup>
11  </servlet>
12  <servlet-mapping>
13    <servlet-name>Faces Servlet</servlet-name>
14    <url-pattern>/faces/*</url-pattern>
15  </servlet-mapping>
16  <session-config>
17    <session-timeout>
18      30
19    </session-timeout>
20  </session-config>
21  <welcome-file-list>
22    <welcome-file>faces/index2.xml</welcome-file>
23  </welcome-file-list>
24 </web-app>
```

A red arrow points from the top of the IDE to the line containing the welcome-file configuration in the XML.



New Facelets Template Client

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

File Name:

Project:

Folder:

Created File:

Template:

Generated Root Tag:  <html>  
 <ui:composition>

Sections To Generate:

<input type="checkbox"/>	top
<input type="checkbox"/>	left
<input checked="" type="checkbox"/>	content
<input type="checkbox"/>	bottom

New Facelets Template Client

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

File Name:

Project:

Folder:

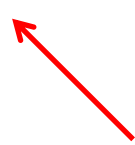
Created File:

Template:

Generated Root Tag:  <html>  
 <ui:composition>

Sections To Generate:

<input type="checkbox"/>	top
<input type="checkbox"/>	left
<input checked="" type="checkbox"/>	content
<input type="checkbox"/>	bottom

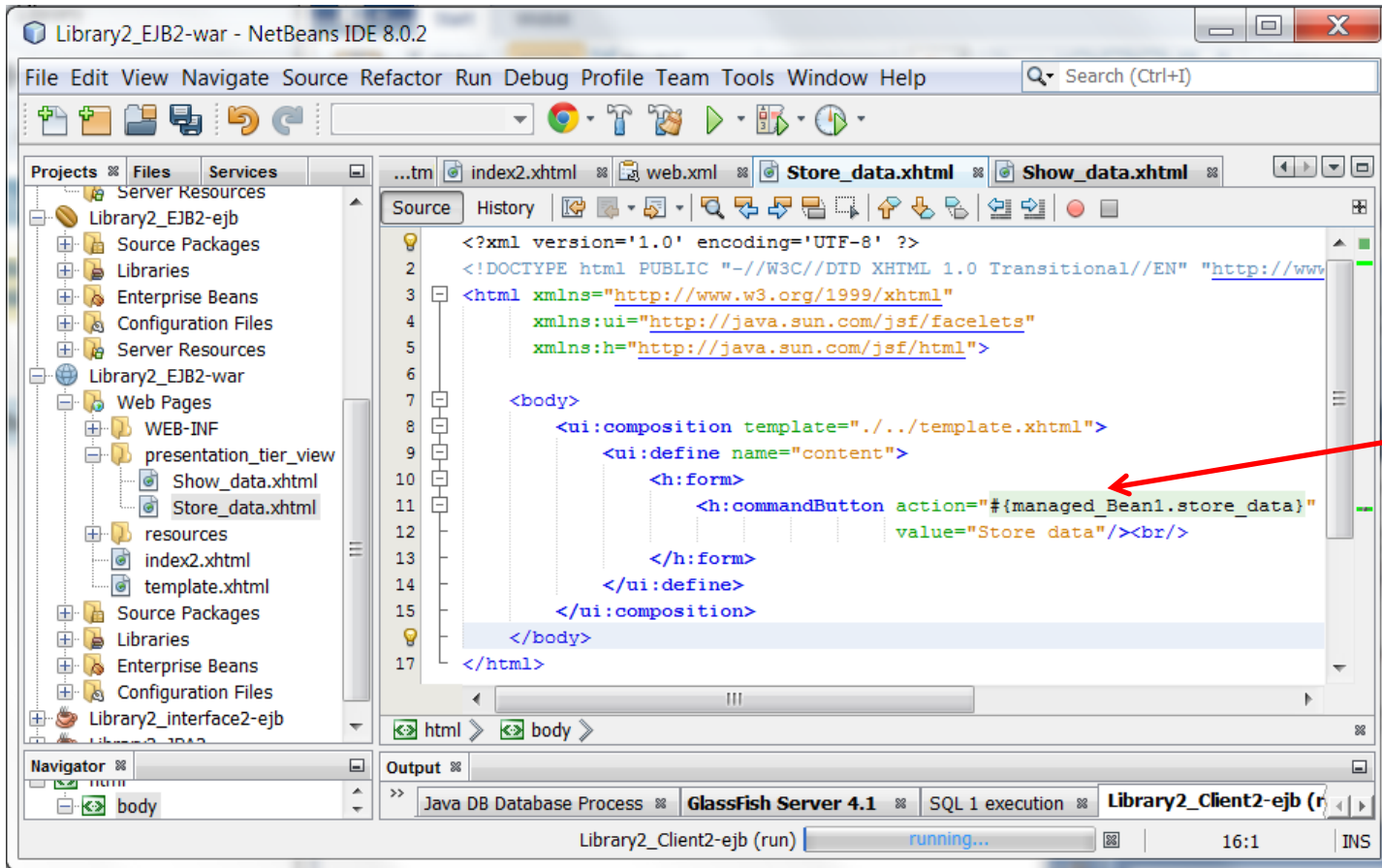


6.5. /6.5.1. Addition the two JSF pages, based on the previous defined template for **inserting data in the database** and **displaying the data from databases**





### 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

The screenshot shows the NetBeans IDE 8.0.2 interface. The main editor displays the source code of the Show\_data.xhtml page. The code is as follows:

```
<?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">
        <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:dataTable>
            </h:panelGroup>
          </h:form>
        </ui:define>
      </ui:define>
    </ui:composition>
  </body>
</html>
```

The IDE interface includes a Project Explorer on the left showing the project structure, a Navigator at the bottom left, and an Output window at the bottom right. The status bar at the bottom indicates the application is running.



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

The screenshot shows the NetBeans IDE interface for a project named 'Library2\_EJB2-war'. The main editor displays the code for 'Show\_data.xhtml'. The code is as follows:

```
27 <h:column>
28     <f:facet name="header">
29         <h:outputText value="ISBN"/>
30     </f:facet>
31     <h:outputText value="#{item.get(1) }"/>
32 </h:column>
33 <h:column>
34     <f:facet name="header">
35         <h:outputText value="Title"/>
36     </f:facet>
37     <h:outputText value="#{item.get(2) }"/>
38 </h:column>
39 <h:column>
40     <f:facet name="header">
41         <h:outputText value="Author"/>
42     </f:facet>
43     <h:outputText value="#{item.get(3) }"/>
44 </h:column>
45 <h:column>
46     <f:facet name="header">
47         <h:outputText value="Actor"/>
48     </f:facet>
49     <h:outputText value="#{item.get(4) }"/>
50 </h:column>
51 </h:dataTable>
52 </h:panelGroup>
53 </h:form>
54 </ui:define>
55 </ui:composition>
56 </body>
57 </html>
```

The IDE interface includes a 'Projects' window on the left showing the project structure, a 'Navigator' window at the bottom left showing the component tree, and an 'Output' window at the bottom right showing the status of the application server and database process.





### 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>
```



Library2\_EJB2-war - NetBeans IDE 8.0.2

File Edit View Navigat Source Refacto Run Debuç Profile Tear Tools Window Help Search (Ctrl+I)

Projects Files Services

- Source Packages
- Test Packages
- Libraries
- Test Libraries
- Configuration Files
- Server Resources
- Library2\_EJB2
  - Java EE Modules
  - Library2\_EJB2-war.war
  - Library2\_EJB2-ejb.jar
  - Configuration Files
  - Server Resources
  - Library2\_EJB2-ejb
  - Source Packages
  - Libraries
  - Enterprise Beans
  - Configurati
  - Server Res
- Web Pages
- WEB-INF
- present
- Store
- resourc
- index2.x
- templat
- Source Pac
- Libraries
- Enterprise E
- Configurati
- Library2\_interf
- Library2\_JPA2

Source

```

27 <h:column>
28   <f:facet name="header">
29     <h:outputText value="ISBN"/>
30   </f:facet>
31   <h:outputText value="#{item.get(1)}"
32 </h:column>
33 <h:column>
34   <f:facet name="header">
35     <h:outputText value="Title"/>
36   </f:facet>
37   <h:outputText value="#{item.get(2)}"
38 </h:column>
39 <h:column>
40   <f:facet name="header">

```

New

- Folder...
- Facellets Template Client...
- Facellets Template...
- Persistence Unit...
- GlassFish Descriptor...
- Session Beans For Entity Classes...
- Session Bean...
- Java Class...
- JSF Managed Bean...
- Java Package...
- Class Diagram...
- RESTful Web Services from Entity Classes...
- Servlet...
- Timer Session Bean...
- Web Service from WSDL...
- Other...

Alt+F6

Navigator

- CSS
- Classes
- inferud

Running: GlassFish Server 4.1, SQL 1 execution, 48:27, INS

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\_EJB2-war item in the Projects item, select the JavaServer Faces/JSF Managed Bean File Types. Click Next.

New File

Steps

1. Choose File Type
2. ...

Choose File Type

Project: Library2\_EJB2-war

Filter:

Categories:

- UML
- JMX
- Web
- HTML5
- JavaServer Faces
- Bean Validation

File Types:

- JSF Page
- JSF Managed Bean
- JSF Faces Configuration
- JSF Composite Component
- JSF Pages from Entity Classes
- JSF Resource Library Contract
- JSF Faces Component
- Facellets Template

Description:

Creates a new managed bean class.

< Back Next > Finish Cancel Help



**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.**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

Add data to configuration file

Configuration File:

Name:

Scope:

< Back   Next >   Finish   Cancel   Help





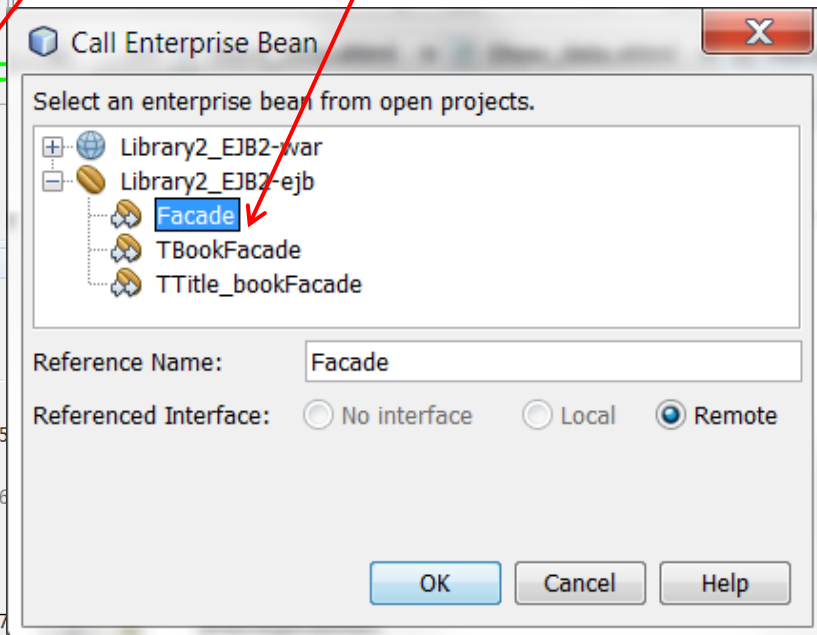
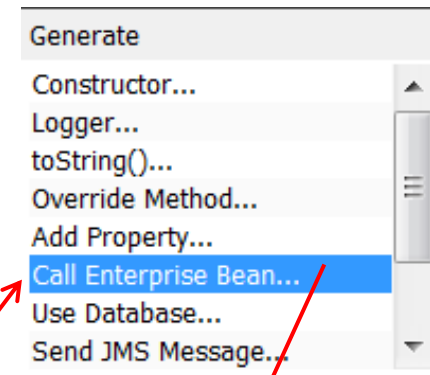
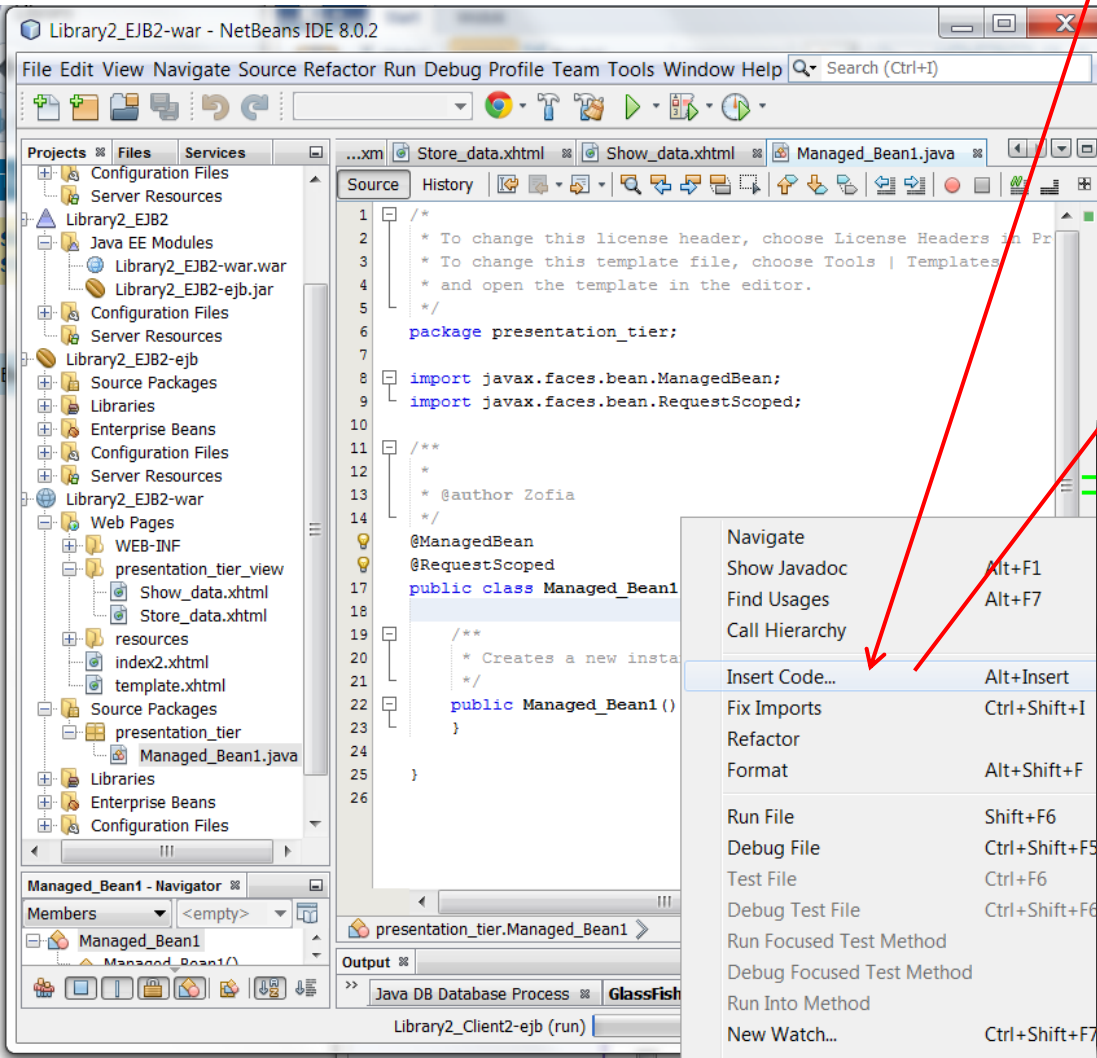
### 6.6.3. Creation of the Managed Bean type of object as the Presentation Tier object based on JSF technology - the generated code of this class

```
1  /*  
2  * To change this license header, choose License Headers in Project Properties.  
3  * To change this template file, choose Tools | Templates | the JSP files under  
4  * and open the template in the editor.  
5  */  
6  package presentation_tier;  
7  
8  import javax.faces.bean.ManagedBean;  
9  import javax.faces.bean.RequestScoped;  
10  
11  /**  
12   *  
13   * @author Zofia  
14   */  
15  @ManagedBean  
16  @RequestScoped  
17  public class Managed_Bean1 {  
18  
19      /**  
20       * Creates a new instance of Managed_Bean1  
21       */  
22      public Managed_Bean1 () {  
23      }  
24  }  
25  
26
```

The screenshot shows the NetBeans IDE interface. The left sidebar contains a Project Explorer with a tree view of the project structure. The main editor window displays the source code of the `Managed_Bean1.java` file. The code is a Managed Bean for JSF, with annotations `@ManagedBean` and `@RequestScoped`. It defines a class `Managed_Bean1` with a no-argument constructor. The bottom status bar shows the application is running.



**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.

The screenshot shows the NetBeans IDE interface for a project named 'Library2\_EJB2-war'. The 'Projects' window on the left shows the project structure, with 'Managed\_Bean1.java' selected under the 'presentation\_tier' source package. The main editor window displays the following code for 'Managed\_Bean1.java':

```
1 package presentation_tier;
2
3 import business_tier.FacadeRemote;
4 import javax.ejb.EJB;
5 import javax.faces.bean.ManagedBean;
6 import javax.faces.bean.RequestScoped;
7
8 @ManagedBean
9 @RequestScoped
10 public class Managed_Bean1 {
11     @EJB
12     private FacadeRemote facade;
13
14     public Managed_Bean1 () {
15     }
16
17 }
18
```

The IDE also shows a 'Managed\_Bean1 - Navigator' window at the bottom, which is currently empty. The status bar at the bottom indicates the current time is 14:5 and the user is logged in as INS.



### 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 facade; }
    public void setFacade(FacadeRemote facade) { this.facade = facade; }

    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; }  
}
```



### 6.6.9. The code of the Managed\_Bean1 class – the result

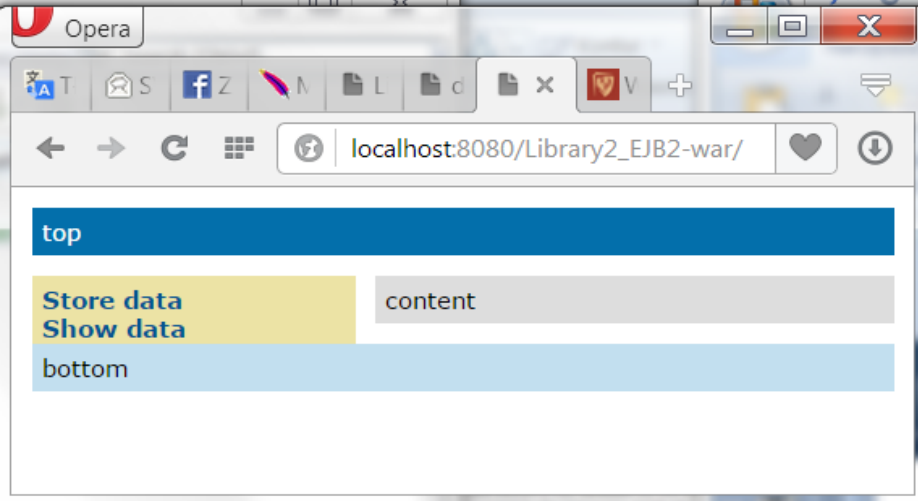
The screenshot displays the NetBeans IDE 8.0.2 interface. The left-hand 'Projects' pane shows a tree view for 'Library2\_EJB2-war', with 'presentation\_tier' and 'Managed\_Bean1.java' selected. The main editor window shows the source code for 'Managed\_Bean1.java'. The code includes package declarations, imports for 'FacadeRemote', 'EJB', 'ManagedBean', 'RequestScoped', 'DataModel', and 'ListDataModel'. The class is annotated with '@ManagedBean' and '@RequestScoped'. It contains a private field 'facade' of type 'FacadeRemote', a private field 'items' of type 'DataModel', a no-argument constructor, a 'getFacade()' method returning 'facade', and a 'setFacade(FacadeRemote facade)' method that assigns the parameter to 'this.facade'.

```
1 package presentation_tier;
2
3 import business_tier.FacadeRemote;
4 import javax.ejb.EJB;
5 import javax.faces.bean.ManagedBean;
6 import javax.faces.bean.RequestScoped;
7 import javax.faces.model.DataModel;
8 import javax.faces.model.ListDataModel;
9
10 @ManagedBean
11 @RequestScoped
12 public class Managed_Bean1 {
13     @EJB
14     private FacadeRemote facade;
15     private DataModel items;
16
17     public Managed_Bean1() {
18     }
19
20     public FacadeRemote getFacade() {
21         return facade;
22     }
23
24     public void setFacade(FacadeRemote facade) {
25         this.facade = facade;
26     }
27 }
```

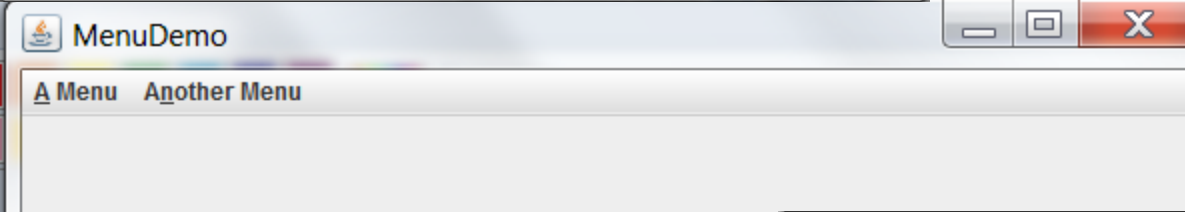


## 7.0. Running the program

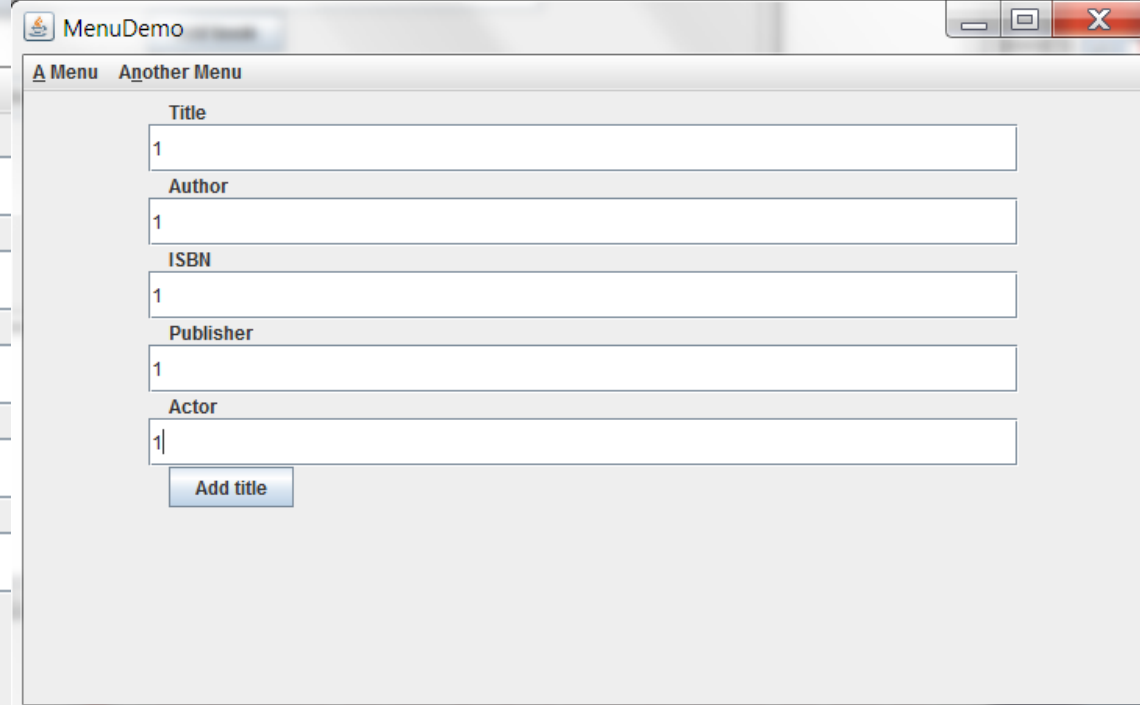
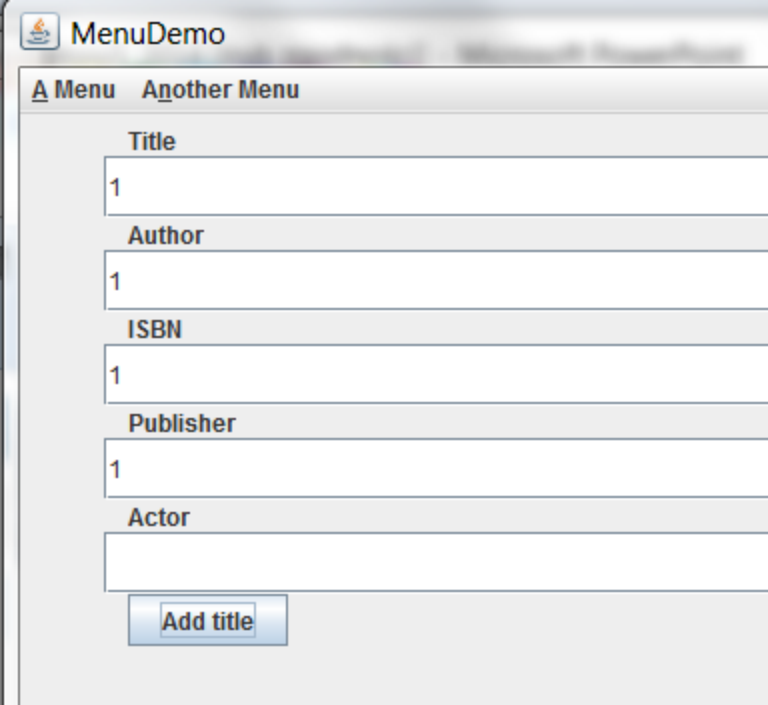
1. Connect with the Library2015 database (5 slide), if you start first one.
2. If you start after introduction of some changes in your programs, you must undeploy all deployed programs: Library2\_EJB2 , Library2\_Client2-ejb and Library2\_EJB2-war, accordingly to the 81 slide. After this development, 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/](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.

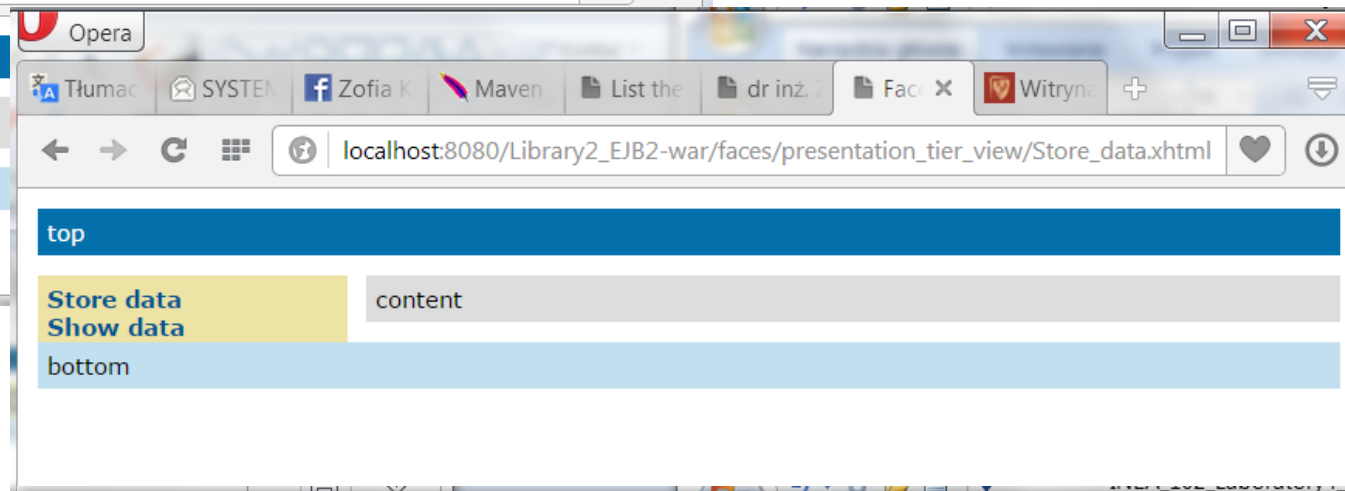
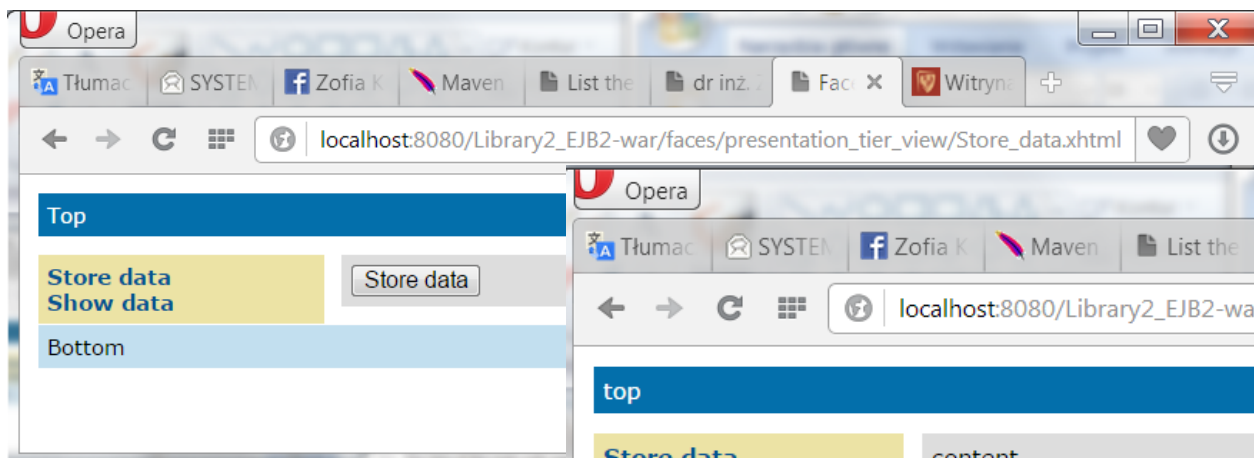




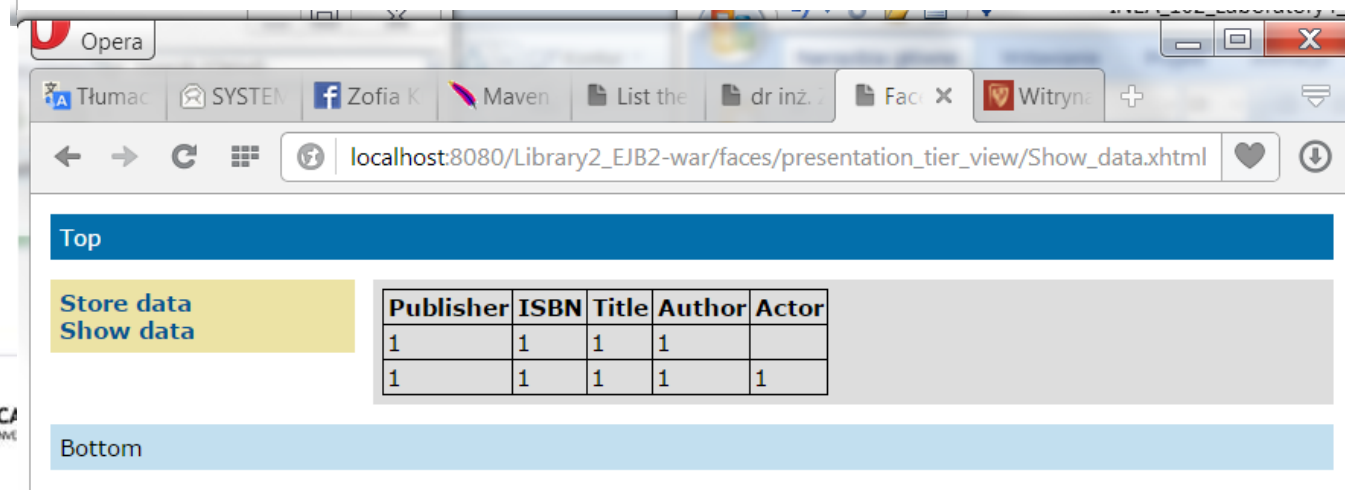


7.3. The view after choose the Store data link (the left part of page) of the **Library2\_EJB2-war** client

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



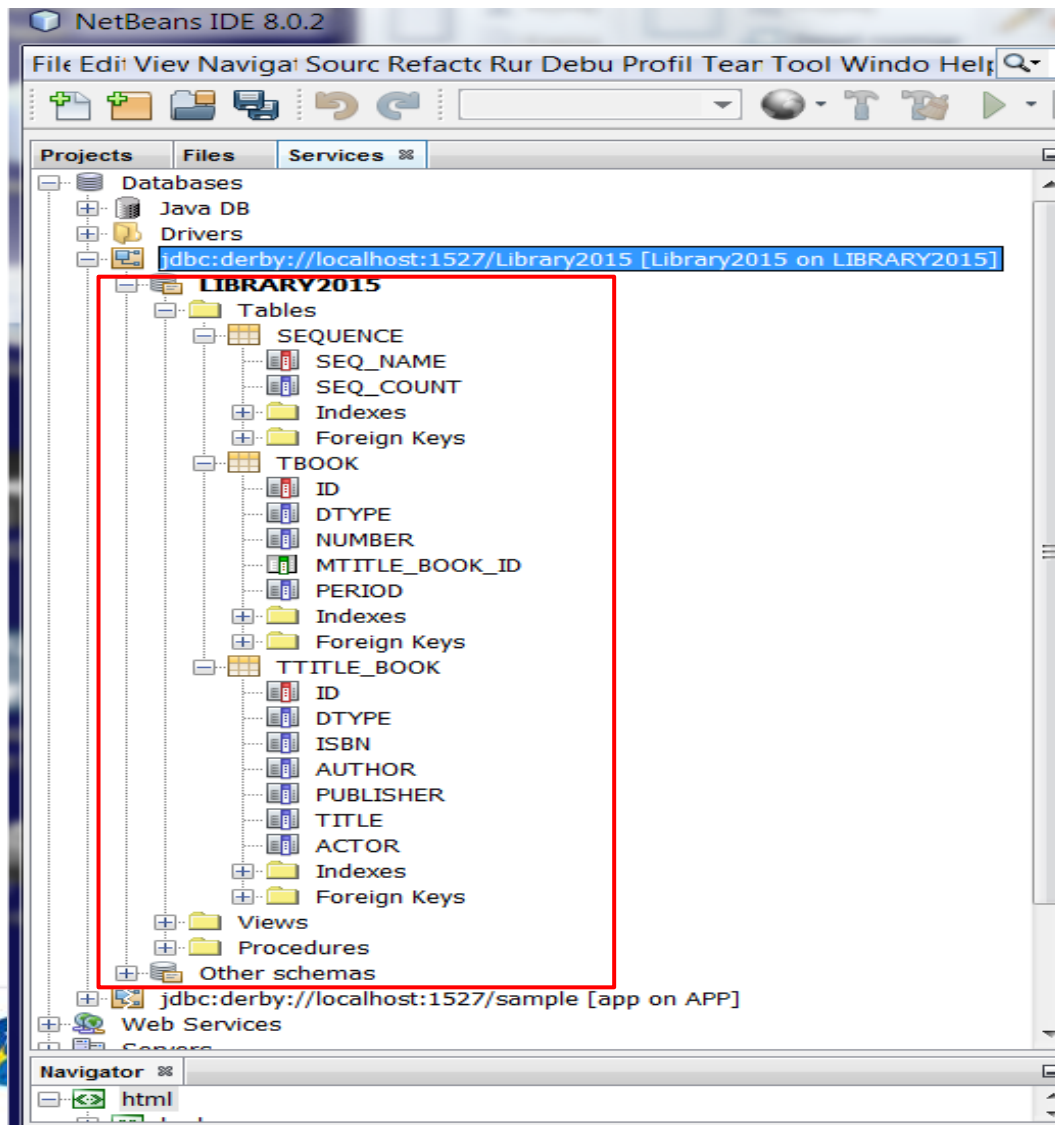
7.5. The view after choosing the Show data button (the left part of page) – i.e. after getting data from the database by the **Library2\_EJB2-war** client



Publisher	ISBN	Title	Author	Actor
1	1	1	1	
1	1	1	1	1



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)

The screenshot shows the NetBeans IDE 8.0.2 interface. The left sidebar displays the 'Services' view with a tree structure for the 'LIBRARY2015' database, including tables like 'TTITLE\_BOOK'. The main editor window shows a SQL query: `select * from LIBRARY2015.TTITLE_BOOK;`. Below the query, the results are displayed in a table with 2 rows and 8 columns: #, ID, DTYPE, ISBN, AUTHOR, PUBLISHER, TITLE, and ACTOR. The second row is highlighted in blue.

#	ID	DTYPE	ISBN	AUTHOR	PUBLISHER	TITLE	ACTOR
1		1 TTITLE_book	1	1	1	1	<NULL>
2		2 TTITLE_book_on_tape	1	1	1	1	1

The bottom of the IDE shows the 'Output' window with a log entry for 'Library2\_Client2-ejb (run) #2'. The log contains the following text:

```
Copying 2 files to C:\EnglishLecture\Kruczkiewicz\Laboratory2015\lab4\Library2_Client2-ejb\dist\Library2_Client2-ejbClient  
Warning: C:\EnglishLecture\Kruczkiewicz\Laboratory2015\lab4\Library2_Client2-ejb\dist\gfdeploy\Library2_Client2-ejb does not exist.
```

The status bar at the bottom indicates 'Library2\_Client2-ejb (run) #2' is running.



MenuDemo

A Menu Another Menu

Publisher	ISBN	Title	Author	Actor
1	1	1	1	
1	1	1	1	1
2	2	2	2	
2	2	2	2	2

Number of a book

4

Period of a book

Add book

Books

Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 3 Period: Mon Feb 23 14:18:10 CET 2015

Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 3 Period: Mon Feb 23 14:18:10 CET 2015

Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 4

7.8. The **Library2\_Client2-ejb** form to adding the new books of the selected title, as the application data.

Opera

Tłumac SYSTEM Zofia K Maven List the dr inż. 2 Face X Witryna

localhost:8080/Library2\_EJB2-war/faces/presentation\_tier\_view/Show\_data.xhtml#

Top

Store data Show data

Publisher	ISBN	Title	Author	Actor
1	1	1	1	
1	1	1	1	1
2	2	2	2	
2	2	2	2	2

Bottom







MenuDemo

Menu Another Menu

Publisher	ISBN	Title	Author	Actor
1	1	1	1	
1	1	1	1	1
2	2	2	2	
2	2	2	2	2

Number of a book

Period of a book

Add book

Books

Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 3 Period: Mon Feb 23 14:18:10 CET 2015

Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 3 Period: Mon Feb 23 14:18:10 CET 2015

Title: 1 Author: 1 ISBN: 1 Publisher: 1 Actor: 1 Number: 4

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

Opera

Tłumac SYSTEM Zofia K Maven List the dr inż. 2 Face X Witryna

localhost:8080/Library2\_EJB2-war/faces/presentation\_tier\_view/Show\_data.xhtml#

Top

Store data Show data

Publisher	ISBN	Title	Author	Actor
1	1	1	1	
1	1	1	1	1
2	2	2	2	
2	2	2	2	2

Bottom



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)

The screenshot shows the NetBeans IDE 8.0.2 interface. The left sidebar displays the 'Projects' view with a tree structure including 'Databases', 'Web Services', and 'Applications'. The 'Applications' folder is expanded to show 'Library2\_Client2-ejb'. The main editor window displays an SQL query: `select * from LIBRARY2015.TTITILE_BOOK;`. Below the query, the 'Output' window shows the execution results in a table format. The table has 7 columns: #, ID, DTYPE, ISBN, AUTHOR, PUBLISHER, TITLE, and ACTOR. The results are as follows:

#	ID	DTYPE	ISBN	AUTHOR	PUBLISHER	TITLE	ACTOR
1	1	TTitle_book	1	1	1	1	<NULL>
2	2	TTitle_book_on_tape	1	1	1	1	1
3	51	TTitle_book	2	2	2	2	<NULL>
4	52	TTitle_book_on_tape	2	2	2	2	2

The 'Output' window also shows the following text: 'Executed successfully in 0 s.', 'Line 1, column 1', and 'Execution finished after 0 s, 0 error(s) occurred.' The status bar at the bottom indicates 'Library2\_Client2-ejb (run) running...'.



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)

The screenshot shows the NetBeans IDE 8.0.2 interface. The left sidebar displays the 'Projects' view with a tree structure of databases and servers. The main editor window shows an SQL query: `select * from LIBRARY2015.TBOOK;`. Below the query, the results are displayed in a table with 3 rows and 6 columns: #, ID, DTYPE, NUMBER, MTITLE\_BOOK\_ID, and PERIOD. The output window at the bottom shows the execution log for 'Library2\_Client2-ejb (run)', indicating successful execution in 0,031 s.

#	ID	DTYPE	NUMBER	MTITLE_BOOK_ID	PERIOD
1	3	TBook	1	1	<NULL>
2	54	TBook	4	2	<NULL>
3	53	TBook_period	3	2	2015-02-23



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)

The screenshot shows the NetBeans IDE 8.0.2 interface. The left sidebar displays a project tree with a database connection 'jdbc:derby://localhost:1527/Library2015' containing a schema 'LIBRARY2015' with tables 'SEQUENCE', 'TBOOK', and 'TTITLE\_BOOK'. The main editor window shows an SQL query: 'select \* from LIBRARY2015."SEQUENCE";'. Below the editor, the results of the query are displayed in a table with columns '#', 'SEQ\_NAME', and 'SEQ\_COUNT'. The table contains one row: '1', 'SEQ\_GEN', '100'. The bottom output window shows the execution log: 'Executed successfully in 0,046 s. Line 1, column 1' and 'Execution finished after 0,046 s, 0 error(s) occurred'. The status bar at the bottom indicates 'Library2\_Client2-ejb (run) running...'.

#	SEQ_NAME	SEQ_COUNT
1	SEQ_GEN	100



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

The screenshot shows the NetBeans IDE 8.0.2 interface. The Services view on the left shows a project structure with 'Library2\_Client2-ejb' selected. A context menu is open over this component, with 'Undeploy' highlighted. The SQL editor in the center shows a query: `select * from LIBRARY2015.TTITLE_BOOK;`. Below the editor, a table displays the results of the query. The Output window at the bottom shows the execution log for 'Library2\_Client2-ejb (run)', indicating successful execution.

#	ID	DTYPE	ISBN	AUTHOR	PUBLISHER	TITLE	ACTOR
1	1	TTITLE_book	1	1	1	1	<NULL>
2	2	TTITLE_book_on_tape	1	1	1	1	1
3	51	TTITLE_book	2	2	2	2	<NULL>
4	52	TTITLE_book_on_tape	2	2	2	2	2

Output window content:  
Java DB Database Process | GlassFish Server 4.1 | SQL 1 execution | Library2\_Client2-ejb (run) | Library2  
Executed successfully in 0 s.  
Line 1, column 1  
Execution finished after 0 s, 0 error(s) occurred.





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