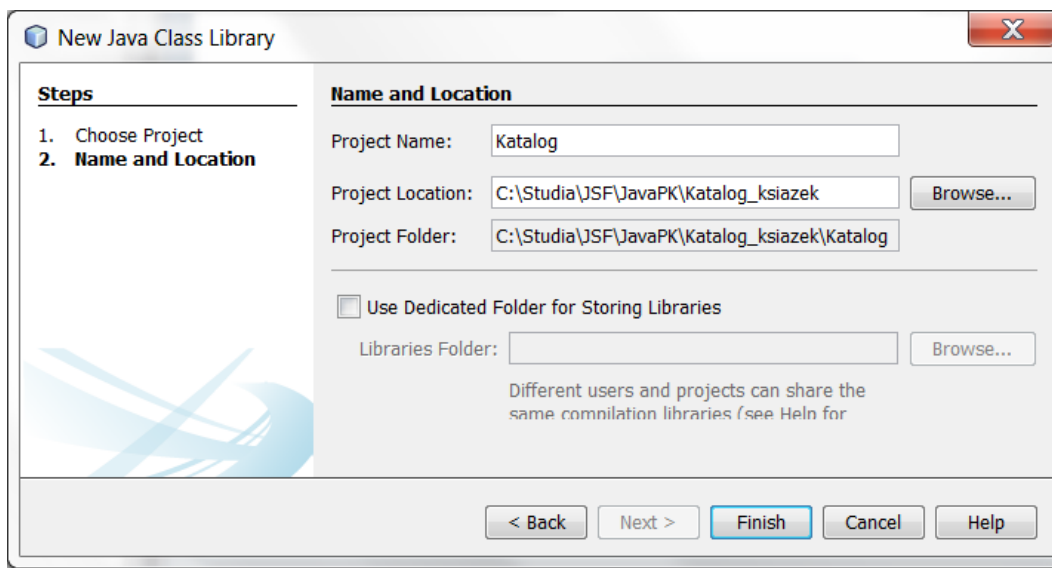
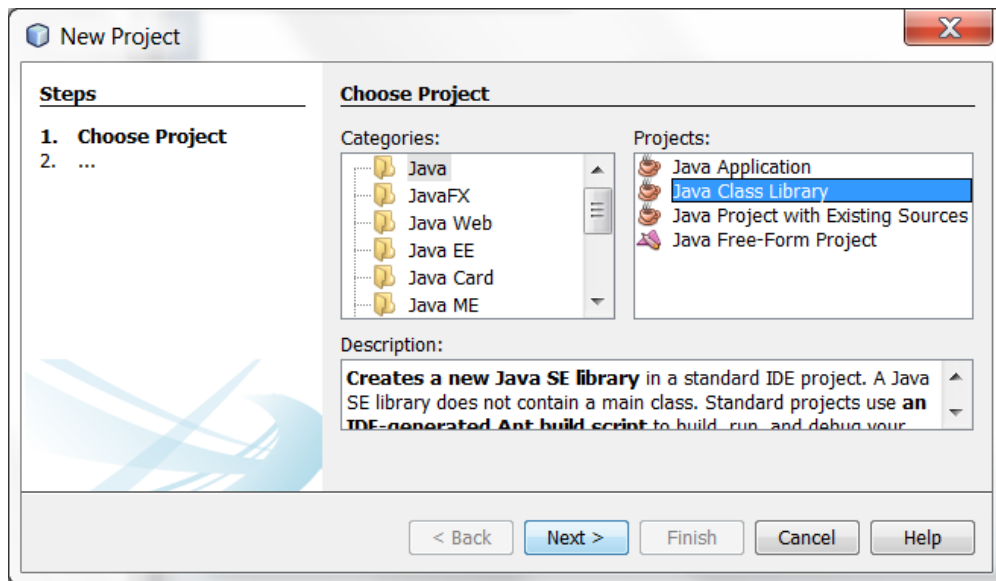


Przekształcanie aplikacji
internetowej typu JSF
wygenerowanej na podstawie bazy
danych do aplikacji typu EE
zawierającej dodatkowo klienta typu
Enterprise.

Zofia Kruczkiewicz

1. Zakładanie projektu Katalog typu Java Class Library do przechowywania obiektowego modelu danych – projekt należy do **warstwy biznesowej**



2. Zakładanie projektu Katalog_interfejs typu Java Class Library do przechowywania interfejsów obiektów typu Session Bean for Entity Class (technologia JTA) – projekt należy do **warstwy biznesowej**

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: Katalog_interfejs

Project Location: C:\Studia\JSF\JavaPK\Katalog_ksiazek

Project Folder: tudia\JSF\JavaPK\Katalog_ksiazek\Katal

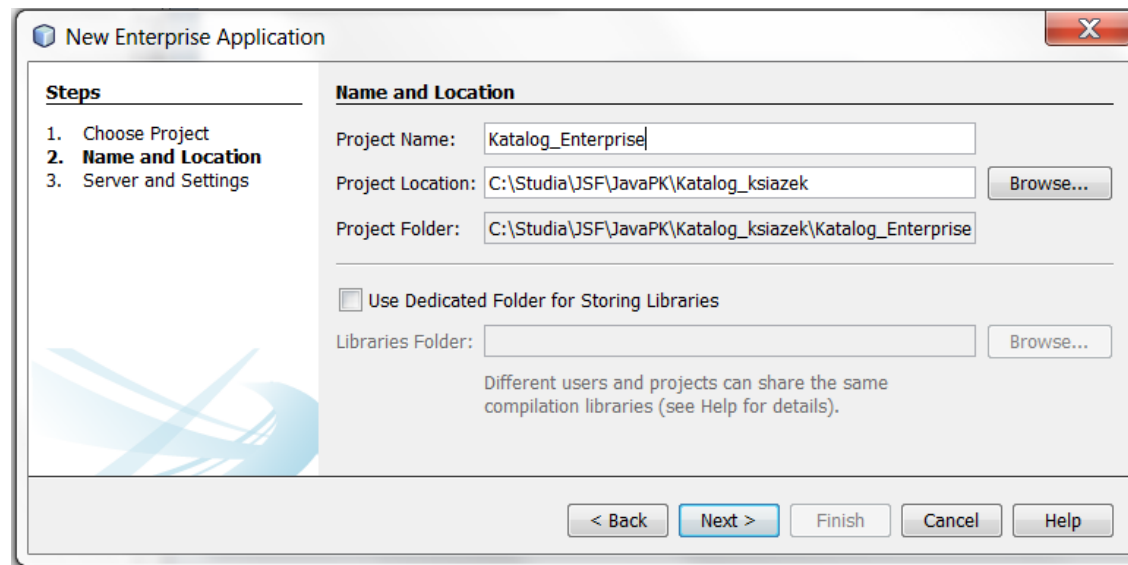
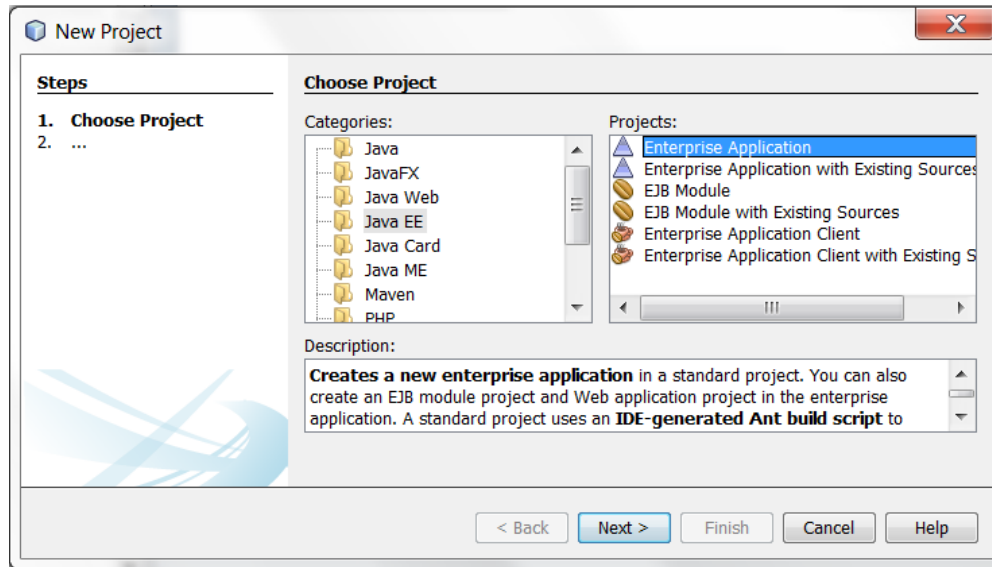
Use Dedicated Folder for Storing Libraries

Libraries Folder:

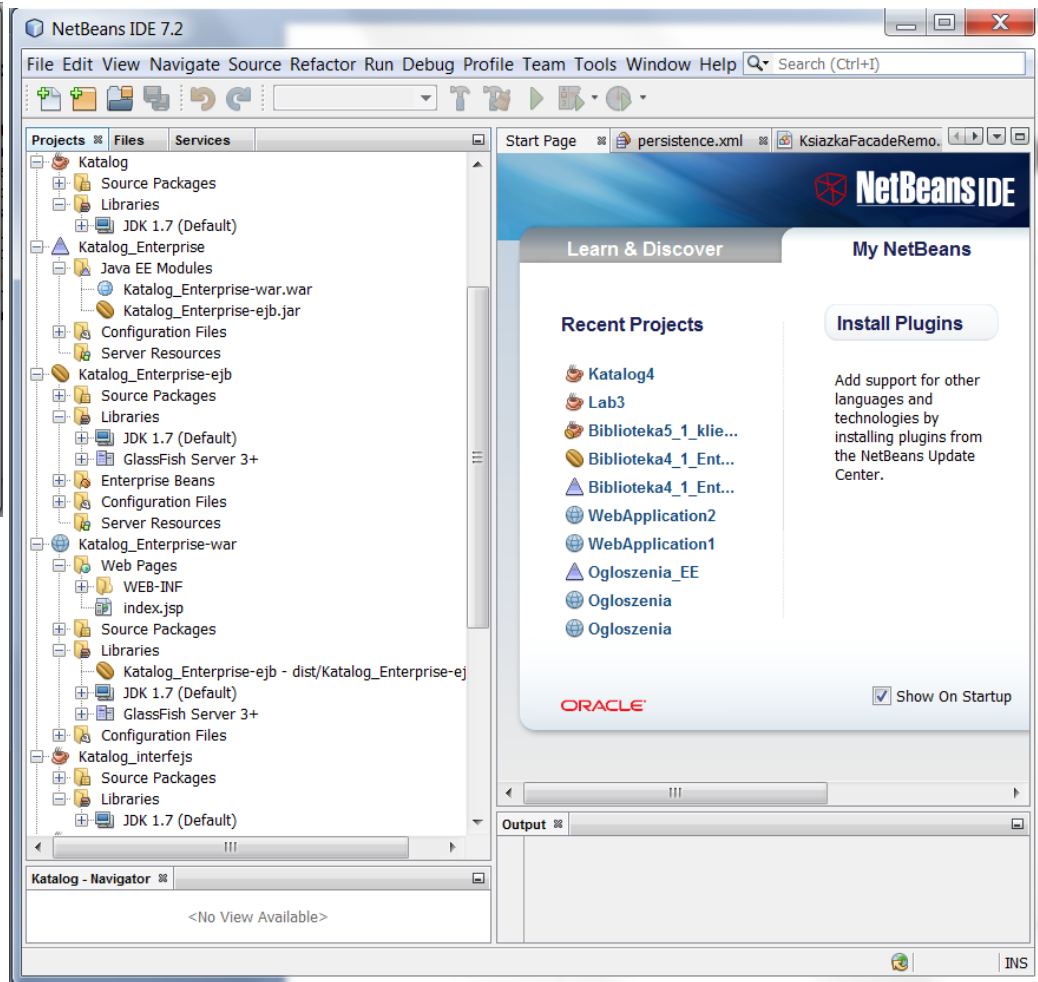
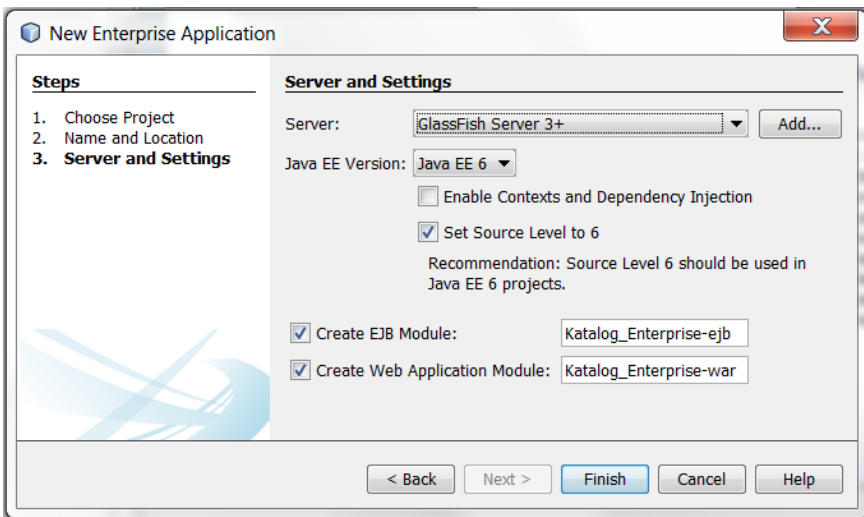
Different users and projects can share the same compilation libraries

< Back Next > **Finish** Cancel Help

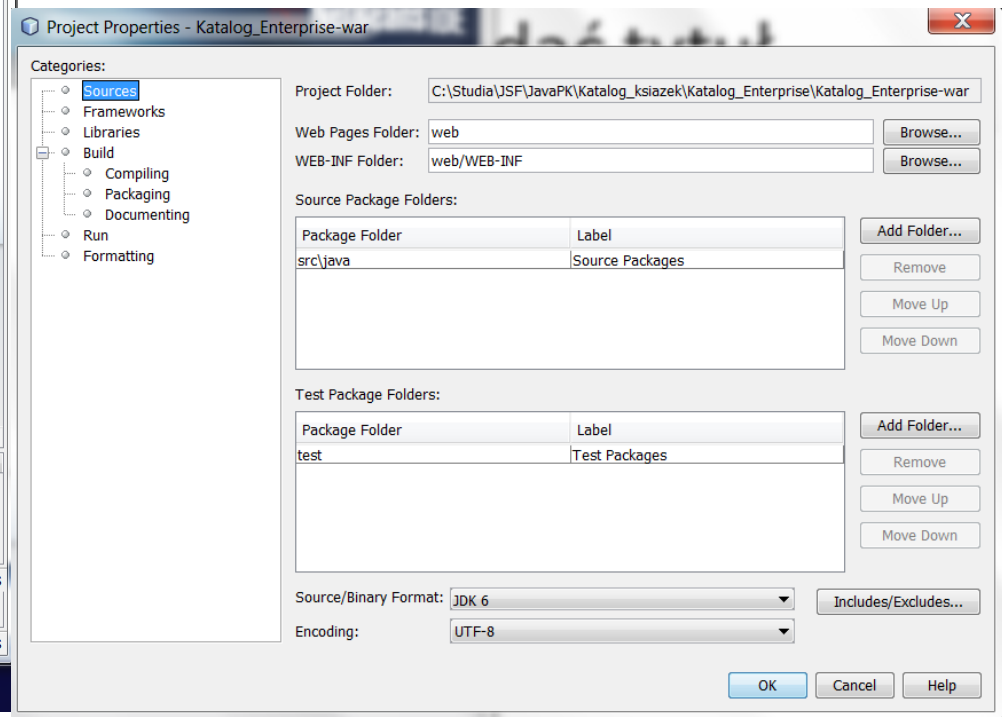
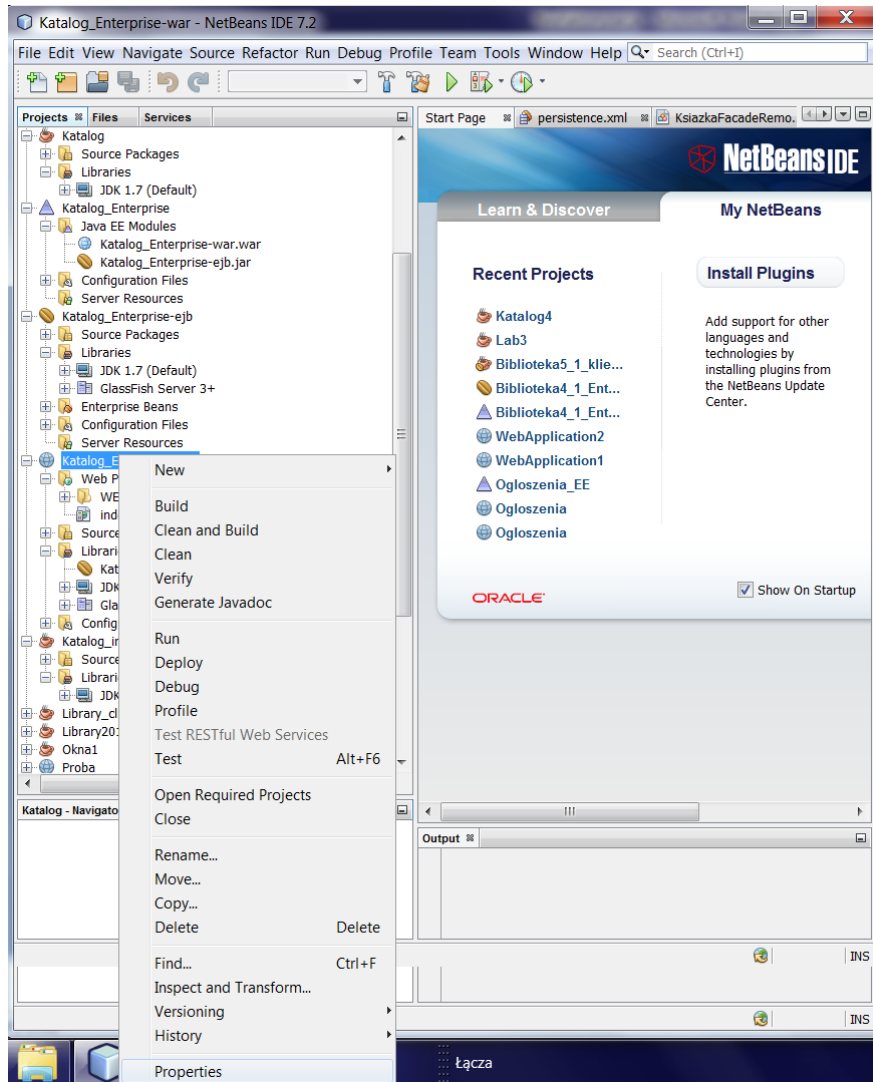
3.1. Zakładanie głównego projektu typu EE – wybór typu aplikacji nadanie nazwy oraz umieszczenie jej w wybranym katalogu. Projekt należy do **warstwy biznesowej**.



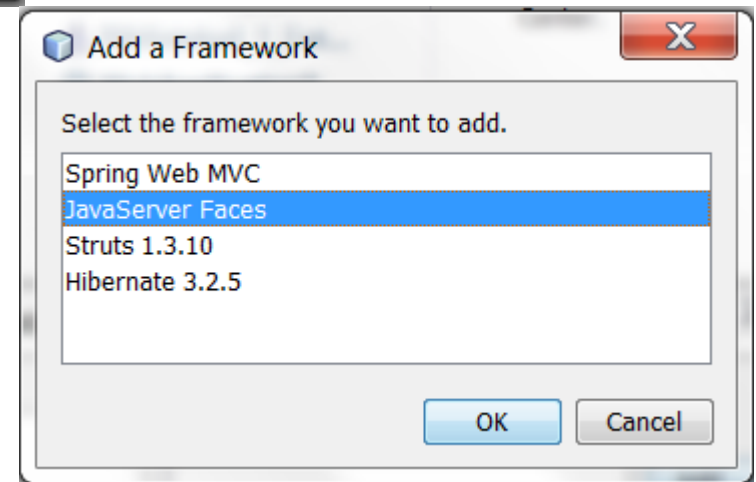
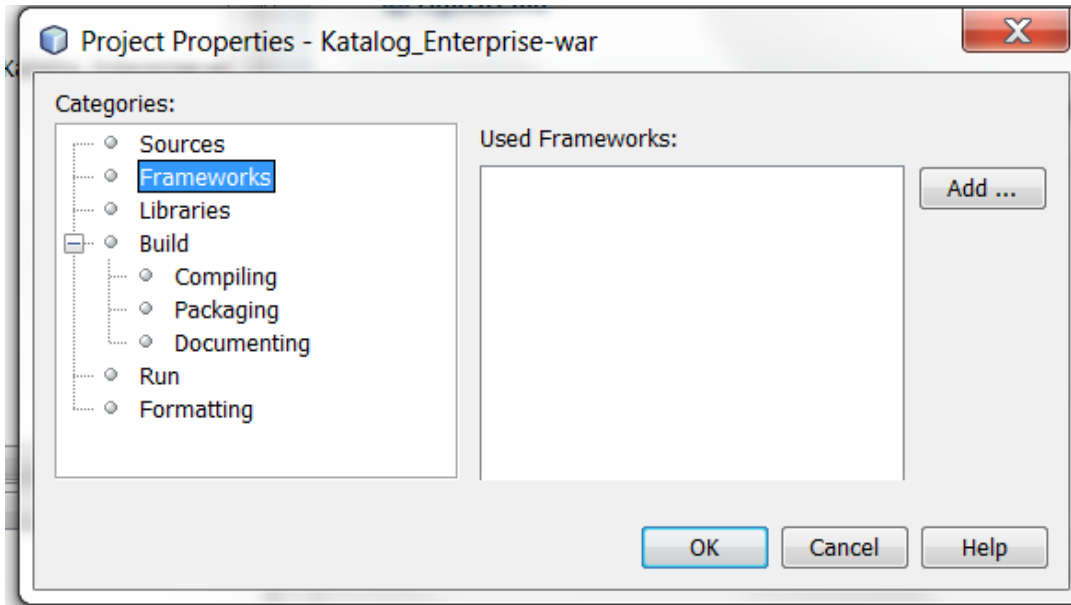
3.2. Wybór serwera aplikacji GlassFish Server 3, platformy Java EE 6 oraz modułu EJB Katalog_Enterprise-ejb oraz modułu internetowego Katalog_Enterprise-war (lewa część strony). Po prawej projekty po zakończeniu czynności ze stron 2-5



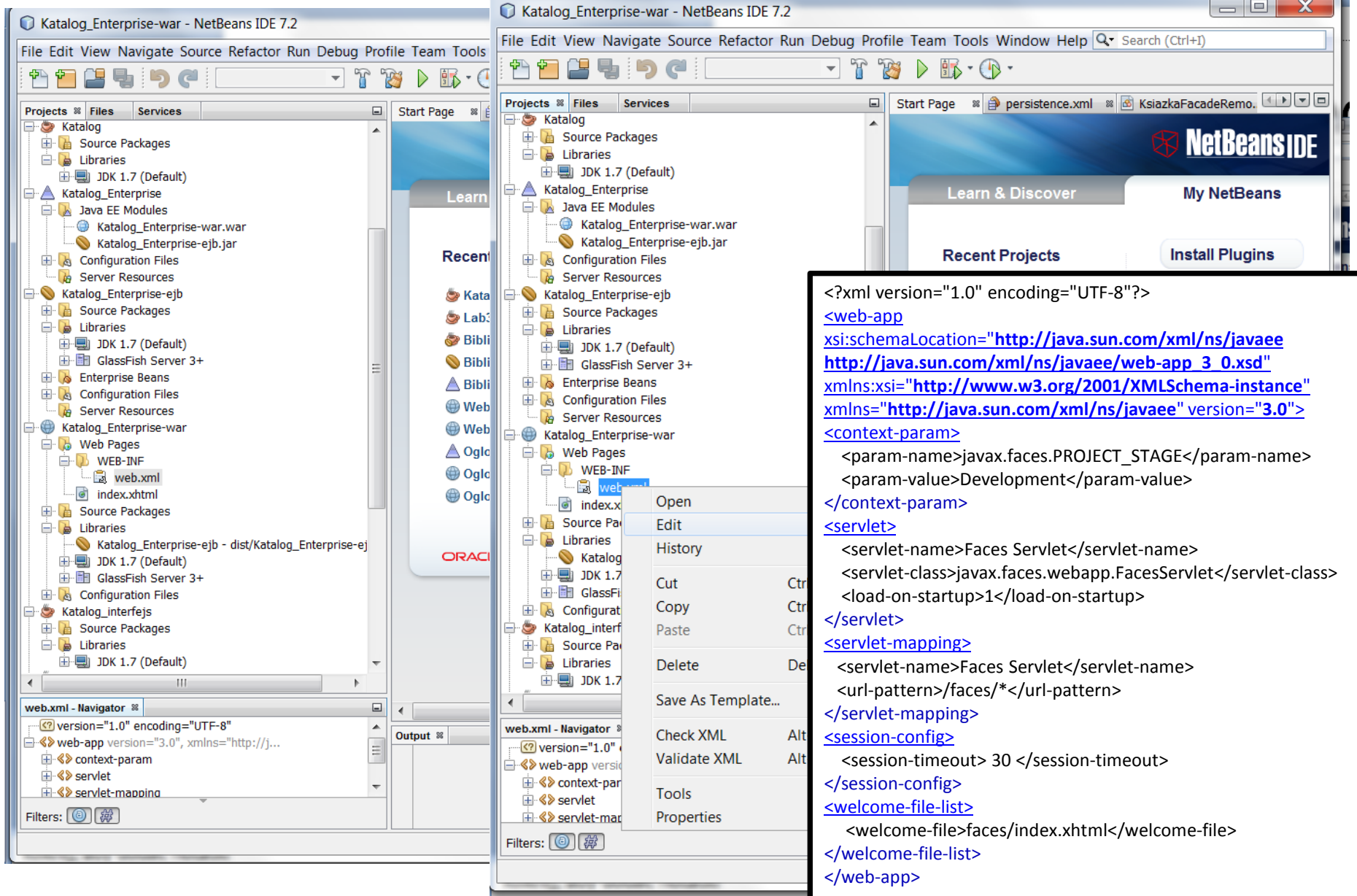
4.1. Zmiana frameworka w module internetowym Katalog_Enterprise-war: po kliknięciu prawym klawiszem myszy nazwę projektu wybór pozycji Properties. Projekt należy **warstwy prezentacji**.



4.2. Zmiana frameworka w module internetowym Katalog_Enterprise-war: wybór pozycji Frameworks (lewa strona) w formularzu Properties oraz kliknięcie na klawisz Add... i wybór w formularzu Add a Framework technologii JavaServer Faces (prawa część strony).



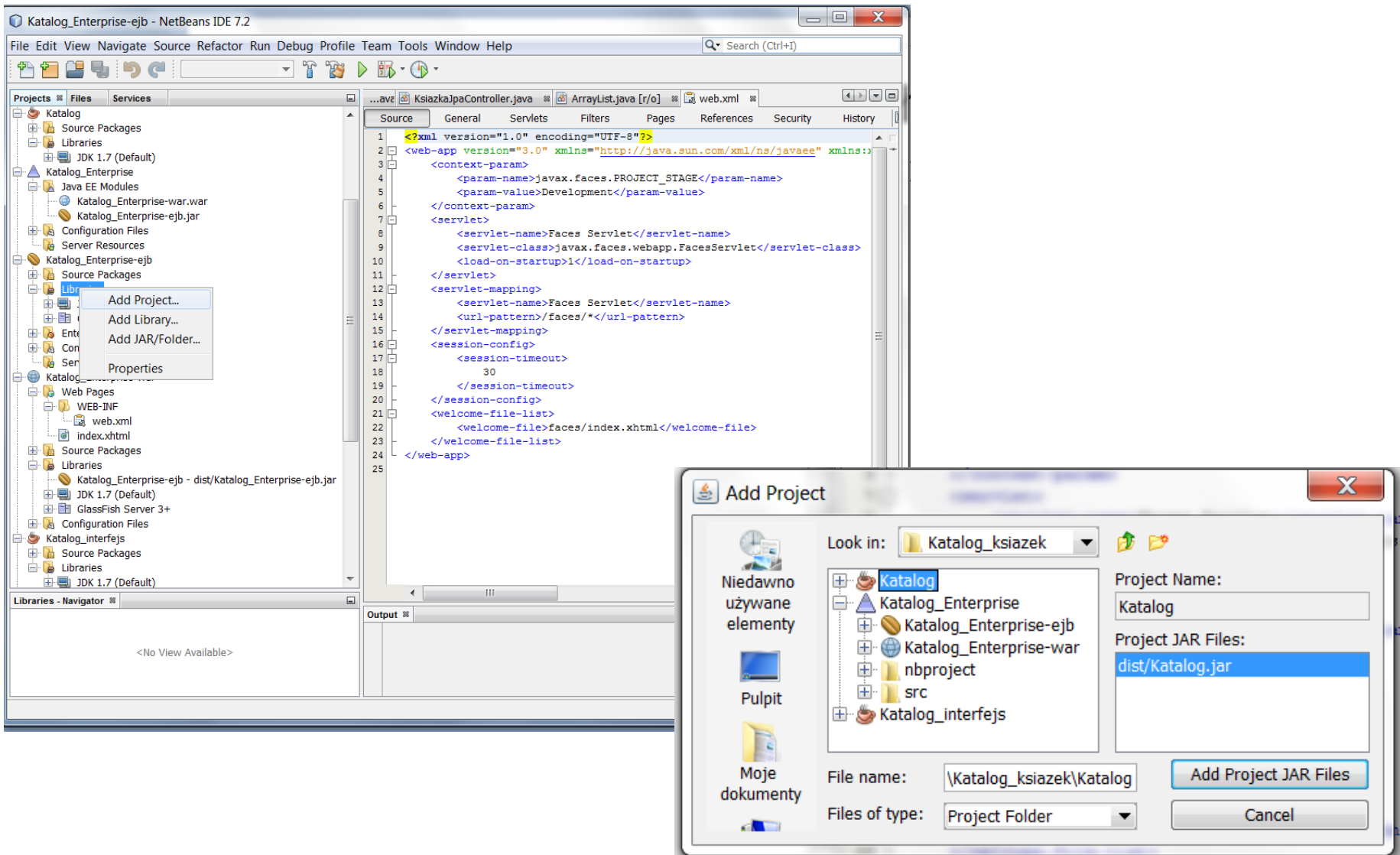
4.3. Sprawdzenie zawartości pliku deskryptora web.xml w module internetowym Katalog_Enterprise-war



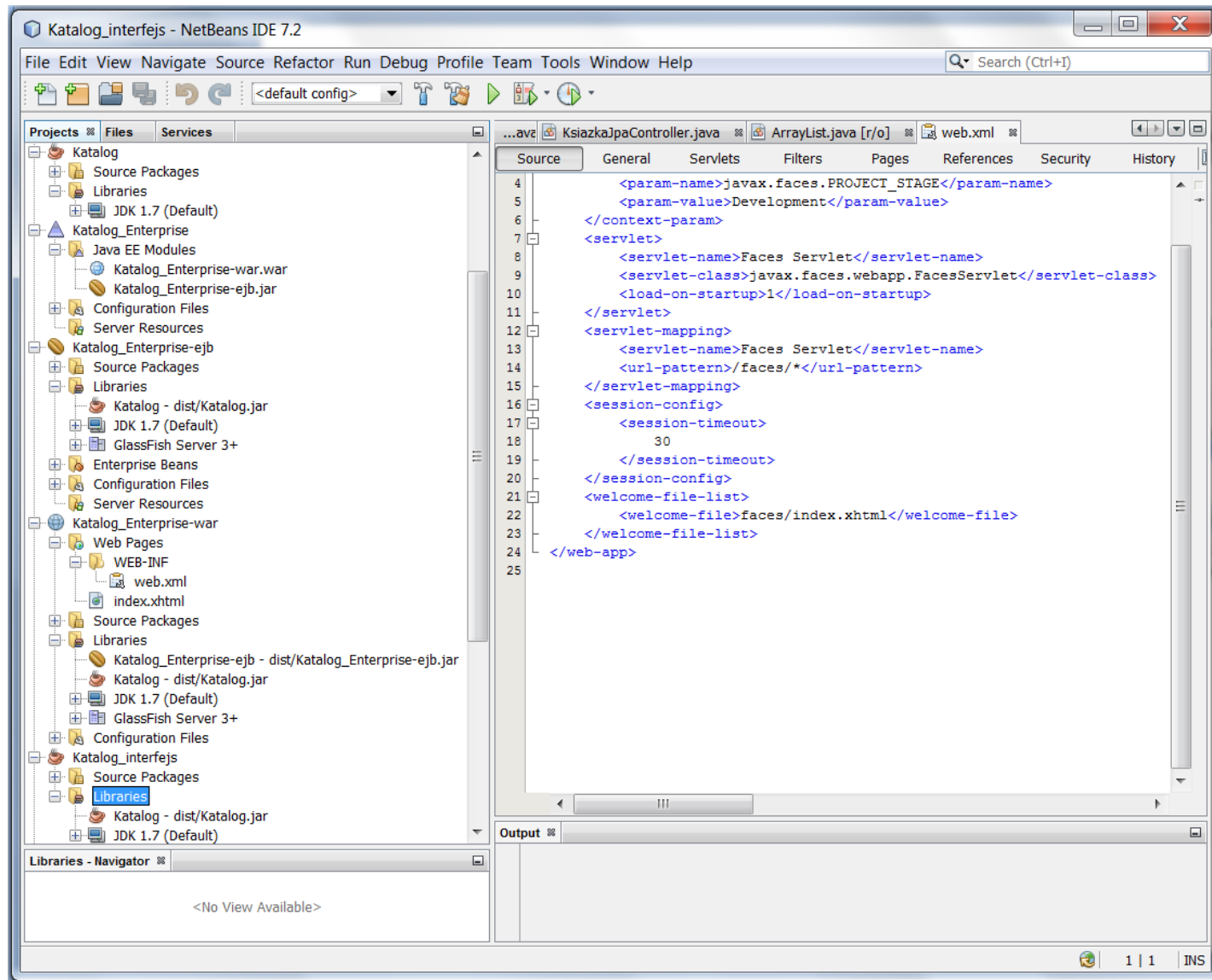
The image shows two overlapping screenshots of the NetBeans IDE 7.2 interface. The left screenshot shows the 'Projects' view with the 'Katalog_Enterprise-war' project selected. The right screenshot shows the 'web.xml' file open in the 'Navigator' window, with a context menu open over the file. The XML content is displayed in the main editor area.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee" version="3.0">
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout> 30 </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>faces/index.xhtml</welcome-file>
  </welcome-file-list>
</web-app>
```


5.1. Dodanie do katalogów Libraries projektów Katalog_Enterprise-ejb, Katalog_interfejs, Katalog_Enterprise-war **projektu Katalog**: po kliknięciu prawym klawiszem myszy na katalog Libraries wybór pozycji Add Project..., następnie wybór projektu Katalog i naciśnięcie klawisza Add Project JAR Files.



5.2. Widok katalogów Libraries w podanych projektach po zakończeniu dodawania projektu Katalog

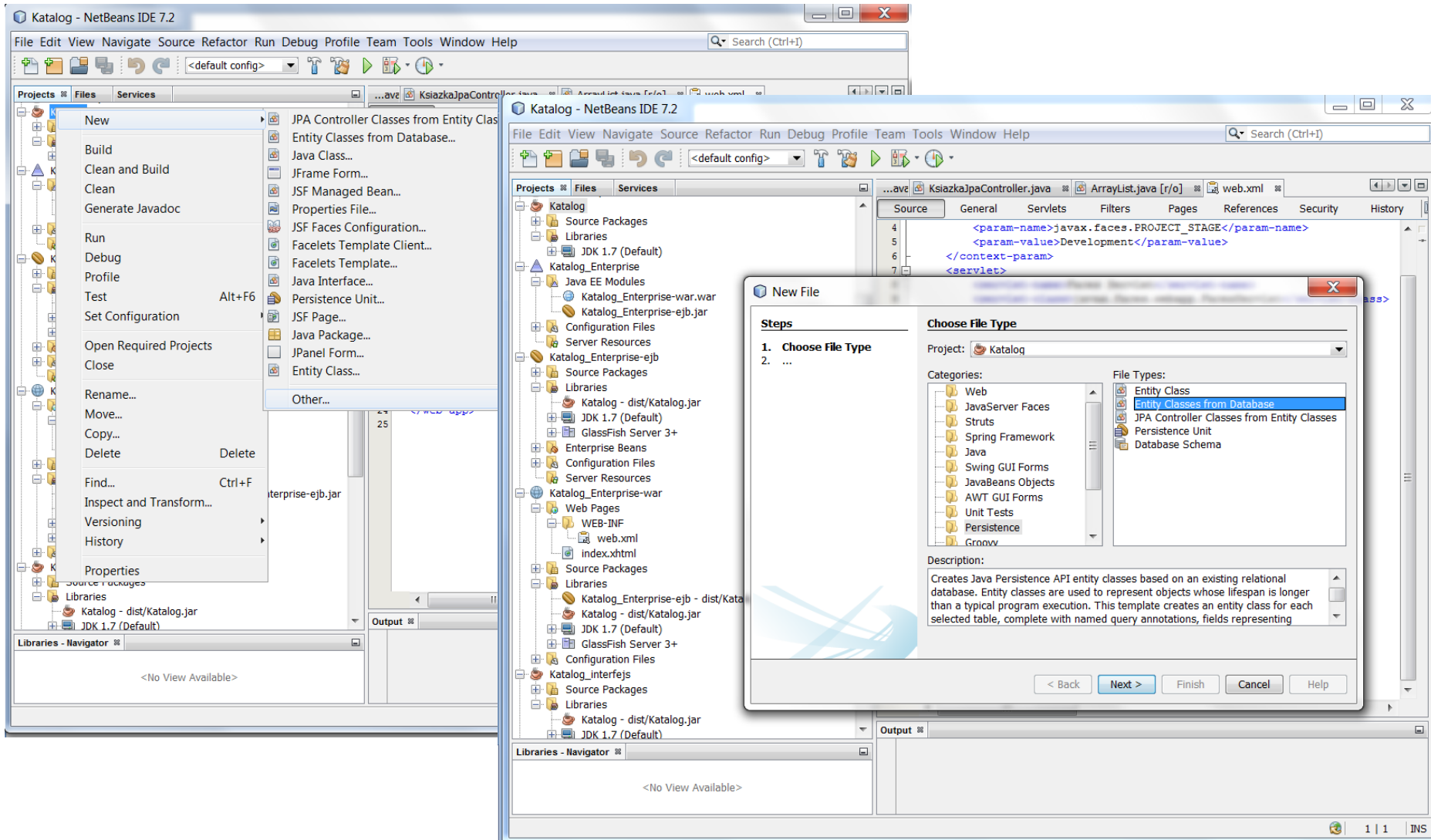


The screenshot displays the NetBeans IDE 7.2 interface. The main window is titled 'Katalog_interfejs - NetBeans IDE 7.2'. The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar shows various icons for file operations and execution. The 'Projects' view on the left shows a tree structure for the 'Katalog' project, including 'Libraries' and 'JDK 1.7 (Default)'. The 'Libraries - Navigator' at the bottom shows '<No View Available>'. The main editor area displays the 'web.xml' file with the following XML content:

```
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/index.xhtml</welcome-file>
23 </welcome-file-list>
24 </web-app>
25
```

The status bar at the bottom right shows '1 | 1 | INS'.

6.1. Dodanie klas typu Entity tworzących obiektowy model danych **warstwy biznesowej** oraz warstwy integracji z bazą danych typu ORM do projektu Katalog, wygenerowanych z bazy danych Katalogksiazek używanej w projekcie internetowym wygenerowanym na podstawie tej bazy danych: należy kliknąć prawym klawiszem na nazwę projektu Katalog w zakładce Projects, wybrać z listy New/Other/Persistence/Entity Classes from Database



6.2. Wybór z listy bazy danych Katalogksiazek, następnie wybór tabel KSIAZKA oraz TYTUL_KSIAZKI przez Add All>> oraz kliknięcie na klawisz Next

The screenshot displays the NetBeans IDE 7.2 interface with the 'New Entity Classes from Database' wizard open in three sequential stages. The background shows the IDE's main window with a project named 'Katalog' and a code editor displaying XML code.

Stage 1: Initial Selection

- Steps:** 1. Choose File Type, 2. Database Tables, 3. Entity Classes, 4. Mapping Options.
- Database Tables:** Database Connection: jdbc:derby://localhost:1527/BibliotekaDB [BibliotekaDB on BI...]. Available Tables: jdbc:derby://localhost:1527/HotelBaza [APP on APP], jdbc:derby://localhost:1527/Katalog_filmow1 [Katalog_filmo...], jdbc:derby://localhost:1527/Katalog_ksiazek [Katalog_ksiaz...], jdbc:derby://localhost:1527/Katalog_ksiazek1 [Katalog_ksiaz...], jdbc:derby://localhost:1527/Katalogksiazek [Katalogksiazek...], jdbc:derby://localhost:1527/Library1 [Library1 on LIBRARY1...], jdbc:derby://localhost:1527/Library_ORM [Library on LIBRAR...].
- Buttons:** << Remove All, Any, Include Related Tables (checked), < Back, Next >, Finish, Cancel, Help.

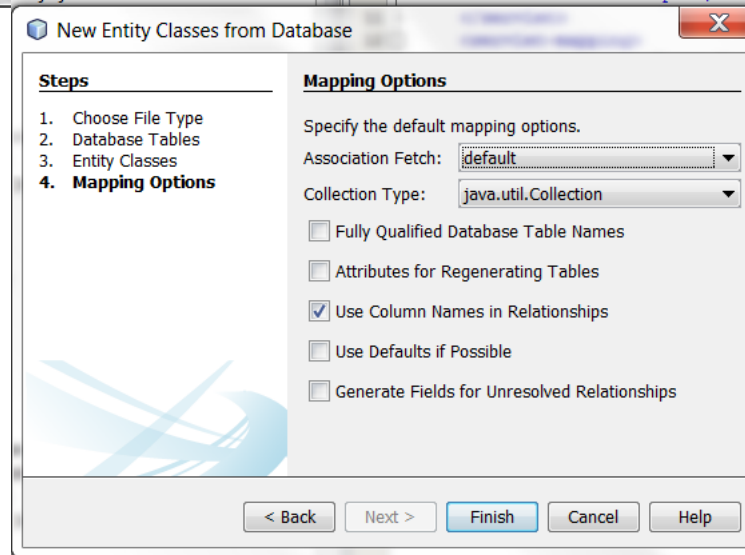
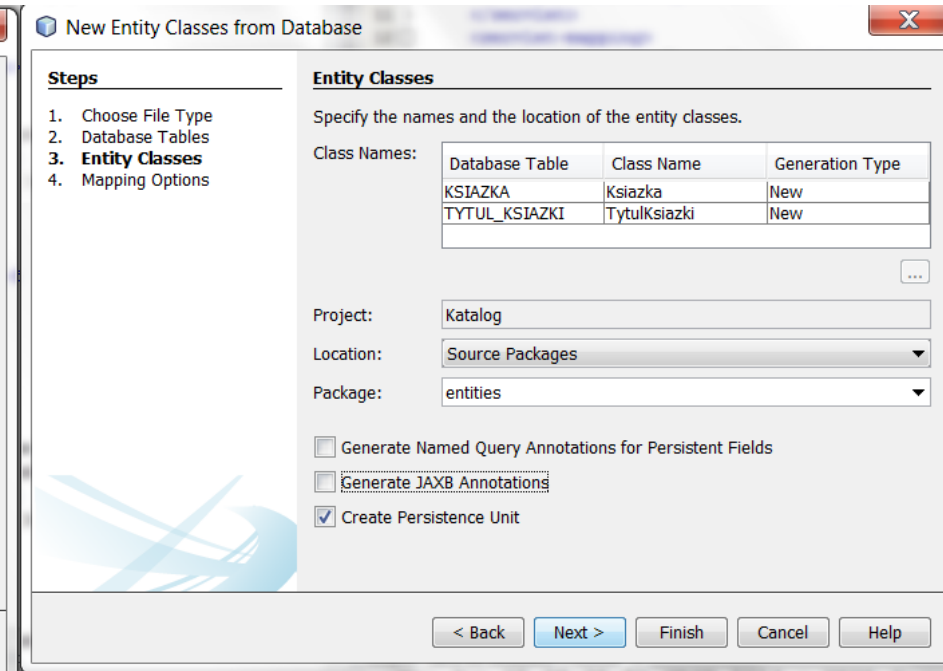
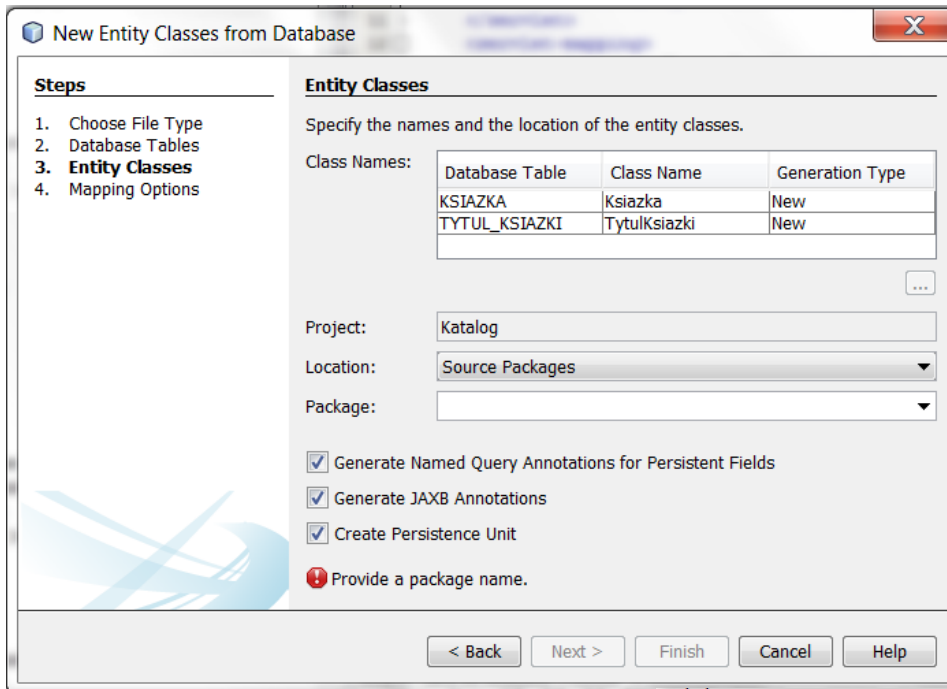
Stage 2: Table Selection

- Steps:** 1. Choose File Type, 2. Database Tables, 3. Entity Classes, 4. Mapping Options.
- Database Tables:** Database Connection: jdbc:derby://localhost:1527/Katalogksiazek [Katalogksiazek on KATALOGKSIAZEK]. Available Tables: KSIAZKA, TYTUL_KSIAZKI. Selected Tables: (empty).
- Buttons:** Add >, < Remove, Add All >>, << Remove All, Any, Include Related Tables (checked), < Back, Next >, Finish, Cancel, Help.

Stage 3: Final Selection

- Steps:** 1. Choose File Type, 2. Database Tables, 3. Entity Classes, 4. Mapping Options.
- Database Tables:** Database Connection: jdbc:derby://localhost:1527/Katalogksiazek [Katalogksiazek on KATALOGKSIAZEK]. Available Tables: (empty). Selected Tables: KSIAZKA, TYTUL_KSIAZKI.
- Buttons:** Add >, < Remove, Add All >>, << Remove All, Any, Include Related Tables (checked), < Back, Next >, Finish, Cancel, Help.

6.3. Podanie nazwy pakietu entities w polu Package, w którym zostaną wygenerowane klasy typu Entity i pozostawienie wybranej opcji Create Persistence Unit. Po naciśnięciu klawisza Next zatwierdzenie procesu klawiszem Finish



6.4. Wygenerowane klasy w pakiecie entities w projekcie Katalog: TytulKsiazki typu Entity z tabeli TYTUL_KSIAZKI oraz Ksiazka z tabeli KSIAZKA

The screenshot displays the NetBeans IDE 7.2 interface. The main editor window shows the source code for `TytulKsiazki.java`. The code includes several imports from the `java.io`, `java.util`, and `javax.persistence` packages. The class `TytulKsiazki` is annotated with `@Entity`, `@Table(name = "TYTUL_KSIAZKI")`, and `@NamedQueries`. It implements the `Serializable` interface and has two attributes: `tytulId` (annotated with `@Id`) and `tytul` (annotated with `@Basic`).

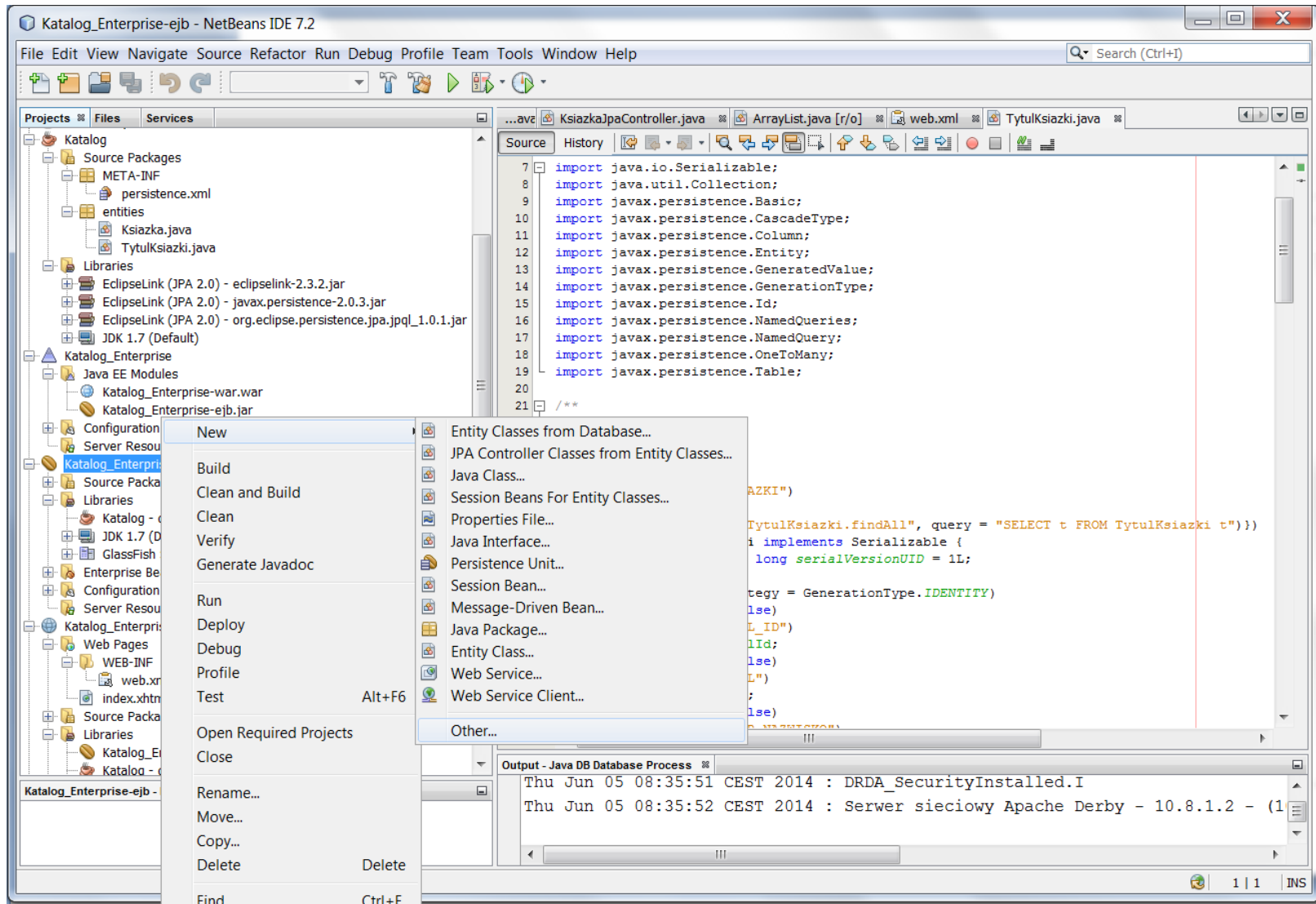
```
7 import java.io.Serializable;
8 import java.util.Collection;
9 import javax.persistence.Basic;
10 import javax.persistence.CascadeType;
11 import javax.persistence.Column;
12 import javax.persistence.Entity;
13 import javax.persistence.GeneratedValue;
14 import javax.persistence.GenerationType;
15 import javax.persistence.Id;
16 import javax.persistence.NamedQueries;
17 import javax.persistence.NamedQuery;
18 import javax.persistence.OneToOne;
19 import javax.persistence.Table;
20
21 /**
22  *
23  * @author Zofia
24  */
25 @Entity
26 @Table(name = "TYTUL_KSIAZKI")
27 @NamedQueries({
28     @NamedQuery(name = "TytulKsiazki.findAll", query = "SELECT t FROM TytulKsiazki t")})
29 public class TytulKsiazki implements Serializable {
30     private static final long serialVersionUID = 1L;
31     @Id
32     @GeneratedValue(strategy = GenerationType.IDENTITY)
33     @Basic(optional = false)
34     @Column(name = "TYTUL_ID")
35     private Integer tytulId;
36     @Basic(optional = false)
37     @Column(name = "TYTUL")
38     private String tytul;
39     @Basic(optional = false)
40     @Column(name = "AUTOR")
41     private String autor;
```

The left sidebar shows the project structure for `Katalog`. The `entities` package contains `Ksiazka.java` and `TytulKsiazki.java`. The `Libraries` section lists dependencies such as `EclipseLink (JPA 2.0)` and `JDK 1.7 (Default)`. The `Output` window at the bottom shows the following messages:

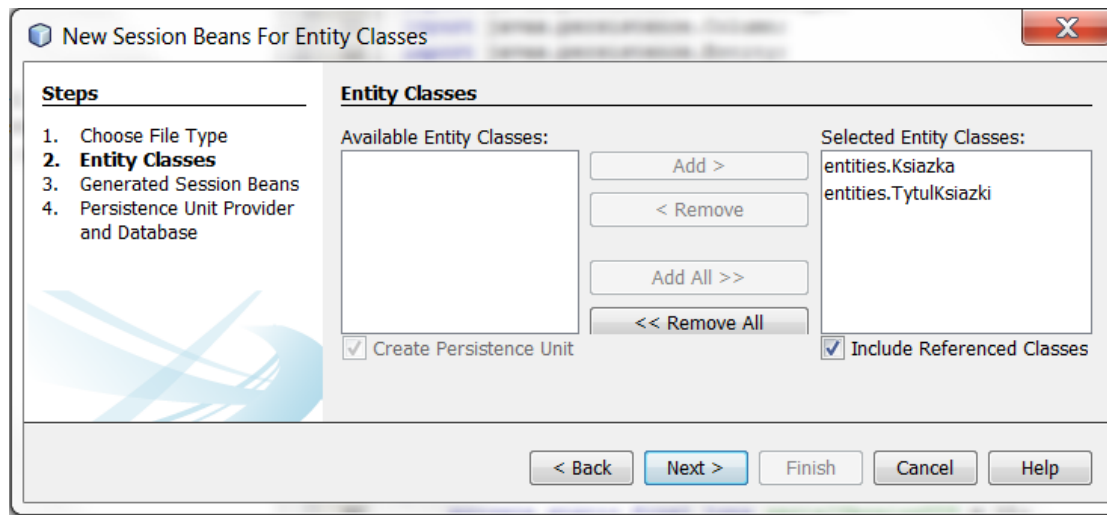
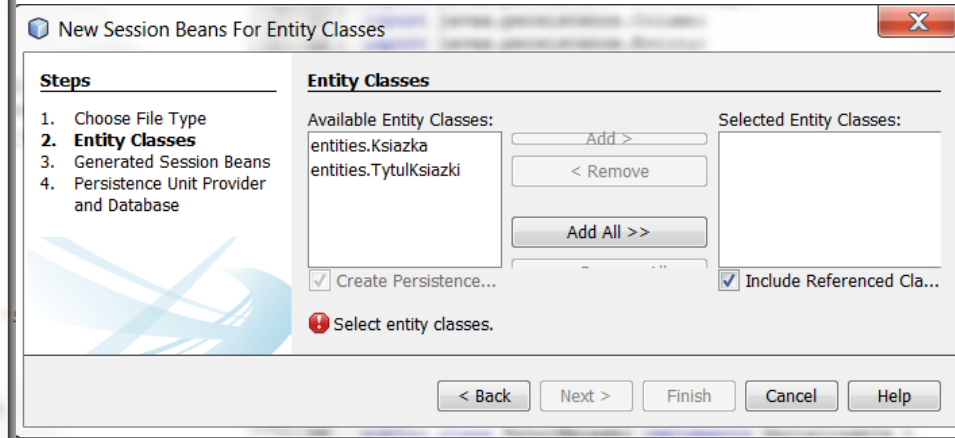
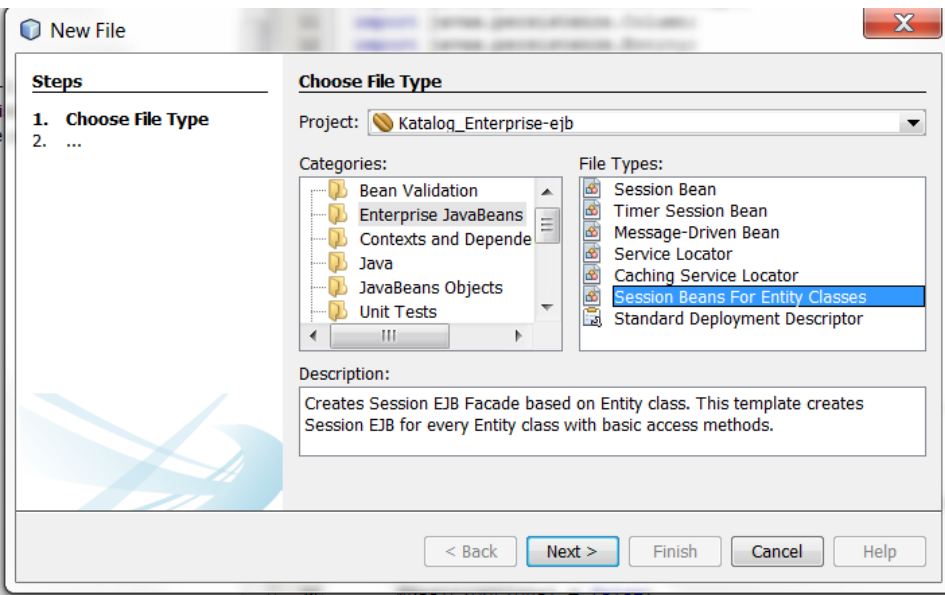
```
Thu Jun 05 08:35:51 CEST 2014 : DRDA_SecurityInstalled.I
Thu Jun 05 08:35:52 CEST 2014 : Serwer sieciowy Apache Derby - 10.8.1.2 - (1
```

The status bar at the bottom right indicates the current page is 1 of 1 and the cursor is in the Insert (INS) mode.

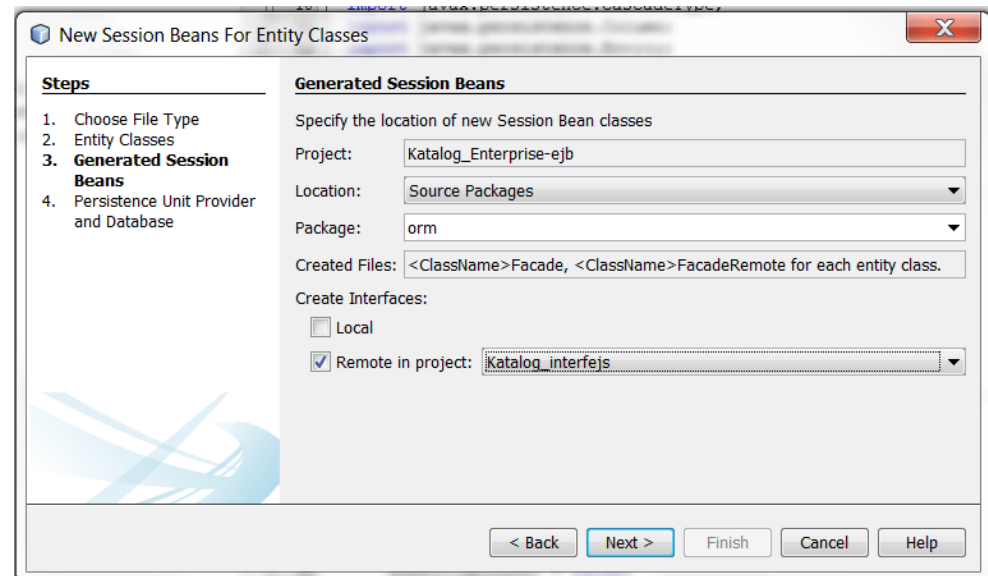
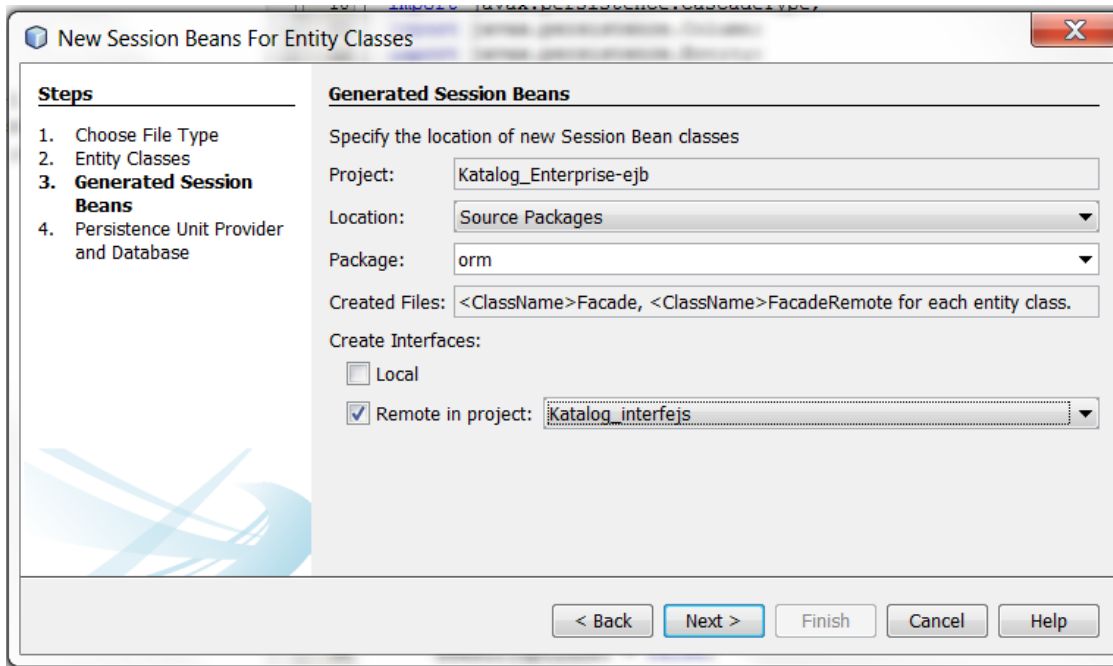
7.1. Wygenerowanie klas typu Session Beans For Entity Classes w module Katalog_Enterprise-ejb, tworzących **warstwę integracji** z bazą danych typu ORM: po kliknięciu prawym klawiszem na nazwę projektu w zakładce Projects wybór pozycji New/Other



7.2. Następnie wybór Enterprise JavaBeans/Session Beans For Entity Classes, wybór klas typu Entity: entites.Ksiazka oraz entities.TytulKsiazki klawiszem Add All i naciśnięcie klawisza Next



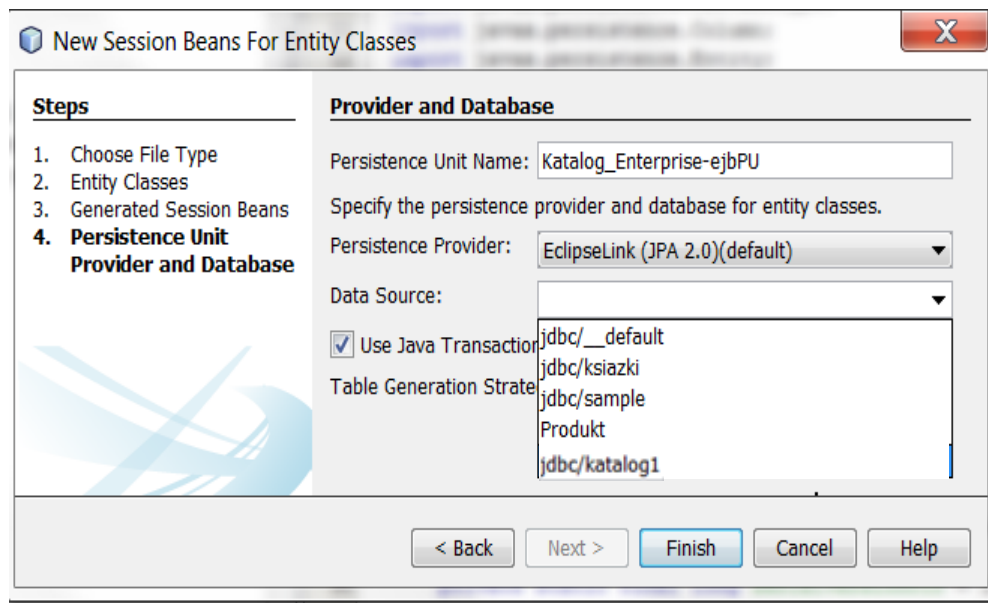
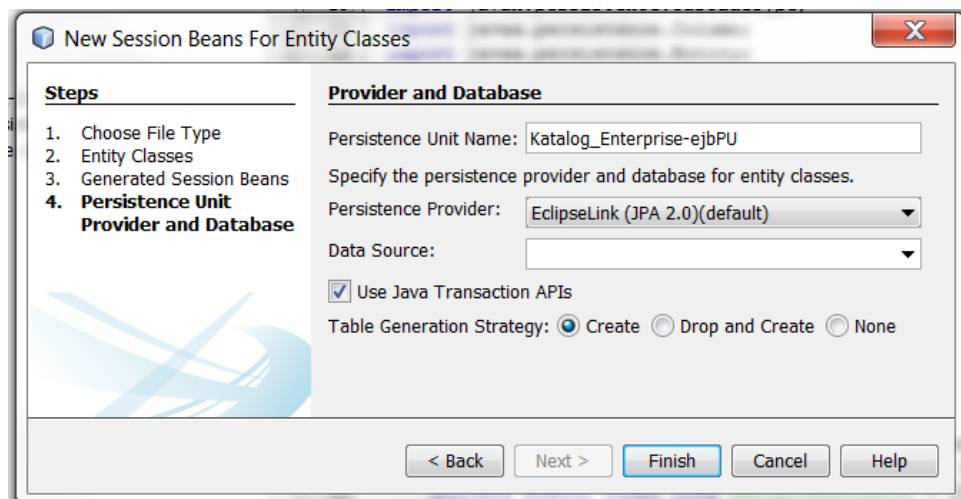
7.3. Utworzenie pakietu orm, zaznaczenie opcji Remote in project, wybór z listy projektu Katalog_interfejs i naciśnięcie klawisza Next



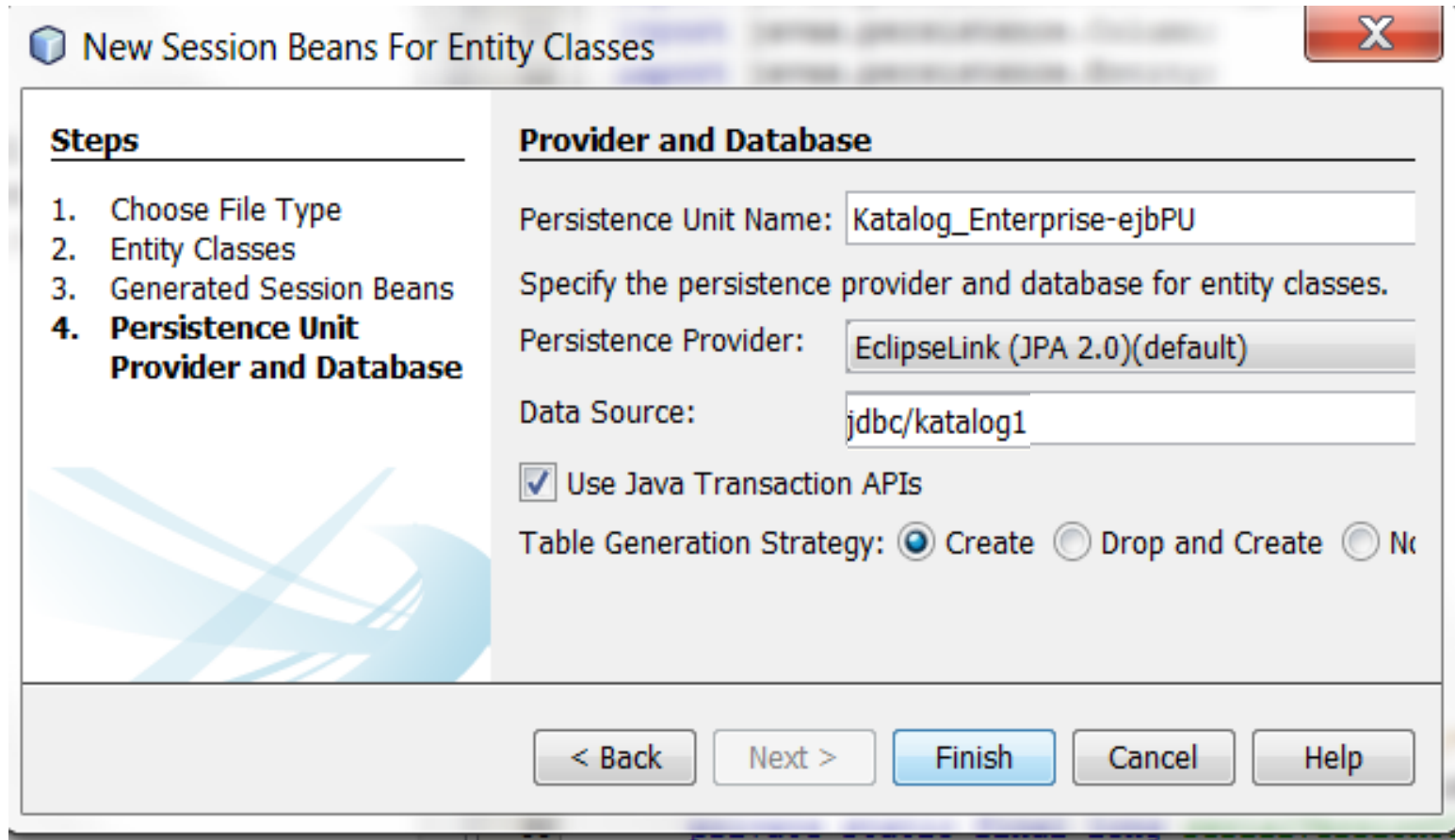
7.4. Rozwinięcie listy Data Source i wybór pozycji **jdbc/katalog1**, łączący z bazą danych **jdbc:derby://localhost:1527/Katalogksiazek [Katalogksiazek on KATALOGKSIAZEK]**

(wykonany wg slajdów 27-33 w pliku

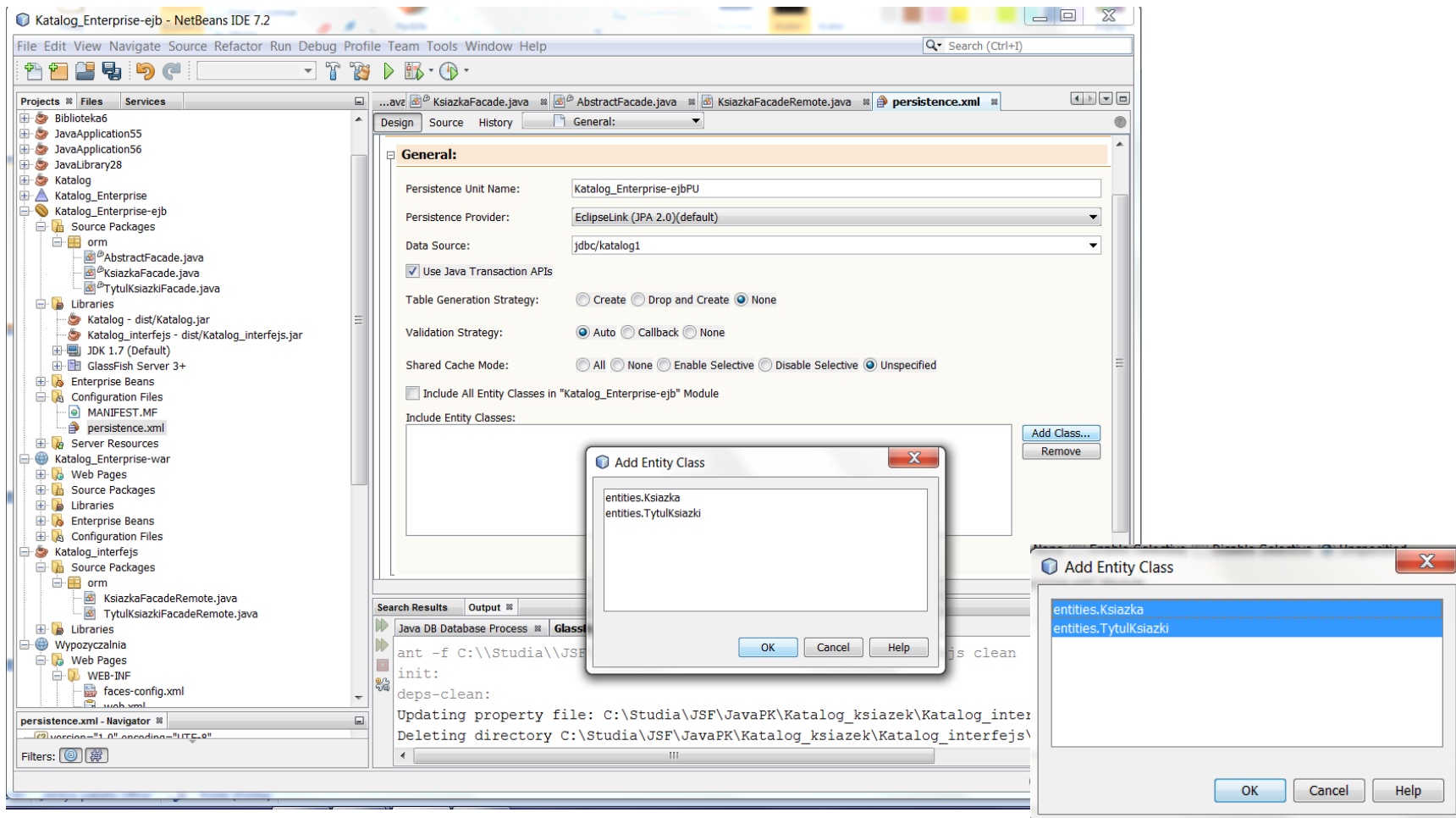
http://zofia.kruckiewicz.staff.iiar.pwr.wroc.pl/wyklady/PiWSI/Aplikacja_internetowa_JSf_z_bazy_danych_2015.pdf)



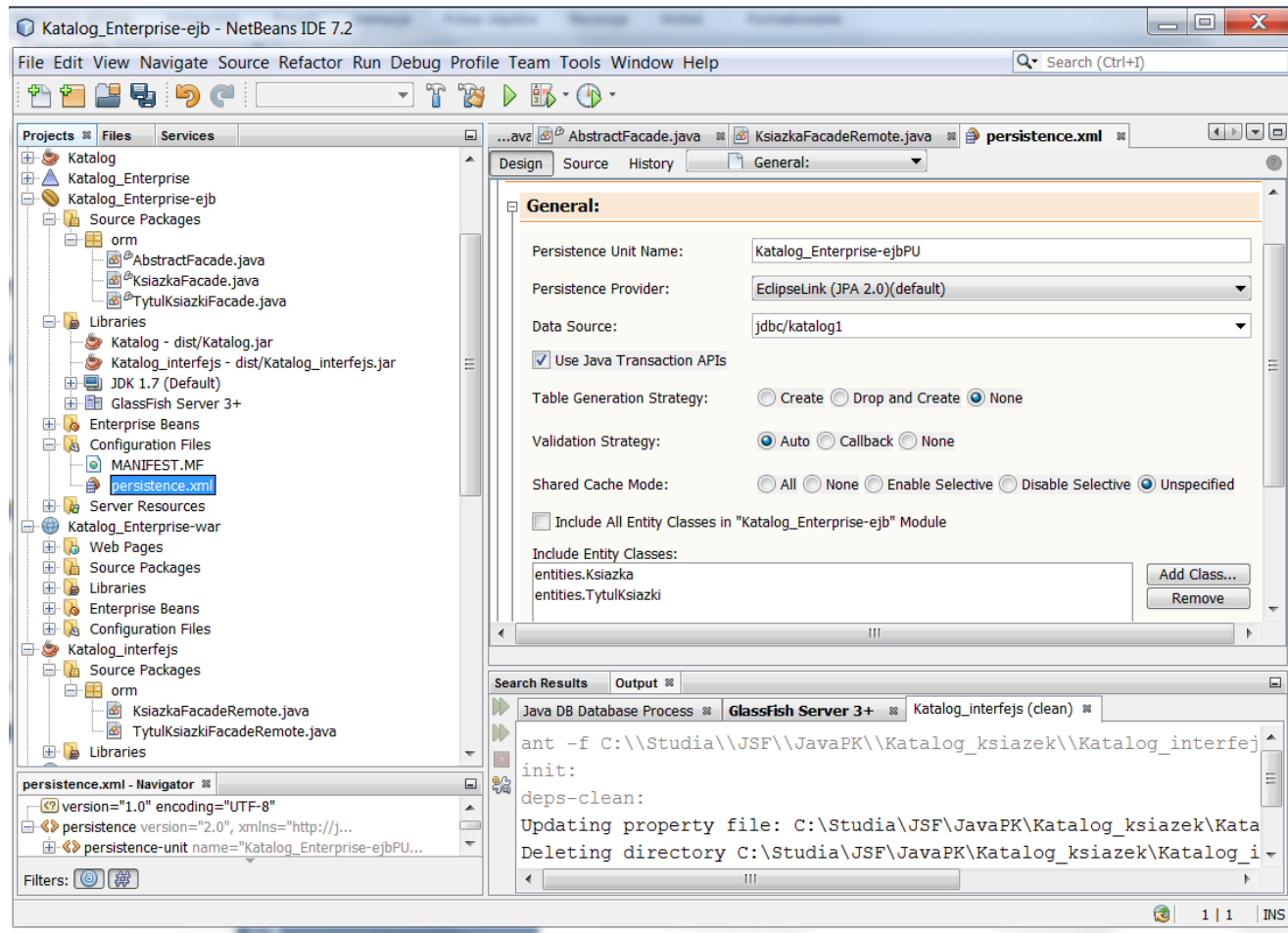
7.5. Zatwierdzenie czynności klawiszem Finish.



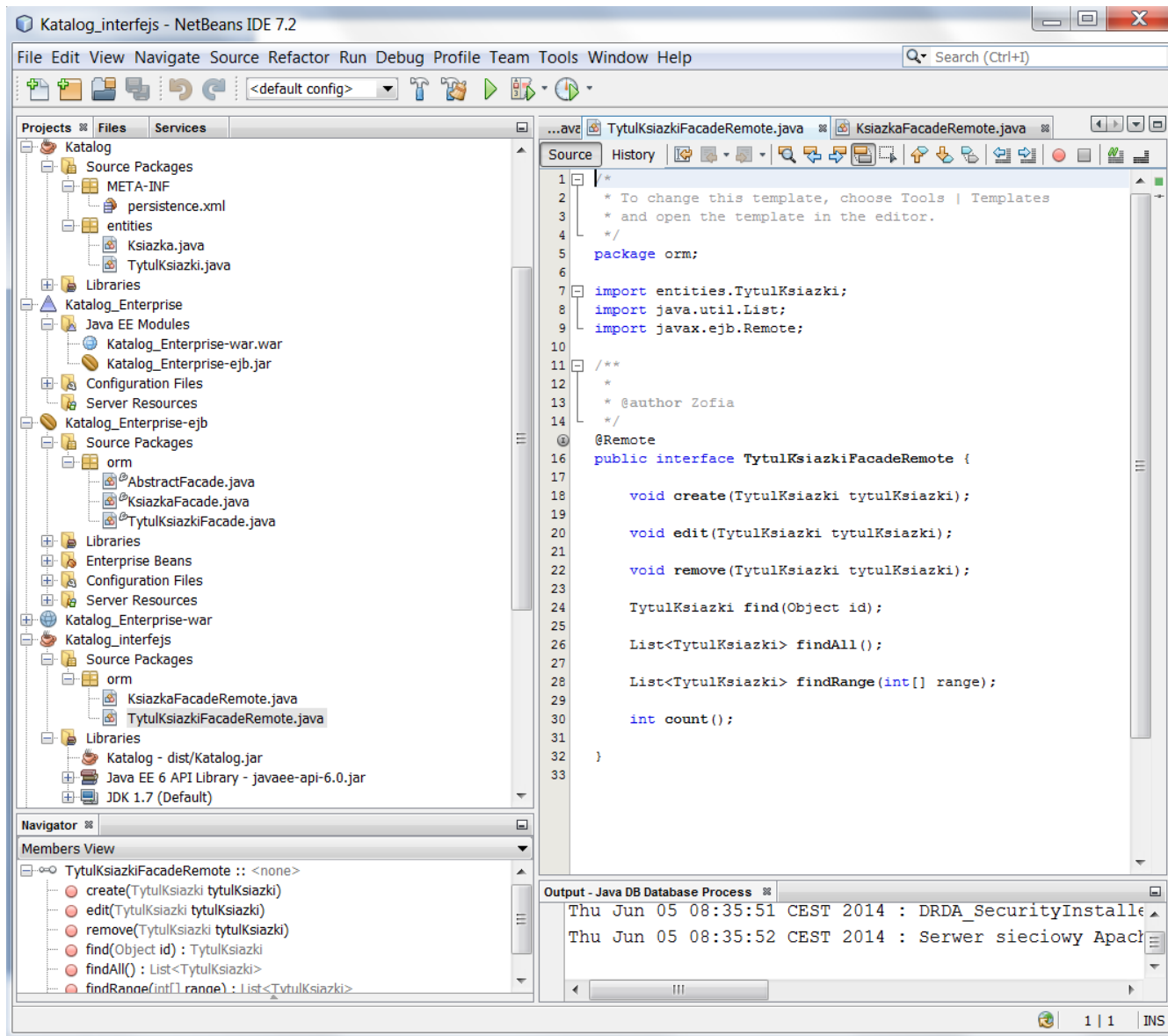
7.6. Ustawienie w module Persistence Unit Table Generation Strategy na None, usunięcie opcji Include All Entities Classes in „Katalog_Enterprise-ejb” Module oraz wybór klas z pakietu entities klawiszem Add Class... Po zaznaczeniu tych klas zatwierdzenie operacji klawiszem OK.



7.7. Zakończenie procesu generowania klas należących do warstwy integracji danych za pomocą technologii ORM.



7.8. Interfejs TytulKsiazkiFacadeRemote komponentu TytulKsiazkiFacade typu Session Bean for Entity class w projekcie Katalog_interfejs.



The screenshot displays the NetBeans IDE 7.2 interface for the 'Katalog_interfejs' project. The left-hand side shows the Project Explorer with the 'orm' package containing 'TytulKsiazkiFacade.java'. The bottom-left pane shows the Members View for the 'TytulKsiazkiFacadeRemote' interface, listing methods: create, edit, remove, find, findAll, and findRange. The main editor window shows the source code of 'TytulKsiazkiFacadeRemote.java'.

```
1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5  package orm;
6
7  import entities.TytulKsiazki;
8  import java.util.List;
9  import javax.ejb.Remote;
10
11  /**
12   *
13   * @author Zofia
14   */
15  @Remote
16  public interface TytulKsiazkiFacadeRemote {
17
18      void create(TytulKsiazki tytulKsiazki);
19
20      void edit(TytulKsiazki tytulKsiazki);
21
22      void remove(TytulKsiazki tytulKsiazki);
23
24      TytulKsiazki find(Object id);
25
26      List<TytulKsiazki> findAll();
27
28      List<TytulKsiazki> findRange(int[] range);
29
30      int count();
31  }
32
33
```

The Output window at the bottom right shows the following log messages:

```
Thu Jun 05 08:35:51 CEST 2014 : DRDA_SecurityInstalle
Thu Jun 05 08:35:52 CEST 2014 : Serwer sieciowy Apache
```

7.9. Interfejs KsiazkaFacadeRemote komponentu KsiazkaFacade typu Session Bean for Entity class w projekcie Katalog_interfejs.

The screenshot displays the NetBeans IDE 7.2 interface. The main window shows the source code of the `KsiazkaFacadeRemote.java` file. The code defines a remote interface for a book facade. The interface includes methods for creating, editing, and removing books, as well as finding books by ID or range, and counting them.

```
1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5  package orm;
6
7  import entities.Ksiazka;
8  import java.util.List;
9  import javax.ejb.Remote;
10
11  /**
12   *
13   * @author Zofia
14   */
15  @Remote
16  public interface KsiazkaFacadeRemote {
17
18      void create(Ksiazka ksiazka);
19
20      void edit(Ksiazka ksiazka);
21
22      void remove(Ksiazka ksiazka);
23
24      Ksiazka find(Object id);
25
26      List<Ksiazka> findAll();
27
28      List<Ksiazka> findRange(int[] range);
29
30      int count();
31
32  }
```

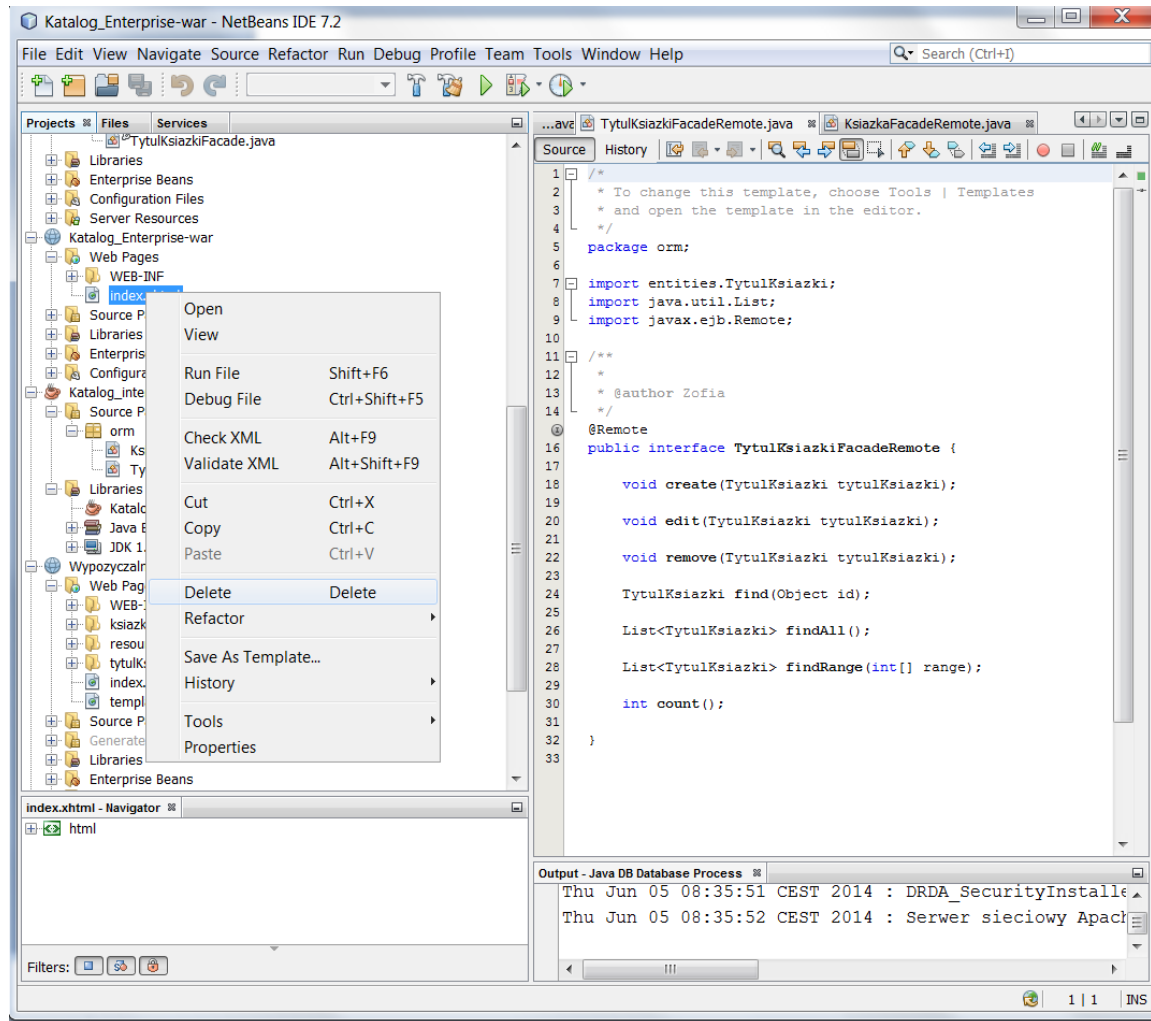
The left sidebar shows the project structure for `Katalog_interfejs`. The `orm` package contains the `KsiazkaFacadeRemote.java` file. The bottom panel shows the `Members View` for the `KsiazkaFacadeRemote` interface, listing the methods: `create(Ksiazka ksiazka)`, `edit(Ksiazka ksiazka)`, `remove(Ksiazka ksiazka)`, `find(Object id) : Ksiazka`, `findAll() : List<Ksiazka>`, and `findRange(int[] range) : List<Ksiazka>`.

The bottom status bar shows the output window with the following messages:

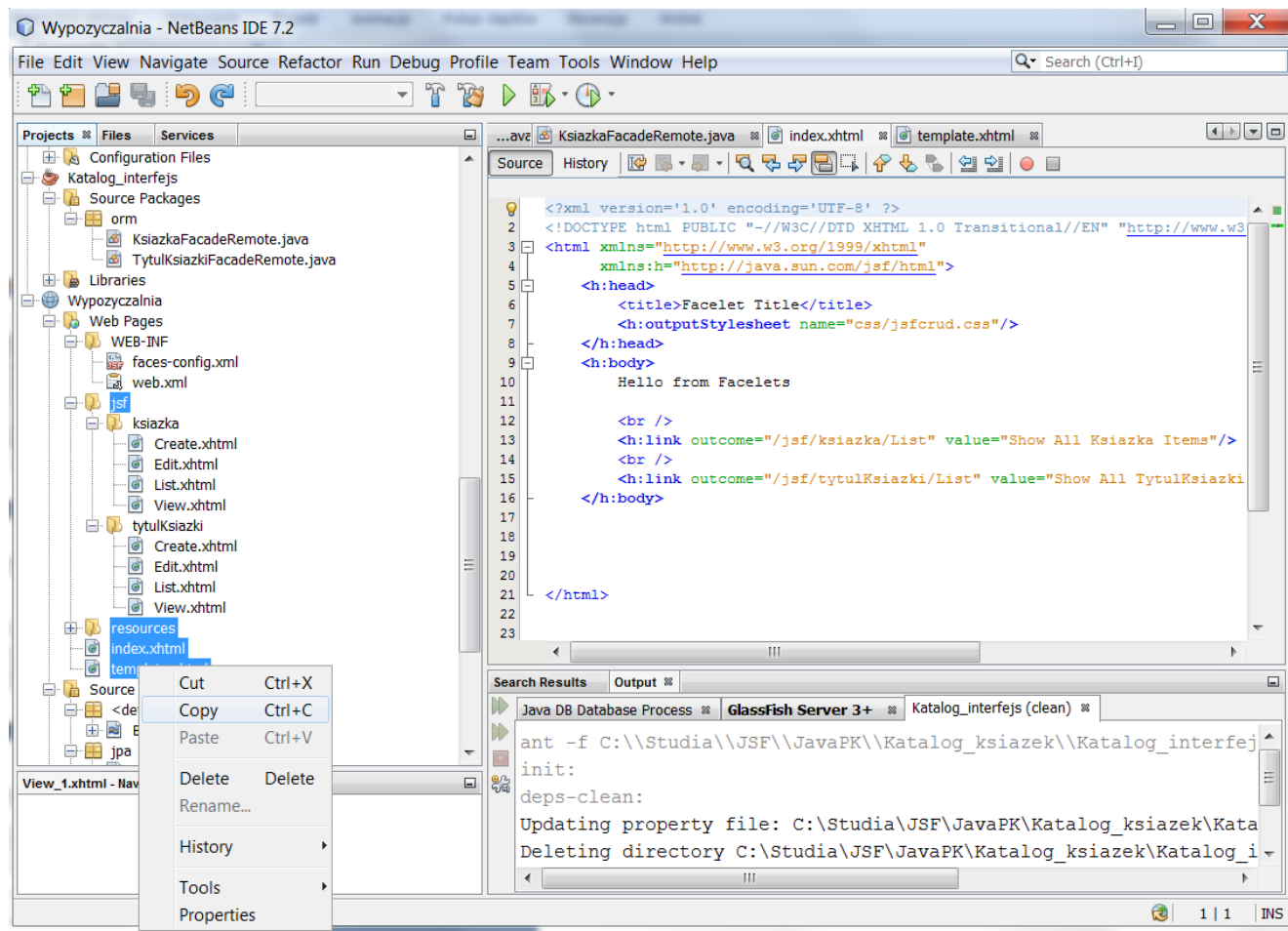
```
Output - Java DB Database Process
Thu Jun 05 08:35:51 CEST 2014 : DRDA_SecurityInstalle
Thu Jun 05 08:35:52 CEST 2014 : Serwer sieciowy Apache
```

The bottom right corner of the IDE shows the page number `2 | 33` and the user `INS`.

8.1. Przebieg wykonania kopii warstwy prezentacji w module Katalog_Enterprise-war z projektu Wypożyczalnia typu Java Web, opartego na technologii JavaServer Faces 2.1, wygenerowanego na podstawie bazy danych Katalogksiazek – na początku usunięcie pliku index.xhtml w module Katalog_Enterprise-war



8.2. Wykonanie kopii katalogów i plików z katalogu Web Pages w projekcie Wypożyczalnia : katalogu jsf, zawierającego strony xhtml, katalogu resources z arkuszami stylów typu css, plik index.xhtml oraz szablon stron template.xhtml.



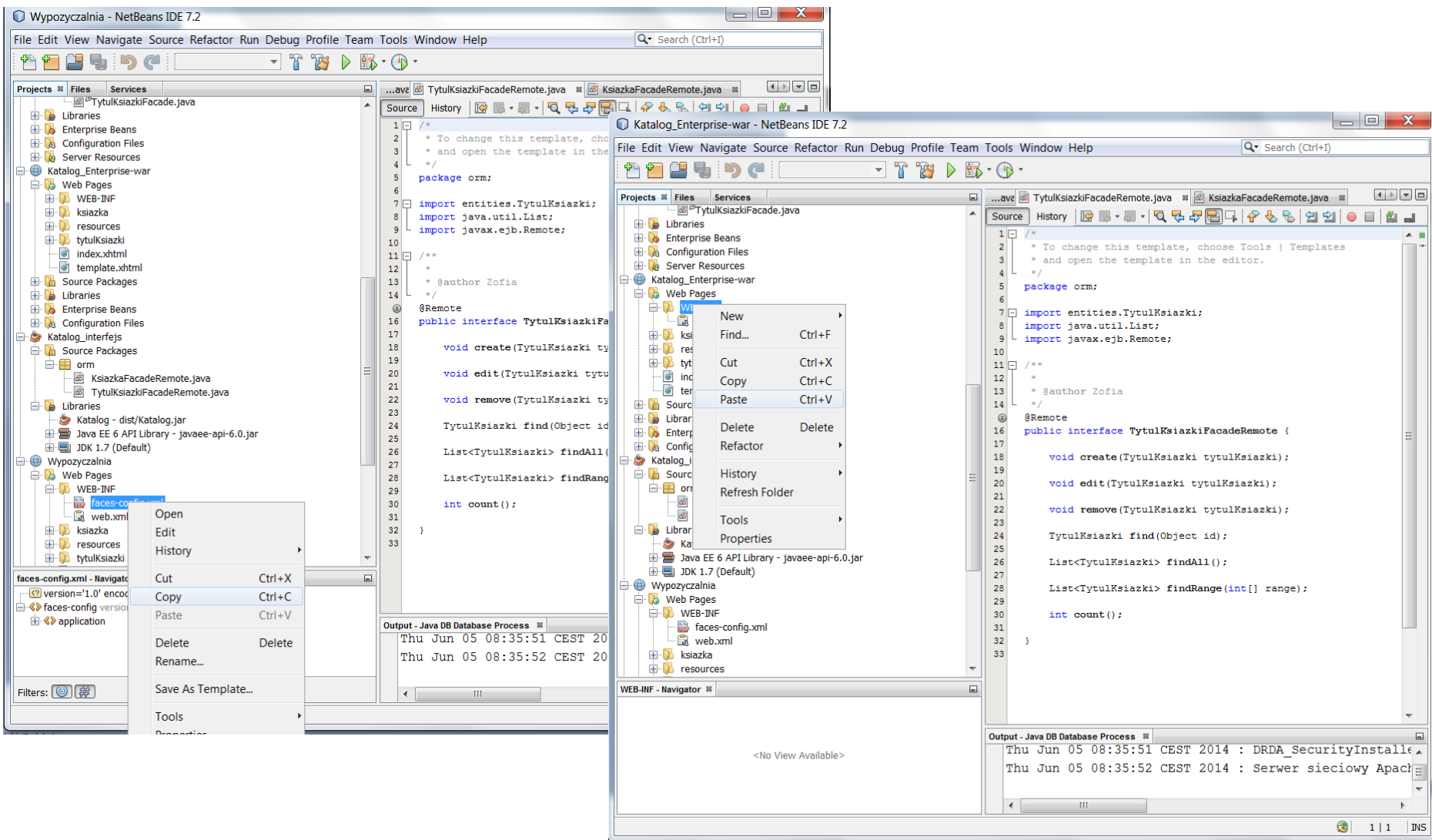
8.3. Wklejenie katalogów i plików podanych w p. 8.3 do katalogu Web Pages w module Katalog_Enterprise-war

The image displays two screenshots of the NetBeans IDE 7.2 interface, illustrating the process of pasting files into a web project.

Left Screenshot: Shows the 'Katalog_Enterprise-war' project in the 'Projects' view. A context menu is open over the 'Web Pages' folder, with the 'Paste' option (Ctrl+V) selected. The 'Source' editor shows the content of 'index.xhtml', which is an XHTML page with a title 'Facelet Title' and a body containing 'Hello from Facelets'. The 'Output' window shows the execution of 'ant -f C:\Studia\JSF\JavaPK\Katalog_ksie...' and the output of 'init:' and 'deps-clean:'. The 'Libraries' view shows various dependencies like 'Katalog - dist/Katalog.jar' and 'GlassFish Server 3+'.

Right Screenshot: Shows the 'Katalog_Enterprise-war' project in the 'Projects' view. The 'Web Pages' folder is expanded, showing the 'WEB-INF' folder containing 'faces-config.xml' and 'web.xml'. The 'jsf' folder contains 'ksiazka' and 'tytulKsiazki' subfolders, each with 'Create.xhtml', 'Edit.xhtml', 'List.xhtml', and 'View.xhtml' files. The 'resources' folder contains 'css', 'index.xhtml', and 'template.xhtml' files. The 'Libraries' view shows the same dependencies as the left screenshot.

8.4. Skopiowanie pliku faces-config.xml, znajdującego się w podkatalogu WEB-INF katalogu Web Pages (z projektu Wypozyczalnia do modułu Katalog_Enterprise-war).



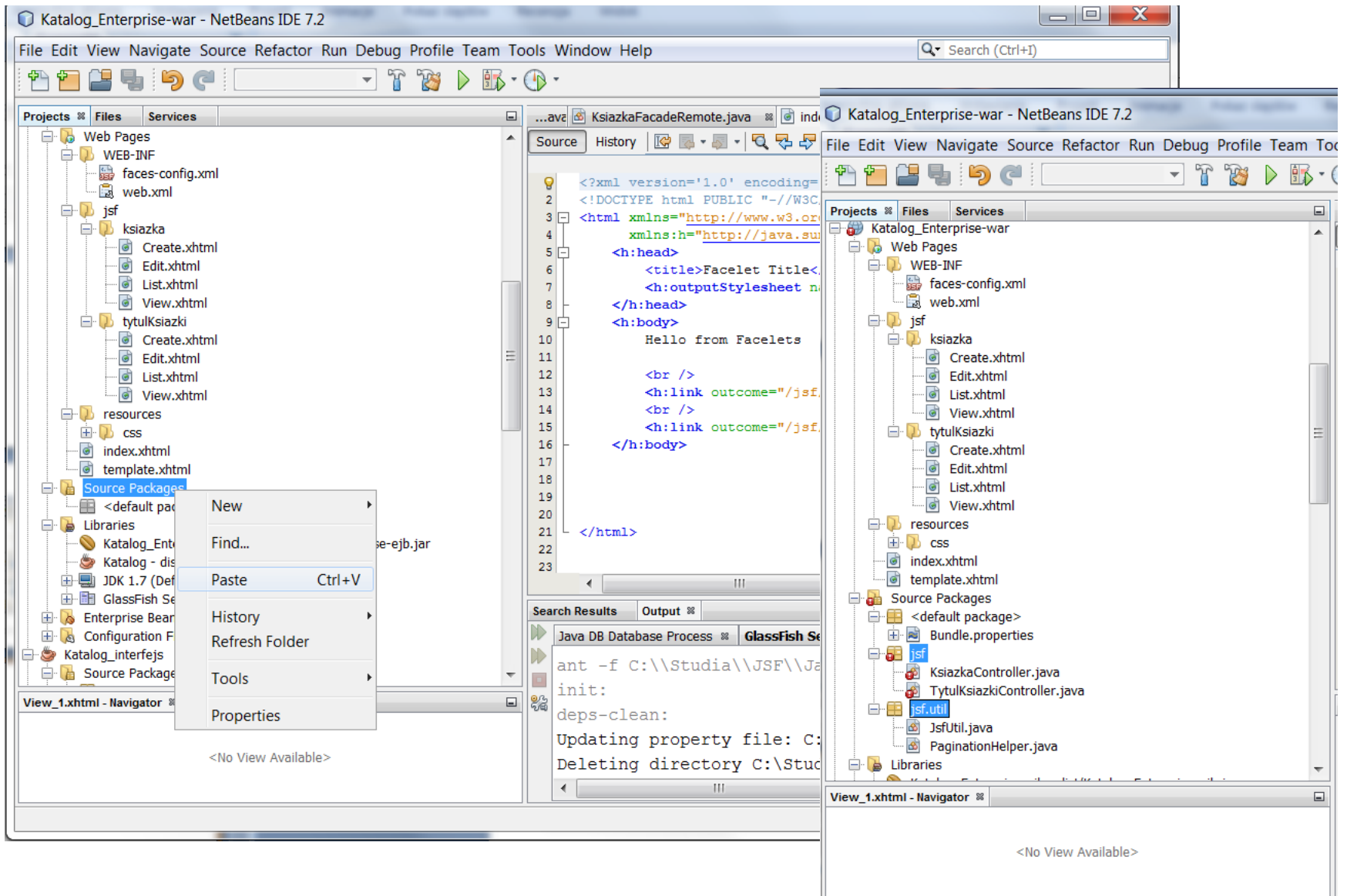
8.5. Skopiowanie następujących pakietów z katalogu Source Packages w projekcie Wypożyczalnia : default package, gdzie znajduje się plik Bundle.properties, pakiet jsf, zawierający komponenty typu Managed Bean oraz pakiet jsf.util, zawierający klasy narzędziowe stronicowania komponentu DataTable oraz obsługi błędów

The screenshot displays the NetBeans IDE 7.2 interface for a project named 'Wypożyczalnia'. The left sidebar shows the project tree with the 'Source Packages' folder expanded, highlighting the '<default package>' and 'jsf' packages. A context menu is open over the '<default package>', showing options like 'Copy' (Ctrl+C) and 'Paste' (Ctrl+V). The main editor window shows the 'KsiazkaFacadeRemote.java' file, which contains the following code:

```
1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5 package orm;
6
7 import entities.TytulKsiazki;
8 import java.util.List;
9 import javax.ejb.Remote;
10
11 /**
12  *
13  * @author Zofia
14  */
15 @Remote
16 public interface TytulKsiazkiFacadeRemote {
17
18     void create(TytulKsiazki tytulKsiazki);
19
20     void edit(TytulKsiazki tytulKsiazki);
21
22     void remove(TytulKsiazki tytulKsiazki);
23
24     TytulKsiazki find(Object id);
25
26     List<TytulKsiazki> findAll();
27
28     List<TytulKsiazki> findRange(int[] range);
29
30     int count();
31
32 }
33
```

The bottom status bar shows the system tray with the date and time 'Thu Jun 05 08:35:51 CEST 2014' and the text 'DRDA_SecurityInstalle' and 'Serwer sieciowy Apach'.

8.6. Wklejenie katalogów i plików podanych w p. 8.5 do katalogu Source Packages w module Katalog_Enterprise-war



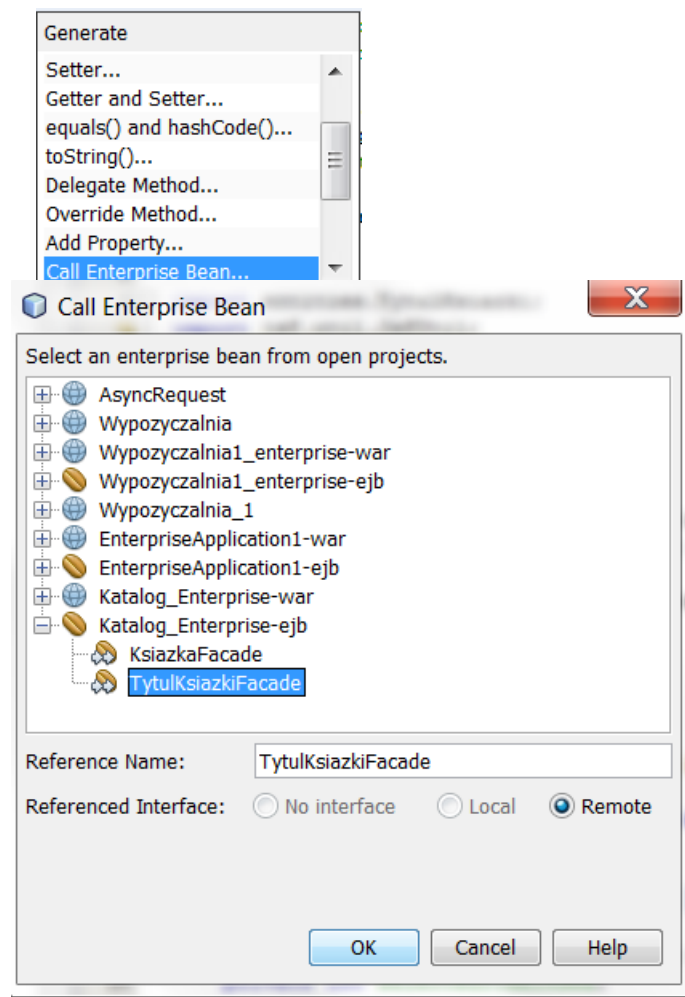
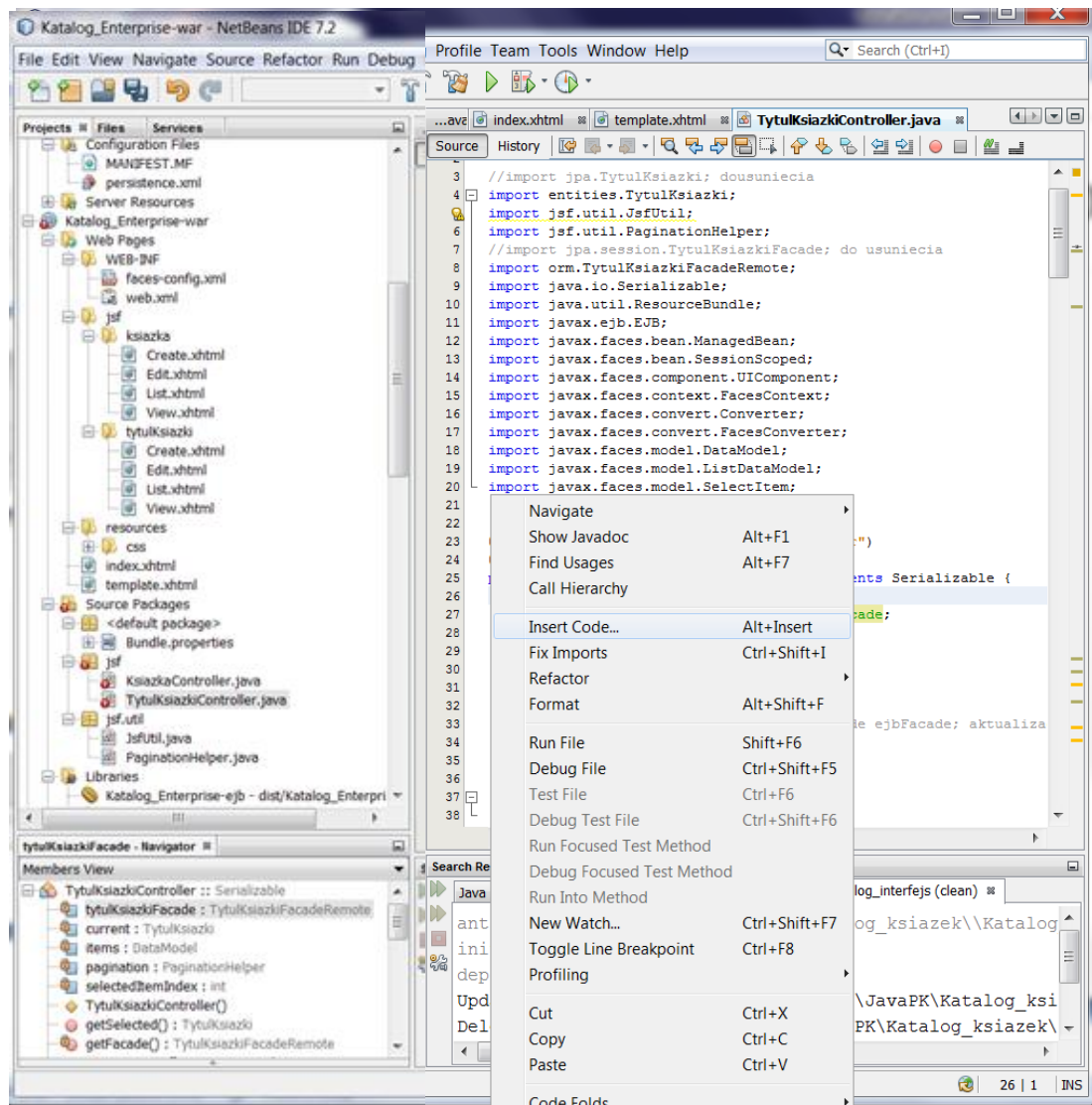
8.7. Usuwanie błędów kompilacji w klasie TytułKsiążkiController - po kliknięciu prawym klawiszem na klasę TytułKsiążkiController należy wybrać pozycję Fix Imports i wybrać właściwe klasy w okienku Fix All Imports.

The screenshot displays the NetBeans IDE interface for the 'Katalog_Enterprise-war' project. The main editor window shows the source code of the `TytułKsiążkiController.java` file. A context menu is open over the `TytułKsiążki` class name, with the 'Fix Imports' option selected. The 'Fix All Imports' dialog box is in the foreground, prompting the user to select the fully qualified name to use in the import statement. The dialog shows the following import statements:

- TytułKsiążki (selected): `entities.TytułKsiążki`
- TytułKsiążkiFacade (selected): `jpa.session.TytułKsiążkiFacade`
- Remove unused imports: `jpa.session.TytułKsiążkiFacade`
- `orm.TytułKsiążkiFacade`

The dialog also includes 'OK' and 'Cancel' buttons. The background shows the IDE interface with the source code of `TytułKsiążkiController.java` and the project structure in the left sidebar.

8.8. Dodanie dostępu do komponentów z warstwy integracji - po kliknięciu prawym klawiszem na klasę TytułKsiążkiController należy wybrać pozycję Insert Code..., wybrać pozycję Call Enterprise Bean... w okienku Generate. Następnie, należy wybrać komponent typu TytułKsiążkiFacade w oknie Select an enterprise bean from open projects.



8.9. Poprawa kodu w klasie TytulKsiazkiController wynikająca ze zmiany nazwy dodanego obiektu oraz zmiany typu tego obiektu dodanego za pomocą adnotacji.

The image displays two side-by-side screenshots of the NetBeans IDE 7.2 interface, illustrating the refactoring process of the `TytulKsiazkiController` class. The left screenshot shows the original code, and the right screenshot shows the code after refactoring.

Left Screenshot (Original Code):

```
import orm.KsiazkaFacadeRemote;
import orm.TytulKsiazkiFacadeRemote;

@ManagedBean(name = "tytulKsiazkiController")
@SessionScoped
public class TytulKsiazkiController implements Serializable {
    @EJB
    private TytulKsiazkiFacadeRemote tytulKsiazkiFacade;

    private TytulKsiazki current;
    private DataModel items = null;
    // @EJB aktualizacja
    // private jpa.session.TytulKsiazkiFacade ejbFacade;
    private PaginationHelper pagination;
    private int selectedItemIndex;

    public TytulKsiazkiController() {
    }

    public TytulKsiazki getSelected() {
        if (current == null) {
            current = new TytulKsiazki();
            selectedItemIndex = -1;
        }
        return current;
    }

    private TytulKsiazkiFacadeRemote getFacade() {
        return tytulKsiazkiFacade;
    }

    public PaginationHelper getPagination() {
        if (pagination == null) {
            pagination = new PaginationHelper(10) {
                @Override
    
```


8.11. Usuwanie błędów kompilacji w klasie KsiazkaController - po kliknięciu prawym klawiszem na klasę KsiazkaController należy wybrać pozycję Fix Imports i wybrać właściwe klasy w okienku Fix All Imports.

The screenshot displays the NetBeans IDE interface. The main editor shows the source code of `KsiazkaController.java`. A context menu is open over the class declaration, with `Fix Imports` selected. To the right, the `Fix All Imports` dialog box is visible, showing a list of imports and a dropdown menu for selecting the correct class to import.

The code in the editor includes the following imports:

```
package jsf;
//import jpa.Ksiazka; aktualizacja
import jsf.util.JsfUtil;
import jsf.util.PaginationHelper;
//import jpa.session.KsiazkaFacade; aktualizacja
import java.io.Serializable;
import java.util.ResourceBundle;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.FacesConverter;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import javax.faces.model.SelectItem;
```

The `Fix All Imports` dialog box shows the following imports:

- `orm.KsiazkaFacade`
- `entities.Ksiazka`

The context menu options include:

- Navigate
- Show Javadoc (Alt+F1)
- Find Usages (Alt+F7)
- Call Hierarchy
- 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)
- Toggle Line Breakpoint (Ctrl+F8)
- Profiling
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)

8.12. Dodanie dostępu do komponentów z warstwy integracji - po kliknięciu prawym klawiszem na klasę KsiazkaController należy wybrać pozycję Insert Code..., wybrać pozycję Call Enterprise Bean... w okienku Generate. Następnie, należy wybrać komponent typu KsiazkaFacade w oknie Select an enterprise bean from open projects.

The screenshot displays the NetBeans IDE 7.2 interface. The main editor shows the source code of the `KsiazkaController` class, which implements `Serializable`. The class contains several private fields and methods, including `getSelected()` and `getFacade()`. A context menu is open over the `getFacade()` method, with the `Insert Code...` option selected. This opens the `Generate` dialog, where the `Call Enterprise Bean...` option is highlighted. This leads to the `Call Enterprise Bean` dialog, which prompts the user to select an enterprise bean from open projects. The `KsiazkaFacade` component is selected in the list. The `Reference Name` is set to `KsiazkaFacade`, and the `Referenced Interface` is set to `Remote`. The `OK` button is visible at the bottom of the dialog.

```
package jsf;

import entities.Ksiazka;
import jsf.util.JsfUtil;
import jsf.util.PaginationHelper;
import java.io.Serializable;
import java.util.ResourceBundle;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.FacesConverter;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import javax.faces.model.SelectItem;
import orm.KsiazkaFacade;

@ManagedBean(name = "ksiazkaController")
@SessionScoped
public class KsiazkaController implements Serializable {

    private Ksiazka current;
    private DataModel items = null;
    private PaginationHelper pagination;
    private int selectedRowIndex;

    public KsiazkaController() {
    }

    public Ksiazka getSelected() {
        if (current == null) {
            current = new Ksiazka();
            selectedRowIndex = -1;
        }

        return current;
    }

    private KsiazkaFacade getFacade() {
        return ejbFacade;
    }
}
```

8.13. Poprawa kodu w klasie KsiazkaController wynikająca ze zmiany nazwy dodanego obiektu oraz zmiany typu obiektu dodanego za pomocą adnotacji.

The image displays two side-by-side screenshots of the NetBeans IDE, illustrating the refactoring process of the `KsiazkaController` class. The left screenshot shows the original code, and the right screenshot shows the code after refactoring.

Original Code (Left Screenshot):

```
package jsf;

import entities.Ksiazka;
import jsf.util.JsfUtil;
import jsf.util.PaginationHelper;
import java.io.Serializable;
import java.util.ResourceBundle;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.FacesConverter;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import javax.faces.model.SelectItem;
import orm.KsiazkaFacade;
import orm.KsiazkaFacadeRemote;

@ManagedBean(name = "ksiazkaController")
@SessionScoped
public class KsiazkaController implements Serializable {
    @EJB
    private KsiazkaFacadeRemote ksiazkaFacade;
    private Ksiazka current;
    private DataModel items = null;
    private PaginationHelper pagination;
    private int selectedItemIndex;

    public KsiazkaController() {
    }

    public Ksiazka getSelected() {
        if (current == null) {
            current = new Ksiazka();
            selectedItemIndex = -1;
        }
        return current;
    }

    private KsiazkaFacade getFacade() {
        return ksiazkaFacade;
    }
}
```

Refactored Code (Right Screenshot):

```
package jsf;

import entities.Ksiazka;
import jsf.util.JsfUtil;
import jsf.util.PaginationHelper;
import java.io.Serializable;
import java.util.ResourceBundle;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.FacesConverter;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import javax.faces.model.SelectItem;
import orm.KsiazkaFacade;
import orm.KsiazkaFacadeRemote;

@ManagedBean(name = "ksiazkaController")
@SessionScoped
public class KsiazkaController implements Serializable {
    @EJB
    private KsiazkaFacadeRemote ejbFacade;
    private Ksiazka current;
    private DataModel items = null;
    private PaginationHelper pagination;
    private int selectedItemIndex;

    public KsiazkaController() {
    }

    public Ksiazka getSelected() {
        if (current == null) {
            current = new Ksiazka();
            selectedItemIndex = -1;
        }
        return current;
    }

    private KsiazkaFacadeRemote getFacade() {
        return ejbFacade;
    }
}
```

Red arrows in both screenshots indicate the changes: the field `ksiazkaFacade` is renamed to `ejbFacade` and annotated with `@EJB`, and the method `getFacade()` is updated to return `ejbFacade`.

8.14. Poprawa importów w klasie KsiazkaController.

The image displays three overlapping windows of the NetBeans IDE, illustrating the process of cleaning up imports in the `KsiazkaController.java` file.

- Top Left Window:** Shows the initial state of the code. The `import` statements are cluttered and not sorted. A context menu is open over the imports, with the **Organize Imports** option selected.
- Top Right Window:** Shows the result of the `Organize Imports` action. The imports are now sorted alphabetically and only the necessary ones remain.
- Bottom Window:** Shows the `KsiazkaController.java` file with the refactored imports and the class implementation. The class implements `Serializable` and has several private fields and methods.

```
package jsf;

import entities.Ksiazka;
import jsf.util.JsfUtil;
import jsf.util.PaginationHelper;
import java.io.Serializable;
import java.util.ResourceBundle;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.FacesConverter;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import javax.faces.model.SelectItem;
import KsiazkaFacadeRemote;

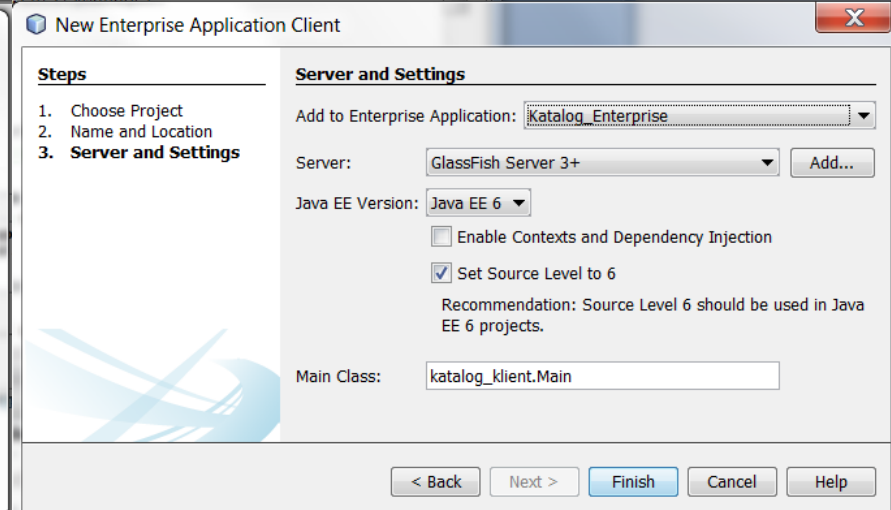
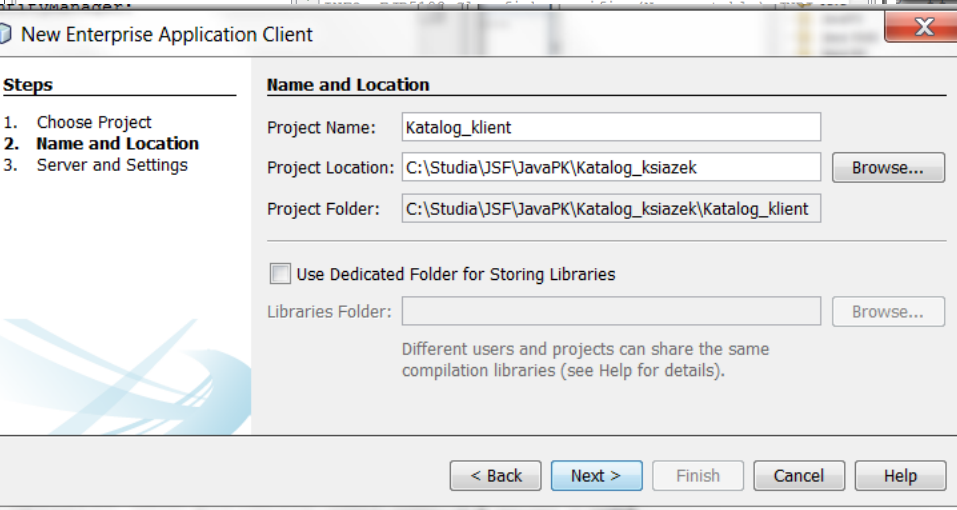
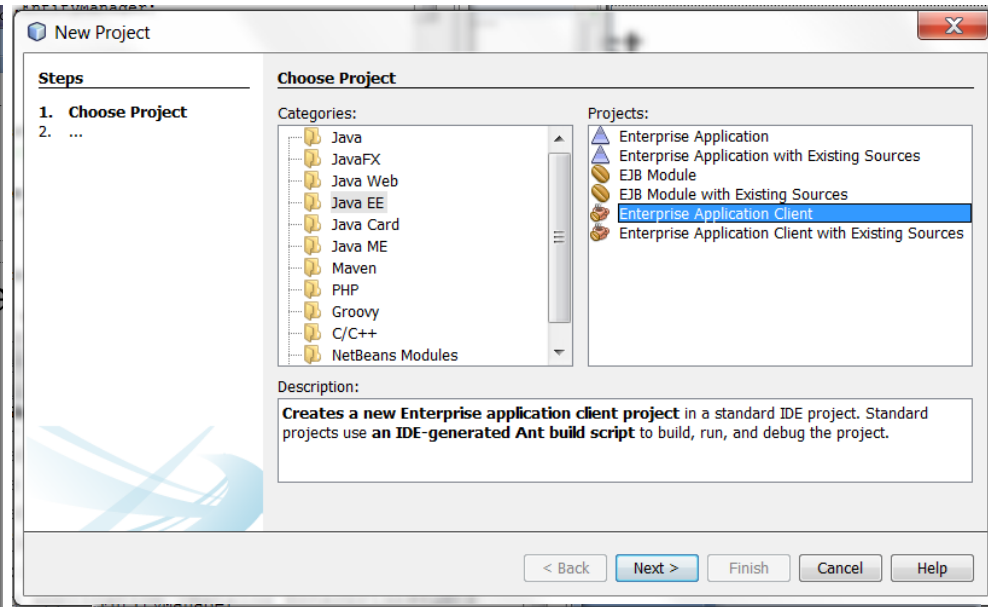
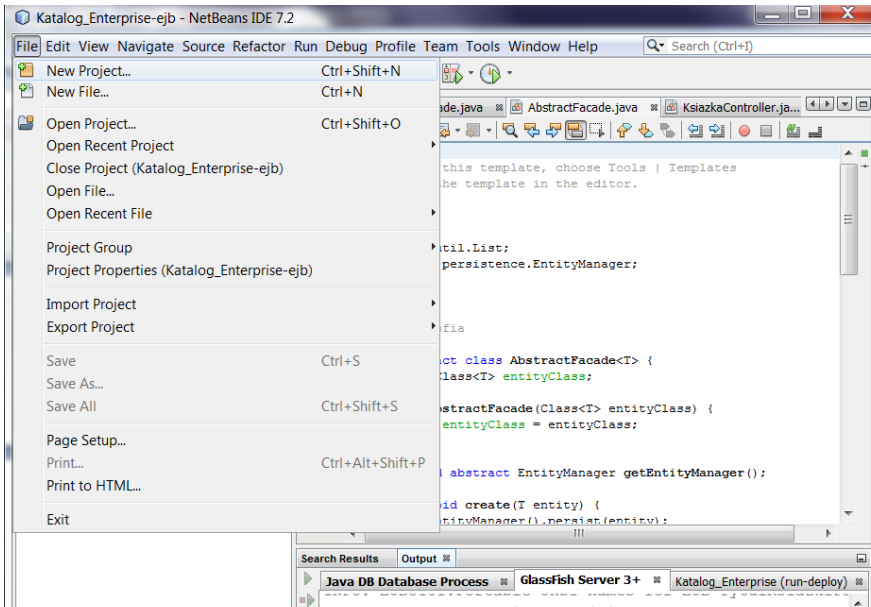
@ManagedBean(name = "ksiazkaController")
@SessionScoped
public class KsiazkaController implements Serializable {
    @EJB
    private KsiazkaFacadeRemote.ejbFacade;
    private Ksiazka current;
    private DataModel items = null;
    private PaginationHelper pagination;
    private int selectedItemIndex;

    public KsiazkaController() {
    }

    public Ksiazka getSelected() {
        if (current == null) {
            current = new Ksiazka();
            selectedItemIndex = -1;
        }
        return current;
    }

    private KsiazkaFacadeRemote getFacade() {
        return.ejbFacade;
    }
}
```

9.1. Utworzenie projektu klienta typu Enterprise : New Project/Java EE/Enterprise Application Client; Next; następnie należy nadać nazwę Katalog_klient (Project Name) w wybranym katalogu (Project Location); Next; następnie należy wybrać Project Katalog_Enterprise (Add to Enterprise Application), serwer GlassFish (Server) i platformę Java EE 6 (Java EE Version)



9.2. Należy usunąć klasę Main z projektu Katalog_klient, utworzoną domyślnie w domyślnym pakiecie katalog_klient

The image shows two overlapping screenshots of the NetBeans IDE 7.2 interface. The background screenshot displays the project structure of 'Katalog_klient' in the 'Projects' view on the left. The 'Main.java' file is highlighted under the 'katalog_klient' source package. The 'Main' class is visible in the 'Members View' at the bottom left. The foreground screenshot shows the 'Main.java' file open in the editor, displaying the following code:

```
public void paintComponent(Graphics g) {  
    super.paintComponent(g);  
    table_content();  
}  
  
public Object[][] titles(List<TytulKsiazki> lista) {  
    Object[][] dane1 = new Object [lista.size()][];  
    for (int i = 0; i < lista.size(); i++) {  
        String[] d2 = lista.get(i).tytul();  
        dane1[i] = d2;  
    }  
}
```

A 'Delete' dialog box is open in the foreground, titled 'Delete Main'. It contains the following options:

- Safely delete (with usage search)
- Search In Comments

At the bottom of the dialog, there are four buttons: 'Preview', 'Refactor', 'Cancel', and 'Help'.

9.3. Należy do klasy TytułKsiążki w projekcie Katalog dodać metodę tytuł() do prezentowania danych obiektu typu TytułKsiążki

```
public String[] tytuł()  
{  
    String[] help={tytułId.toString(), tytuł, autorNazwisko, autorImię,  
                  isbn, wydawnictwo};  
    return help;  
}
```


9.4. Należy do klasy Ksiazka w projekcie Katalog dodać metodę toString_() do prezentowania danych obiektu typu Ksiazka

```
public String toString_() {  
    return "entities.Ksiazka[ ksiazkaId=" + ksiazkaId + " ] " +idTytul.toString()+  
        " numer: "+numer;  
}
```

9.5. Należy do klasy KsiazkaFacade w projekcie Katalog dodać metodę findKsiazkaEntities do pobrania danych typu Ksiazka obiektu typu TytulKsiazki

@Override

```
public List<Ksiazka> findKsiazkaEntities(TytulKsiazki tytul) {  
    Query q = em.createQuery("SELECT k FROM Ksiazka k WHERE  
                                k.idTytul.tytulId = " + tytul.getTytulId());  
    return q.getResultList();  
}
```

9.6. Uzupełnienie definicji interfejsu KsiazkaFacadeRemote o metodę findKsiazkaEntities

The screenshot displays the NetBeans IDE 7.2 interface. The main editor window shows the `KsiazkaFacadeRemote.java` file with the following code:

```
4  */
5  package orm;
6
7  import entities.Ksiazka;
8  import entities.TytulKsiazki;
9  import java.util.List;
10 import javax.ejb.Remote;
11
12 /**
13  *
14  * @author Zofia
15  */
16 @Remote
17 public interface KsiazkaFacadeRemote {
18
19     void create(Ksiazka ksiazka);
20
21     void edit(Ksiazka ksiazka);
22
23     void remove(Ksiazka ksiazka);
24
25     Ksiazka find(Object id);
26
27     List<Ksiazka> findAll();
28
29     List<Ksiazka> findRange(int[] range);
30
31     int count();
32
33     List<Ksiazka> findKsiazkaEntities(TytulKsiazki tytul);
34 }
35
```

The `findKsiazkaEntities` method is highlighted in blue. The left sidebar shows the project structure for `Katalog_interfejs`, including the `orm` package and the `KsiazkaFacadeRemote.java` file. The bottom panel shows the `Members View` for `KsiazkaFacadeRemote`, listing the methods: `create`, `edit`, `remove`, `find`, `findAll`, `findRange`, `count`, and `findKsiazkaEntities`. The `findKsiazkaEntities` method is highlighted in red. The bottom status bar shows the page number 33 and the text INS.

9.7. Uzupełnie głównej klasy Client o dwie adnotacje do obu obiektów typu Session Bean for Entity class: TytulKsiazkiFacade oraz KsiazkaFacade (prawym klawiszem należy kliknąć na kod klasy Client, następnie wybrać Insert Code., i potem wybrać z listy formularza Call Enterprise Bean obiekty typu Session Bean for Entity class: TytulKsiazkiFacade oraz KsiazkaFacade)

The screenshot shows the NetBeans IDE interface. The left pane displays the project structure for 'Katalog_klient'. The main editor shows the 'Client.java' file with the following code:

```
1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5  package katalog_klient;
6
7  import java.awt.BorderLayout;
8  import java.awt.CardLayout;
9  import java.awt.Container;
10 import java.awt.event.ActionEvent;
11 import java.awt.event.ActionListener;
12 import java.awt.event.KeyEvent;
13 import javax.ejb.EJB;
14 import javax.swing.JFrame;
15 import javax.swing.JMenuBar;
16 import javax.swing.JMenu;
17 import javax.swing.JMenuItem;
18 import javax.swing.JPanel;
19 import javax.swing.KeyStroke;
20 import orm.KsiazkaFacadeRemote;
21 import orm.TytulKsiazkiFacadeRemote;
22
23 /**
24  *
25  * @author Zofia
26  */
27 public class Client implements ActionListener {
28     @EJB
29     private static KsiazkaFacadeRemote ksiazkaFacade;
30     @EJB
31     private static TytulKsiazkiFacadeRemote tytulKsiazkiFacade;
32
33
34     JPanel cards; // a panel that uses CardLayout
35     final static String NOTHING1 = "Empty1";
36     final static String BOOK = "Book and Title form";
37
38     public JMenuBar createMenuBar() {
39         JMenuBar menuBar;
40         JMenu menu, submenu;
```

The 'Katalog_klient' project folder is highlighted with a red box in the left pane. The 'Members View' at the bottom shows the following members:

- Client :: ActionListener
- ksiazkaFacade : KsiazkaFacadeRemote
- tytulKsiazkiFacade : TytulKsiazkiFacadeRemote

The status bar at the bottom right shows '156 | 32 | INS'.

9.8. Kod klas z pakietu katalog_klient z projektu Katalog_klient: Book_form

```
package katalog_klient;

import entities.Ksiazka;
import entities.TytulKsiazki;
import java.awt.Dimension;
import java.awt.Graphics;
import java.util.Iterator;
import java.util.List;
import javax.swing.BoxLayout;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.AbstractTableModel;

public class Book_form extends JPanel {

    private JTable table;
    int row = 0;
    Client client;
    MyTableModel model;
    JComboBox books;
```

```
public Book_form(Client client) {
    super();
    this.client = client;
    setLayout(new BorderLayout(this, BorderLayout.Y_AXIS));
    model = new MyTableModel();
    table_content();
    table = new JTable(model);
    table.setPreferredScrollableViewportSize(new Dimension(500, 100));
    table.setFillViewportHeight(true);
    table.getSelectionModel().addListSelectionListener(new RowListener());

    add(new JScrollPane(table));

    JLabel lbooks = new JLabel("Books");
    add(lbooks);
    books = new JComboBox();
    add(books);
}
```

@Override

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    table_content();
}
```

```

//wyswietlanie tytulow
void table_content() {
    List<TytulKsiazki> titles = Client.getTytulKsiazkiFacade().findAll();
    model.setData(titles_(titles));
}

public Object[][] titles_(List<TytulKsiazki> lista) {
    Object[][] dane1 = new Object[lista.size()][];
    for (int i = 0; i < lista.size(); i++) {
        String[] d2 = lista.get(i).tytul();
        dane1[i] = d2;
    }
    return dane1;
}

// wyswietlanie ksiazek podanego tytulow
void print_books() {
    TytulKsiazki tytul = Client.getTytulKsiazkiFacade().find(title_id());
    List<Ksiazka> ksiazki = Client.getKsiazkaFacade().findKsiazkaEntities(tytul);
    if (ksiazki == null) {
        return;
    }
    list_content(ksiazki, books);
}

int title_id() {
    String data = (String) model.getValueAt(row, 0);
    return Integer.parseInt(data);
}

private void list_content(List<Ksiazka> ksiazki, JComboBox list) {
    String s;
    list.removeAllItems();
    Iterator<Ksiazka> iterator = ksiazki.iterator();
    while (iterator.hasNext()) {
        s = iterator.next().toString_();
        list.addItem(s);
    }
}

```

```
private class RowListener implements ListSelectionListener {  
  
    @Override  
    public void valueChanged(ListSelectionEvent event) {  
        if (event.getValueIsAdjusting()) {  
            return;  
        }  
        row = table.getSelectionModel().getLeadSelectionIndex();  
        print_books();  
    }  
}
```

```
class MyTableModel extends AbstractTableModel {
```

```
    private String[] columnNames = {"ID", "Title",  
        "First Name",  
        "Name",  
        "ISBN",  
        "Publisher"};  
    private Object[][] data;
```

```
    public void setData(Object[][] val) {  
        data = val;  
    }
```

```
    @Override  
    public int getColumnCount() {  
        return columnNames.length;  
    }
```

```
    @Override  
    public int getRowCount() {  
        return data.length;  
    }
```

```
    @Override  
    public String getColumnName(int col) {  
        return columnNames[col];  
    }
```


@Override

```
public Object getValueAt(int row, int col) {  
    return data[row][col];  
  
}
```

@Override

```
public Class getColumnClass(int c) {  
    return getValueAt(0, c).getClass();  
}
```

@Override

```
public boolean isCellEditable(int row, int col) {  
    if (col < 0) {  
        return false;  
    } else {  
        return true;  
    }  
}
```

@Override

```
public void setValueAt(Object value, int row, int col) {  
    data[row][col] = value;  
    fireTableCellUpdated(row, col);  
}  
}  
}
```

9.9. Kod klas z pakietu katalog_klient z projektu Katalog_klient: Client

```
package katalog_klient;

import java.awt.BorderLayout;
import java.awt.CardLayout;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import javax.ejb.EJB;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.KeyStroke;
import orm.KsiazkaFacadeRemote;
import orm.TytulKsiazkiFacadeRemote;

public class Client implements ActionListener {
    @EJB
    private static KsiazkaFacadeRemote ksiazkaFacade;
    @EJB
    private static TytulKsiazkiFacadeRemote tytulKsiazkiFacade;

    JPanel cards; //a panel that uses CardLayout
    final static String NOTHING1 = "Empty1";
    final static String BOOK = "Book and Title form";
```

```
public JMenuBar createMenuBar() {
    JMenuBar menuBar;
    JMenu menu, submenu;
    JMenuItem menuItem;

    //Create the menu bar.
    menuBar = new JMenuBar();

    menu = new JMenu("A Menu");
    menu.setMnemonic(KeyEvent.VK_A);
    menuBar.add(menu);

    menuItem = new JMenuItem(BOOK);
    menuItem.setMnemonic(KeyEvent.VK_B);
    menuItem.addActionListener(this);
    menu.add(menuItem);

    menuItem = new JMenuItem(NOTHING1);
    menuItem.setMnemonic(KeyEvent.VK_E);
    menuItem.addActionListener(this);
    menu.add(menuItem);

    menu.addSeparator();

    submenu = new JMenu("A submenu");
    submenu.setMnemonic(KeyEvent.VK_S);

    menuItem = new JMenuItem(NOTHING1);
    menuItem.setAccelerator(KeyStroke.getKeyStroke(
        KeyEvent.VK_2, ActionEvent.ALT_MASK));
    menuItem.addActionListener(this);
    submenu.add(menuItem);

    menuItem = new JMenuItem(NOTHING1);
    menuItem.addActionListener(this);
    submenu.add(menuItem);
    menu.add(submenu);

    //Build second menu in the menu bar.
    menu = new JMenu("Another Menu");
    menu.setMnemonic(KeyEvent.VK_N);
    menuBar.add(menu);

    return menuBar;
}
```

```
public static KsiazkaFacadeRemote getKsiazkaFacade() {
    return ksiazkaFacade;
}

public static void setKsiazkaFacade(KsiazkaFacadeRemote ksiazkaFacade) {
    Client.ksiazkaFacade = ksiazkaFacade;
}

public static TytulKsiazkiFacadeRemote getTytulKsiazkiFacade() {
    return tytulKsiazkiFacade;
}

public static void setTytulKsiazkiFacade(TytulKsiazkiFacadeRemote tytulKsiazkiFacade) {
    Client.tytulKsiazkiFacade = tytulKsiazkiFacade;
}

public Container createContentPane() {
    //Create the content-pane-to-be.

    Card0 card0 = new Card0();

    Book_form card2 = new Book_form(this);

    //Create the panel that contains the "cards".
    cards = new JPanel(new CardLayout());
    cards.add(card0, NOTHING1);

    cards.add(card2, BOOK);

    JPanel p1 = new JPanel();

    p1.add(cards, BorderLayout.CENTER);
    return p1;
}
```

```

public void actionPerformed(ActionEvent e) {

    JMenuItem source = (JMenuItem) (e.getSource());
    CardLayout cl = (CardLayout) (cards.getLayout());
    if (source.getText().equals(BOOK)) {
        cl.show(cards, BOOK);
    } else if (source.getText().equals(NOTHING1)) {
        cl.show(cards, NOTHING1);
    }
}

/**
 * Create the GUI and show it. For thread safety, this method should be
 * invoked from the event-dispatching thread.
 */
private static void createAndShowGUI() {
    //Create and set up the window.
    JFrame frame = new JFrame("MenuDemo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(800, 460);
    //Create and set up the content pane.
    Client demo = new Client();
    frame.setJMenuBar(demo.createMenuBar());
    frame.setContentPane(demo.createContentPane());

    //Display the window.
    frame.setVisible(true);
}

public static void main(String[] args) {
    //Schedule a job for the event-dispatching thread:
    //creating and showing this application's GUI.
    java.awt.EventQueue.invokeLater(new Runnable() {
        //javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            createAndShowGUI();
        }
    });
}
}

```

9.9. Kod klas z pakietu katalog_klient z projektu Katalog_klient: Card0

```
package katalog_klient;

import javax.swing.JPanel;

public class Card0 extends JPanel{
    public Card0() {
    }

}
```

10. Uruchomienie aplikacji

1. Wykonanie operacji Clean/Build w zakładce Projects dla projektu Katalog
2. Wykonanie operacji Clean/Build w zakładce Projects dla projektu Katalog_interfejs.
3. Wykonanie operacji Clean/Build w zakładce Projects dla projektu Katalog_Enterprise-ejb.
4. Wykonanie operacji Clean/Build w zakładce Projects dla projektu Katalog_Enterprise-war.
5. Wykonanie operacji Clean/Build w zakładce Projects dla projektu Katalog_klient.
6. Wykonanie operacji run dla projektu Katalog_Enterprise – w oknie domyślnej przeglądarki powinna uruchomić się aplikacja klienta internetowego (slajd 56).
7. Wykonanie operacji run dla klienta typu Enterprise Katalog_klient (slajd 57).

10.1. Widok formularzy klienta internetowego

Facelet Title

localhost:8080/Katalog_Enterprise-war/

Hello from Facelets

[Show All Ksiazka Items](#)

[Show All TytulKsiazki Items](#)

List

1..6/6

TytuId	Tytuł	AutorNazwisko	AutorImie	Isbn	Wydawnictwo	
1	Krzyzacy	Sienkiewicz	Henryk	1234567	PWN	View Edit Destroy
3	1	1	1	1	1	View Edit Destroy
4	2	2	2	2	2	View Edit Destroy
6	5	5	5	5	5	View Edit Destroy
7	6	6	6	6	6	View Edit Destroy
8	8	8	8	8	8	View Edit Destroy

[Create New TytulKsiazki](#)

[Index](#)

Facelet Title

localhost:8080/Katalog_Enterprise-war/faces/index.xhtml

Hello from Facelets

[Show All Ksiazka Items](#)

[Show All TytulKsiazki Items](#)

List

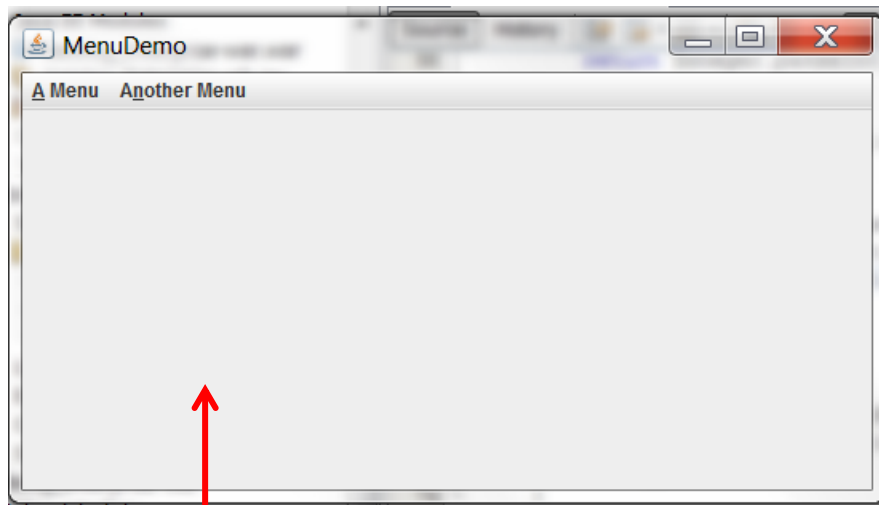
1..4/4

KsiazkaId	Numer	IdTytuł	
1	1	1	View Edit Destroy
3	1	1	View Edit Destroy
4	10	3	View Edit Destroy
5	1	4	View Edit Destroy

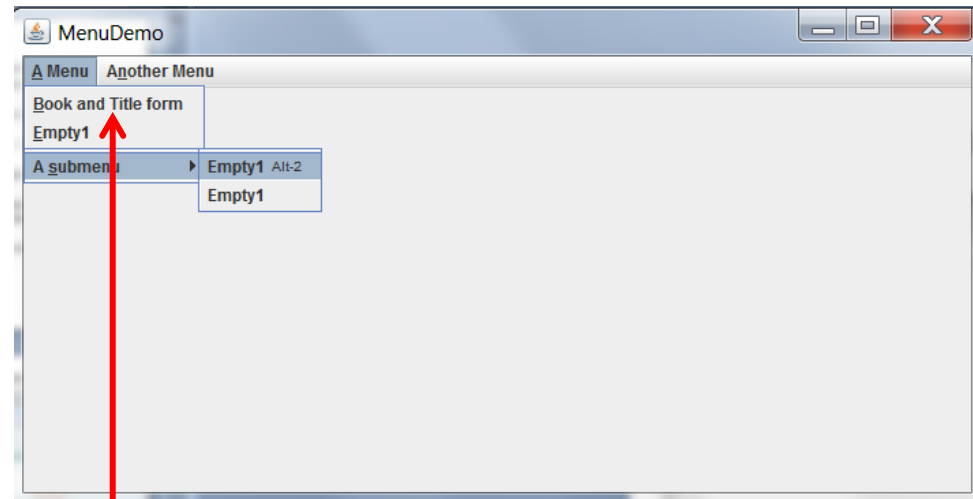
[Create New Ksiazka](#)

[Index](#)

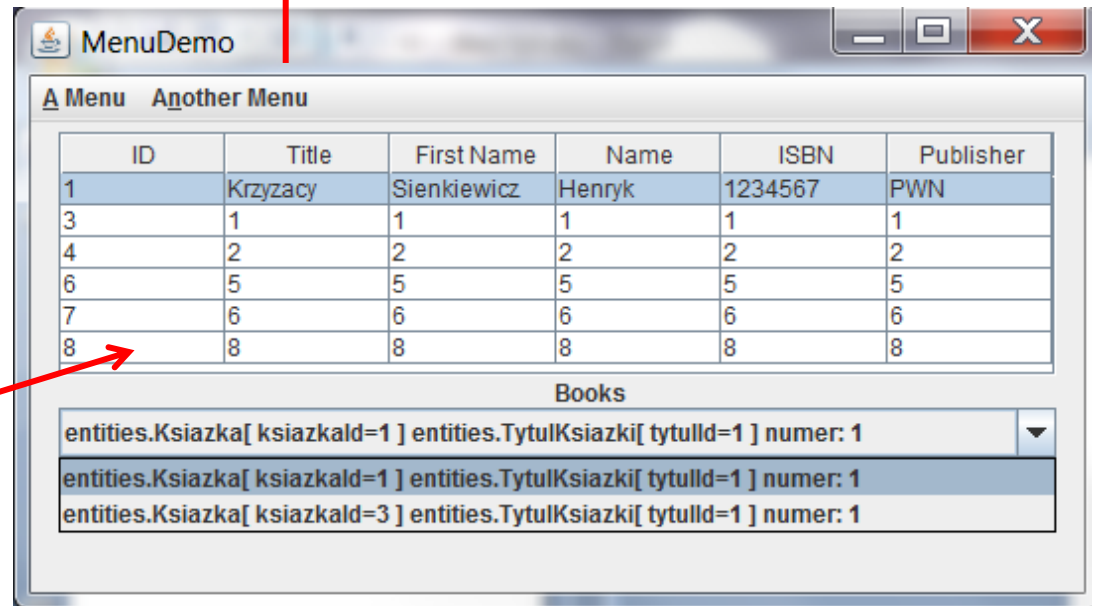
10.1. Widok formularzy klienta Enterprise



Widok formularza Card0



Widok formularza Book_form



11. Dalsze prace - propozycja

1. Wprowadzenie nowego szablonu strony internetowej
2. Wprowadzenie obiektów transferowych do przesyłania danych między warstwą klienta Enterprise i warstwą integracji oraz warstwą prezentacji klienta internetowego i warstwą integracji.