

Projektowanie oprogramowania

Warstwa integracji z bazą danych
oparta na technologii ORM

Platforma Java EE

Autor: Zofia Kruczkiewicz

Wykonanie czterowarstwowej aplikacji EE z dostępem do bazy danych, opartym na wzorcu Domain Store

(<http://corej2eepatterns.com/DomainStore.htm>)

1. Utworzenie pustej bazy danych – w okienku Services należy prawym klawiszem myszy wybrać **Databases\Java DB\Create Database**

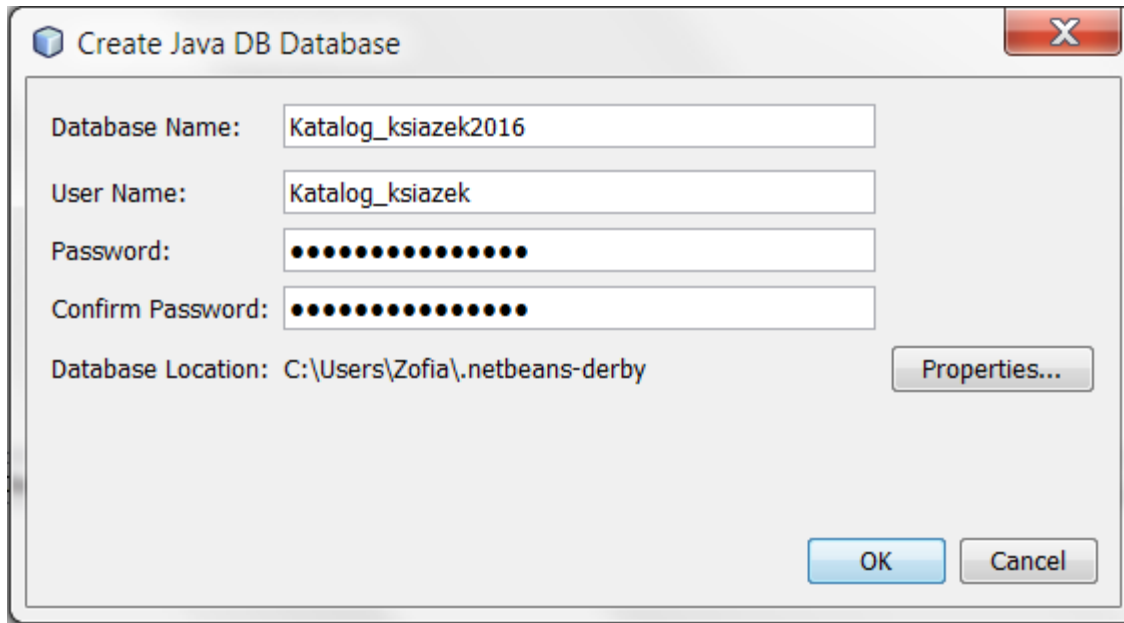
The screenshot shows the NetBeans IDE 7.2 interface. The 'Services' window is open, displaying a tree view of services. A context menu is open over the 'Java DB' service, with the 'Create Database...' option selected. The main editor area shows a design view for a database table with the following fields:

- Tytuł książki
- Nazwisko autora książki
- Imię autora książki
- ISBN tytułu książki
- Wydawnictwo książki
- Numer książki

Below the fields are several buttons for saving and displaying data:

- Zapisz tytuł
- Zapisz książkę
- Wyświetl tytuły
- Wyświetl książki
- Tytuły do bazy
- Książki do bazy
- Tytuły z bazy
- Książki z bazy

1.1. Wykonanie pustej bazy danych Katalog_ksiazek2016 login: Katalog_ksiazek haslo: Katalog_ksiazek



The screenshot shows a dialog box titled "Create Java DB Database" with a close button (X) in the top right corner. The dialog contains the following fields and buttons:

- Database Name:
- User Name:
- Password:
- Confirm Password:
- Database Location:
- Properties... button
- OK button
- Cancel button



Projects Profiler Services

Databases

- MySQL Server at localhost:3306 [root]
- Java DB
- Drivers
- jdbc:derby://localhost:1527/aa [aa on AA]
- jdbc:derby://localhost:1527/Dostawca [Dostawca on DOSTAWCA]
- jdbc:derby://localhost:1527/katalog11 [katalog1 on KATALOG1]
- jdbc:derby://localhost:1527/Katalog_ksiazek [Katalog_ksiazek on KATALOG_KSIAZEK]
- jdbc:derby://localhost:1527/Katalog_ksiazek2016 [Katalog_ksiazek on KATALOG_KSIAZEK]**
- jdbc:derby://localhost:1527/KatalogAwarii [KatalogAwarii on KATALOGAWARII]
- jdbc:derby://localhost:1527/MP1 [MP1 on MP1]
- jdbc:derby://localhost:1527/MP2 [MP2 on MP2]
- jdbc:derby://localhost:1527/MP3 [MP3 on MP3]
- jdbc:derby://localhost:1527/MP4 [MP4 on MP4]
- jdbc:derby://localhost:1527/MP5 [MP5 on MP5]
- jdbc:derby://localhost:1527/MP6 [MP6 on MP6]
- jdbc:derby://localhost:1527/MP7 [MP7 on MP7]
- jdbc:derby://localhost:1527/MP8 [MP8 on MP8]
- jdbc:derby://localhost:1527/MP9 [MP9 on MP9]
- jdbc:derby://localhost:1527/MP10 [MP10 on MP10]
- jdbc:derby://localhost:1527/MP11 [MP11 on MP11]
- jdbc:derby://localhost:1527/MP12 [MP12 on MP12]
- jdbc:derby://localhost:1527/MP13 [MP13 on MP13]
- jdbc:derby://localhost:1527/MP14 [MP14 on MP14]
- jdbc:derby://localhost:1527/MP15 [MP15 on MP15]
- jdbc:derby://localhost:1527/MP16 [MP16 on MP16]
- jdbc:derby://localhost:1527/MP17 [MP17 on MP17]
- jdbc:derby://localhost:1527/MP18 [MP18 on MP18]
- jdbc:derby://localhost:1527/MP19 [MP19 on MP19]
- jdbc:derby://localhost:1527/MP20 [MP20 on MP20]
- jdbc:mysql://localhost:3306/ksiazki [ksiazki on KSIAZKI]

NetBeans IDE 8.0.2

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

1.2. Połączenie z pustą bazą danych o nazwie Katalog_ksiazek2016

Projects Services Profiler

Databases

- MySQL Server at localhost:3306 [root]
- Java DB
- Drivers
- jdbc:derby://localhost:1527/aa [aa on AA]
- jdbc:derby://localhost:1527/Dostawca [Dostawca on DOSTAWCA]
- jdbc:derby://localhost:1527/katalog11 [katalog1 on KATALOG1]
- jdbc:derby://localhost:1527/Katalog_ksiazek [Katalog_ksiazek on KATALOG_KSIAZEK]
- jdbc:derby://localhost:1527/Katalog_ksiazek2016 [Katalog_ksiazek on KATALOG_KSIAZEK]**
- jdbc:derby://localhost:1527/KatalogAwarii [KatalogAwarii on KATALOGAWARII]
- jdbc:derby://localhost:1527/MP1 [MP1 on MP1]
- jdbc:derby://localhost:1527/MP2 [MP2 on MP2]
- jdbc:derby://localhost:1527/MP3 [MP3 on MP3]
- jdbc:derby://localhost:1527/MP4 [MP4 on MP4]
- jdbc:derby://localhost:1527/MP5 [MP5 on MP5]
- jdbc:derby://localhost:1527/MP6 [MP6 on MP6]
- jdbc:derby://localhost:1527/MP7 [MP7 on MP7]
- jdbc:derby://localhost:1527/MP8 [MP8 on MP8]
- jdbc:derby://localhost:1527/MP9 [MP9 on MP9]
- jdbc:derby://localhost:1527/MP10 [MP10 on MP10]
- jdbc:derby://localhost:1527/MP11 [MP11 on MP11]
- jdbc:derby://localhost:1527/MP12 [MP12 on MP12]
- jdbc:derby://localhost:1527/MP13 [MP13 on MP13]
- jdbc:derby://localhost:1527/MP14 [MP14 on MP14]
- jdbc:derby://localhost:1527/MP15 [MP15 on MP15]
- jdbc:derby://localhost:1527/MP16 [MP16 on MP16]
- jdbc:derby://localhost:1527/MP17 [MP17 on MP17]
- jdbc:derby://localhost:1527/MP18 [MP18 on MP18]
- jdbc:derby://localhost:1527/MP19 [MP19 on MP19]
- jdbc:derby://localhost:1527/MP20 [MP20 on MP20]
- jdbc:mysql://localhost:3306/ksiazki [ksiazki on KSIAZKI]

Web Services

Servers

Maven Repository

Cloud

Hudson Builders

Task Repository

JS Test Driver

NetBeans IDE 8.0.2

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

Projects Services Profiler

Databases

- MySQL Server at localhost:3306 [root]
- Java DB
- Drivers
- jdbc:derby://localhost:1527/aa [aa on AA]
- jdbc:derby://localhost:1527/Dostawca [Dostawca on DOSTAWCA]
- jdbc:derby://localhost:1527/katalog11 [katalog1 on KATALOG1]
- jdbc:derby://localhost:1527/Katalog_ksiazek [Katalog_ksiazek on KATALOG_KSIAZEK]
- jdbc:derby://localhost:1527/Katalog_ksiazek2016 [Katalog_ksiazek on KATALOG_KSIAZEK]**
- jdbc:derby://localhost:1527/KatalogAwarii [KatalogAwarii on KATALOGAWARII]
- jdbc:derby://localhost:1527/MP1 [MP1 on MP1]
- jdbc:derby://localhost:1527/MP2 [MP2 on MP2]
- jdbc:derby://localhost:1527/MP3 [MP3 on MP3]
- jdbc:derby://localhost:1527/MP4 [MP4 on MP4]
- jdbc:derby://localhost:1527/MP5 [MP5 on MP5]
- jdbc:derby://localhost:1527/MP6 [MP6 on MP6]
- jdbc:derby://localhost:1527/MP7 [MP7 on MP7]
- jdbc:derby://localhost:1527/MP8 [MP8 on MP8]
- jdbc:derby://localhost:1527/MP9 [MP9 on MP9]
- jdbc:derby://localhost:1527/MP10 [MP10 on MP10]
- jdbc:derby://localhost:1527/MP11 [MP11 on MP11]
- jdbc:derby://localhost:1527/MP12 [MP12 on MP12]
- jdbc:derby://localhost:1527/MP13 [MP13 on MP13]
- jdbc:derby://localhost:1527/MP14 [MP14 on MP14]
- jdbc:derby://localhost:1527/MP15 [MP15 on MP15]
- jdbc:derby://localhost:1527/MP16 [MP16 on MP16]
- jdbc:derby://localhost:1527/MP17 [MP17 on MP17]
- jdbc:derby://localhost:1527/MP18 [MP18 on MP18]
- jdbc:derby://localhost:1527/MP19 [MP19 on MP19]
- jdbc:derby://localhost:1527/MP20 [MP20 on MP20]
- jdbc:mysql://localhost:3306/ksiazki [ksiazki on KSIAZKI]

Context menu for selected database:

- Connect...
- Disconnect
- Execute Command...
- Refresh
- Delete
- Rename...
- Properties

NetBeans IDE 8.0.2

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

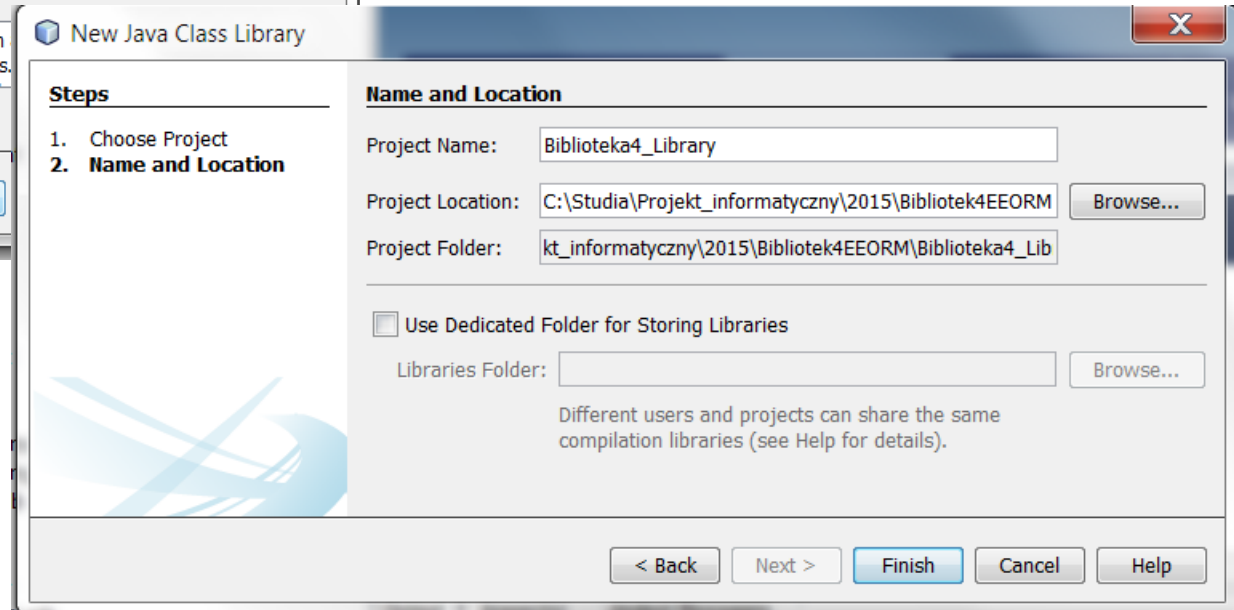
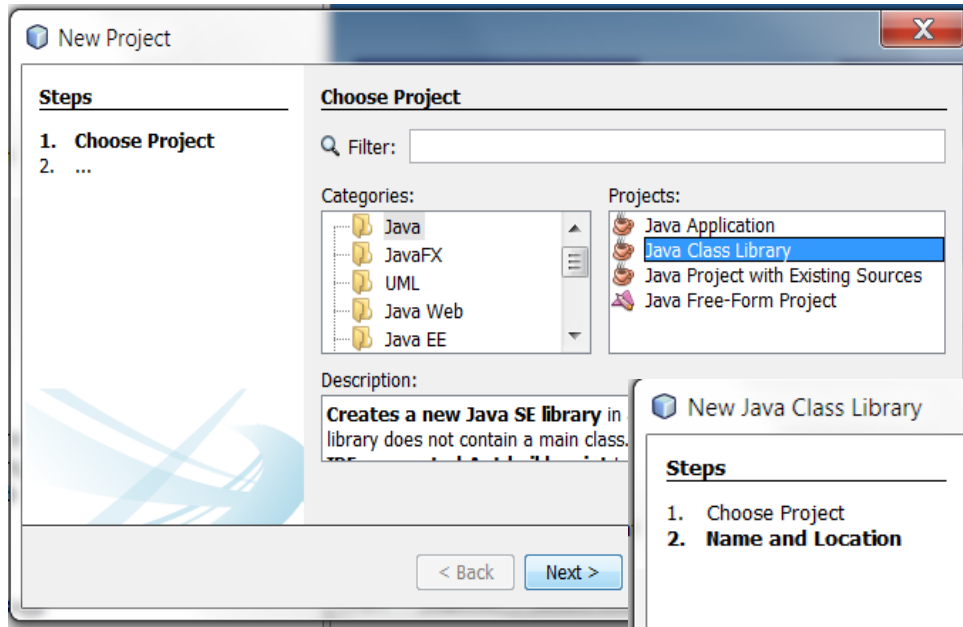
Projects Profiler Services

Databases

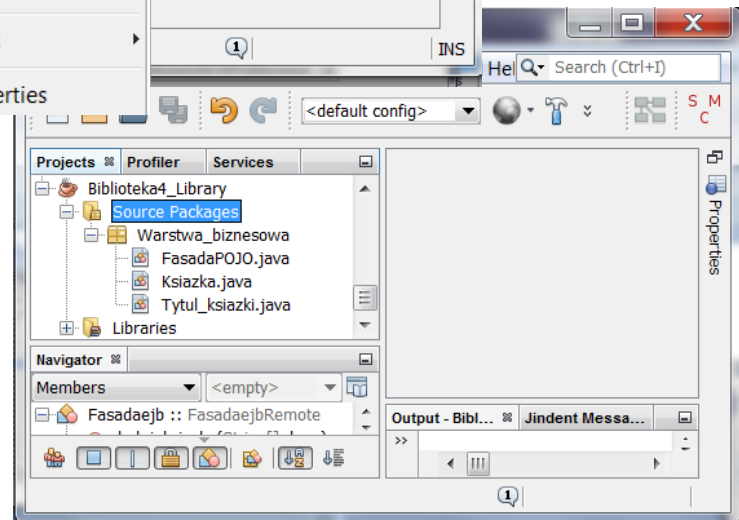
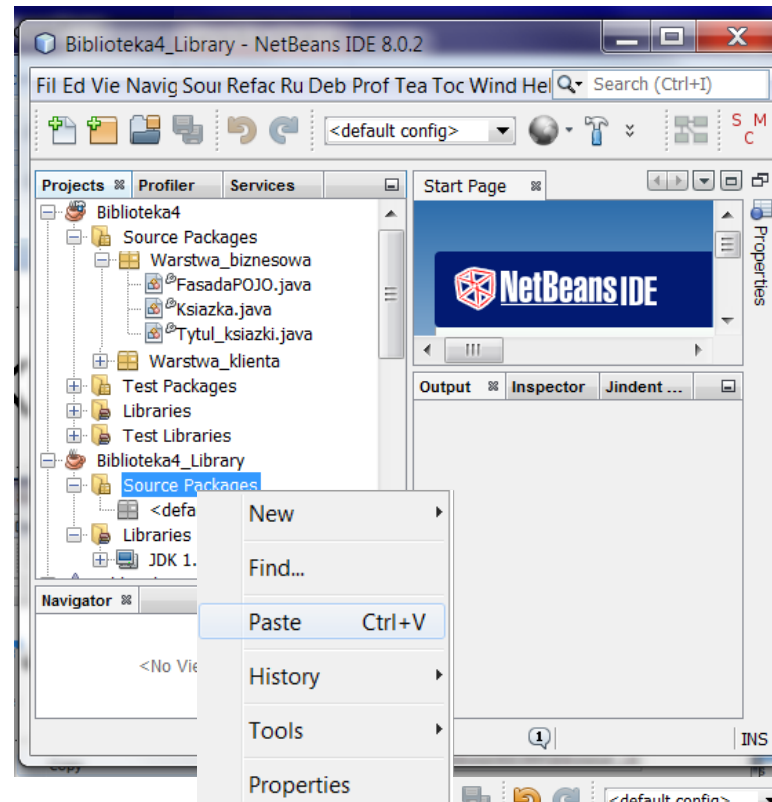
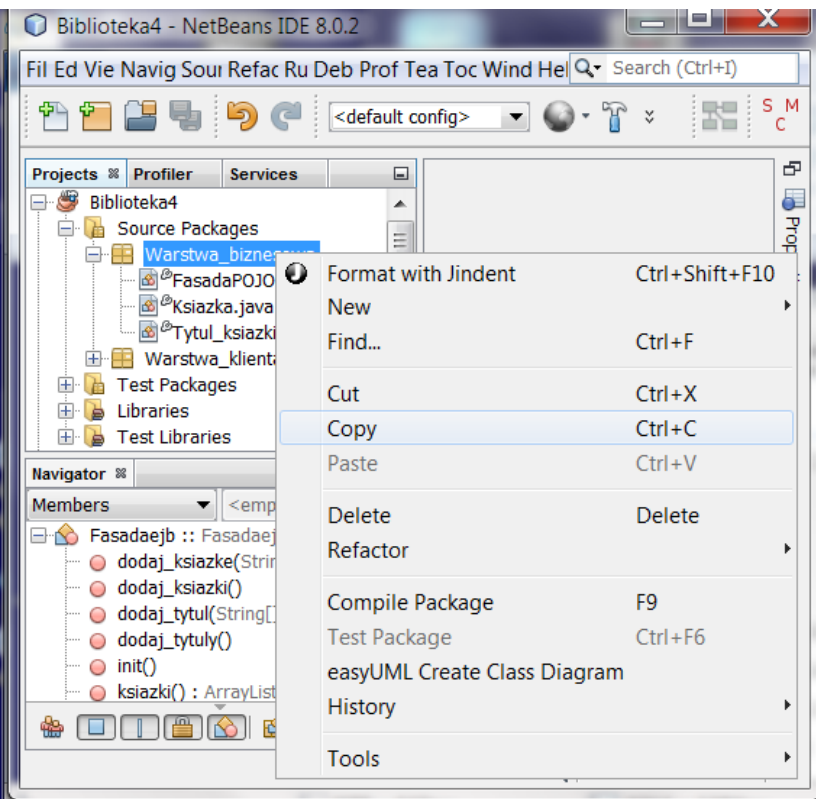
- MySQL Server at localhost:3306 [root]
- Java DB
- Drivers
- jdbc:derby://localhost:1527/aa [aa on AA]
- jdbc:derby://localhost:1527/Dostawca [Dostawca on DOSTAWCA]
- jdbc:derby://localhost:1527/katalog11 [katalog1 on KATALOG1]
- jdbc:derby://localhost:1527/Katalog_ksiazek [Katalog_ksiazek on KATALOG_KSIAZEK]
- jdbc:derby://localhost:1527/Katalog_ksiazek2016 [Katalog_ksiazek on KATALOG_KSIAZEK]**
- KATALOG_KSIAZEK
 - Tables
 - Views
 - Procedures
- Other schemas
- jdbc:derby://localhost:1527/KatalogAwarii [KatalogAwarii on KATALOGAWARII]
- jdbc:derby://localhost:1527/MP1 [MP1 on MP1]
- jdbc:derby://localhost:1527/MP2 [MP2 on MP2]
- jdbc:derby://localhost:1527/MP3 [MP3 on MP3]
- jdbc:derby://localhost:1527/MP4 [MP4 on MP4]
- jdbc:derby://localhost:1527/MP5 [MP5 on MP5]
- jdbc:derby://localhost:1527/MP6 [MP6 on MP6]
- jdbc:derby://localhost:1527/MP7 [MP7 on MP7]
- jdbc:derby://localhost:1527/MP8 [MP8 on MP8]
- jdbc:derby://localhost:1527/MP9 [MP9 on MP9]
- jdbc:derby://localhost:1527/MP10 [MP10 on MP10]
- jdbc:derby://localhost:1527/MP11 [MP11 on MP11]
- jdbc:derby://localhost:1527/MP12 [MP12 on MP12]
- jdbc:derby://localhost:1527/MP13 [MP13 on MP13]
- jdbc:derby://localhost:1527/MP14 [MP14 on MP14]
- jdbc:derby://localhost:1527/MP15 [MP15 on MP15]
- jdbc:derby://localhost:1527/MP16 [MP16 on MP16]
- jdbc:derby://localhost:1527/MP17 [MP17 on MP17]
- jdbc:derby://localhost:1527/MP18 [MP18 on MP18]
- jdbc:derby://localhost:1527/MP19 [MP19 on MP19]
- jdbc:derby://localhost:1527/MP20 [MP20 on MP20]
- jdbc:mysql://localhost:3306/ksiazki [ksiazki on KSIAZKI]

2. Wykonanie adnotacji w obiektowym modelu danych

- 2.1. Należy wykonać projekt typu **Java Class Library** np. o nazwie **Biblioteka4_Library** (**File/New Project/Java/Java Class Library** i nacisnąć klawisz **Next**). Następnie w polu **Project Name** wpisać nazwę projektu po wyborze katalogu w polu **Project Location** i nacisnąć klawisz **Finish**.



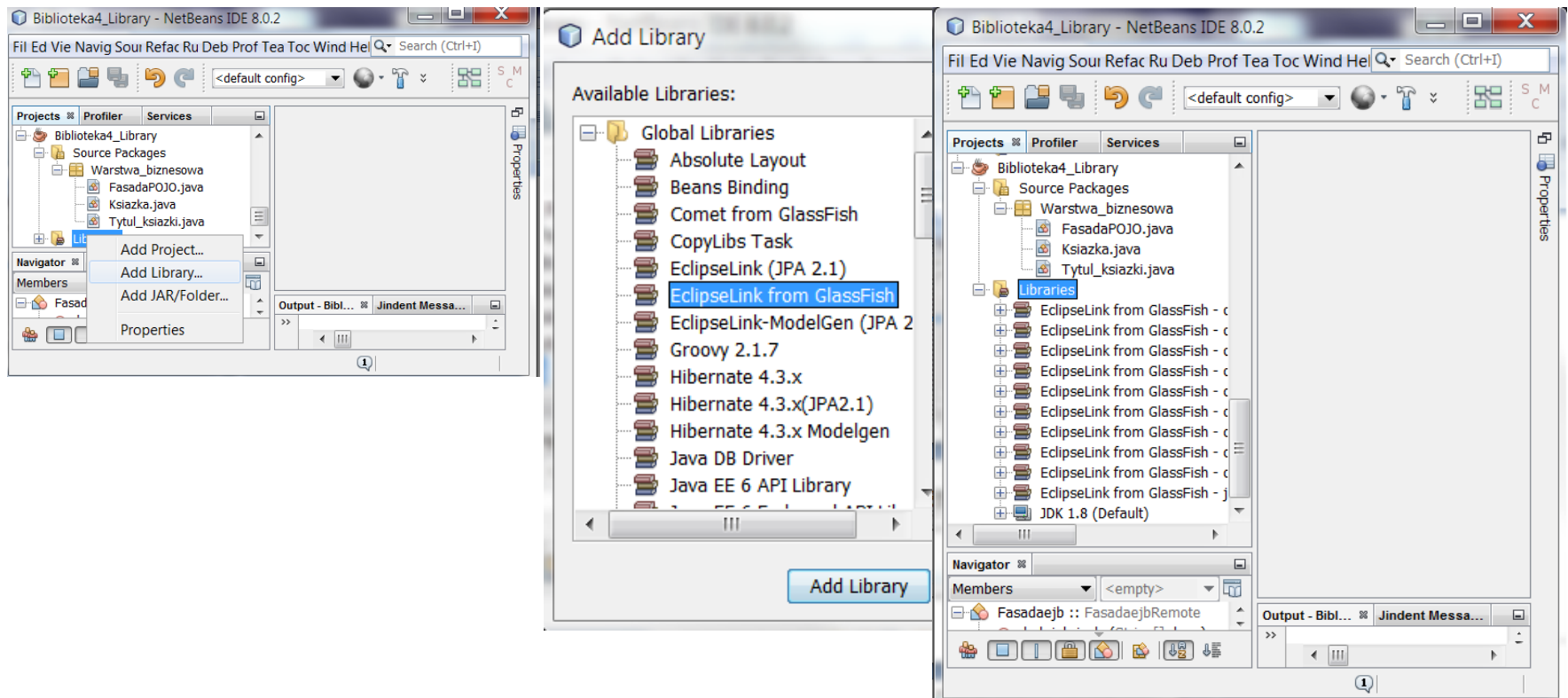
2.2. Wkleić pakiet Warstwa_biznesowa z projektu Biblioteka4 pobranego ze strony:
<http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/OBPROG/Biblioteka4.rar>



2.3. Wykonanie adnotacji w obiektowym modelu danych

2.3.1. Należy wykonać projekt typu **Java Class Library** np. o nazwie **Biblioteka4_Library**

2.3.2. Należy kliknąć prawym klawiszem myszy na folder **Libraries**, wybrać pozycję **Add Library...**, a następnie wybrać bibliotekę **EclipseLink from GlassFish**. Teraz projekt zawiera biblioteki umożliwiające przekształcenie klas z obiektowego modelu danych na klasy typu Entity za pomocą tzw. adnotacji wg <https://docs.oracle.com/javasee/7/tutorial/persistence-intro.htm#BNBPZ>



2.3.3. Na kolejnych pokazano, jak przekształcić klasy Tytul_książki oraz Książka na klasy typu Entity


```

package Warstwa_biznesowa;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.xml.bind.annotation.XmlTransient;

```

```

/**...*/

```

```

@Entity

```

```

public class Tytul_ksiazki implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Basic(optional = false)
    @Column(name = "ID_TYTUL")
    private Long idTytul;
    @Column(name = "TYTUL")
    private String tytul;
    @Column(name = "AUTOR_NAZWISKO")
    private String nazwisko;
    @Column(name = "AUTOR_IMIE")
    private String imie;
    @Column(name = "ISBN")
    private String ISBN;
    @Column(name = "WYDAWNICTWO")
    private String wydawnictwo;
    @OneToMany(mappedBy = "mTytul_ksiazki", cascade=CascadeType.ALL)
    private Collection<Ksiazka> mKsiazka;

```

3.1. Przekształcenie klasy **Tytul_ksiazki** na typ **Entity** - w celu utrwalania jej w bazie danych technologii ORM (JPA)

3.1. Przekształcenie klasy Tytul_książki na typ Entity - w celu utrwalania jej w bazie danych technologią ORM (JPA) - cd

```
public Tytul_książki ()
{
}
public Tytul_książki(int i)
{ mKsiążka=new ArrayList();
  idTytul=null;
}
public Tytul_książki(Long idTytul)
public Long getIdTytul()
public void setIdTytul(Long idTytul)
public String getTytul()
public void setTytul(String tytul)
public String getNazwisko()
public void setNazwisko(String autorNazwisko)
public String getImie()
public void setImie(String autorImie)
public String getISBN()
public void setISBN(String isbn)
public String getWydawnictwo()
public void setWydawnictwo(String wydawnictwo)
public Collection<Książka> getMKsiążka()
{ return mKsiążka; }
public void setMKsiążka(Collection<Książka> książkaCollection)
{ this.mKsiążka = książkaCollection; }
```

idTytul ma wartość domyślną null- przypisanie jedynie podkreśla ten fakt. Przypisanie konkretnej wartości następuje automatycznie podczas utrwalania danych tego obiektu w procesie ORM

```
{ this.idTytul = idTytul; }
{ return idTytul; }
{ this.idTytul = idTytul; }
{ return tytul; }
{ this.tytul = tytul; }
{ return nazwisko; }
{ this.nazwisko = autorNazwisko; }
{ return imie; }
{ this.imie = autorImie; }
{ return ISBN; }
{ this.ISBN = isbn; }
{ return wydawnictwo; }
{ this.wydawnictwo = wydawnictwo; }
```

3.1. Przekształcenie klasy Tytul_książki na typ Entity - w celu utrwalania jej w bazie danych technologią ORM (JPA) - cd

Metoda hashCode() powinna być dodana do dodaniu adnotacji

```
@Override
public int hashCode() {
    int hash = 0;
    hash += (idTytul != null ? idTytul.hashCode() : 0);
    return hash;}

```

```
@Override
public String toString() {
    String pom = "Tytul: " + getTytul();
    pom += " Autor: " + getNazwisko() + " " + getImie();
    pom += " ISBN: " + getISBN();
    pom += " Wydawnictwo: " + getWydawnictwo();
    return pom;    }

```

```
@Override
public boolean equals(Object ob) { //your code here
    String isbn2 = ((Tytul_książki) ob).getISBN();
    return ISBN.equals(isbn2); }

```

3.1. Przekształcenie klasy Tytul_książki na typ Entity cd – ta część definicji służy do realizacji usług warstwy biznesowej aplikacji

```
public void dodaj_książke(String dane[]) // your code here
{
    Książka nowa= new Książka(1);
    if (nowa != null) {
        nowa.setNumer(Integer.parseInt(dane[1]));
        addKsiążka(nowa);
    }
}

public void addKsiążka(Książka nowa) {
    if (!this.mKsiążka.contains(nowa)) {
        this.mKsiążka.add(nowa);
        nowa.setTytuł_książki(this);
    }
}

public ArrayList<String> książki() {
    ArrayList<String> książki = new ArrayList();
    Iterator<Książka> it = mKsiążka.iterator();
    while (it.hasNext()) {
        książki.add(it.next().toString());
    }
    return książki;
}
}
```

3.2. Przekształcenie klasy Ksiazka na typ Entity - w celu utrwalania jej w bazie danych technologią ORM (JPA)

```
package Warstwa_biznesowa;
import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
/**...*/
@Entity
public class Ksiazka implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Basic(optional = false)
    @Column(name = "ID_KSIAZKA")
    private Long idKsiazka;
    @Column(name = "NUMER")
    private int numer;
    @JoinColumn(name = "ID_TYTUL_", referencedColumnName = "ID_TYTUL")
    @ManyToOne
    private Tytul_ksiazki mTytul_ksiazki;

    public Ksiazka() { }
    public Ksiazka(int i) { idKsiazka = null; }
    public Long getIdKsiazka() { return idKsiazka; }
    public void setIdKsiazka(Long idKsiazka) { this.idKsiazka = idKsiazka; }
    public int getNumer() { return numer; }
    public void setNumer(int numer) { this.numer = numer; }
    public Tytul_ksiazki getTytul_ksiazki() { return mTytul_ksiazki; }
    public void setTytul_ksiazki(Tytul_ksiazki idTytul) { this.mTytul_ksiazki = idTytul; }
}
```

3.2. Przekształcenie klasy Ksiazka na typ Entity - w celu utrwalania jej w bazie danych technologią ORM (JPA) cd

```
@Override
public int hashCode() {
    int hash = 0;
    hash += (idKsiazka != null ? idKsiazka.hashCode() : 0);
    return hash;
}
```

Metoda hashCode() powinna być dodana do dodaniu adnotacji

```
public boolean equals(Object ob) // your code here
{
    return numer == ((Ksiazka) ob).getNumer();
}
```

```
public String toString() // your code here
{
    String pom = mTytul_ksiazki.toString();
    pom += " Numer: " + getNumer();
    return pom;
}
```

```
}
```

3.3. Kod klasy FasadaPOJO, reprezentującą wzorzec strukturalny Fasada

```
package Warstwa_biznesowa;
...4 lines
import java.util.ArrayList;
import java.util.Iterator;
/**...4 lines */
public class FasadaPOJO {
    private ArrayList<Tytul_książki> tytul_y_książek;

    public FasadaPOJO() {
        tytul_y_książek = new ArrayList();
    }
    public ArrayList<Tytul_książki> getTytul_y_książek() {
        return tytul_y_książek;
    }
    public void setTytul_y_książek(ArrayList<Tytul_książki> val) {
        this.tytul_y_książek = val;
    }
    public void dodaj_tytul(String dane_tytul[]) {
        Tytul_książki tytul_książki = new Tytul_książki(1);
        tytul_książki.setTytul(dane_tytul[0]);
        tytul_książki.setNazwisko(dane_tytul[1]);
        tytul_książki.setImie(dane_tytul[2]);
        tytul_książki.setISBN(dane_tytul[3]);
        tytul_książki.setWydawnictwo(dane_tytul[4]);
        addtytul_książki(tytul_książki);
    }
    public void addtytul_książki(Tytul_książki val) {
        boolean czy_jest = tytul_y_książek.contains(val);
        if (!czy_jest) {
            tytul_y_książek.add(val);
        }
    }
    public void dodaj_książke(String dane[]) // your code here
    {
        Tytul_książki pom = new Tytul_książki();
        pom.setISBN(dane[0]);
        int idx = tytul_y_książek.indexOf(pom);
        if (idx != -1) {
            Tytul_książki pom1 = tytul_y_książek.get(idx);
            pom1.dodaj_książke(dane);
        }
    }
}
```

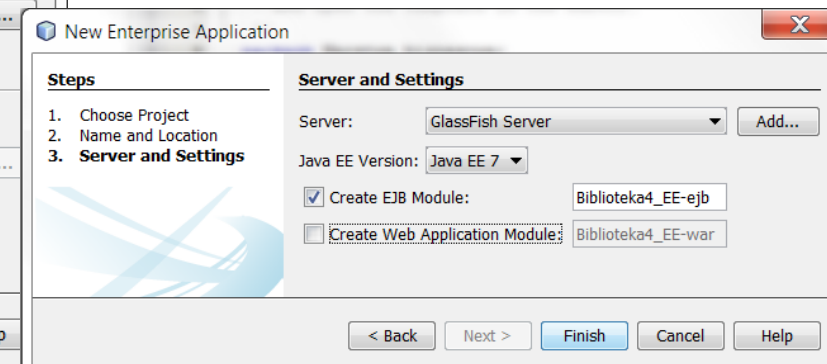
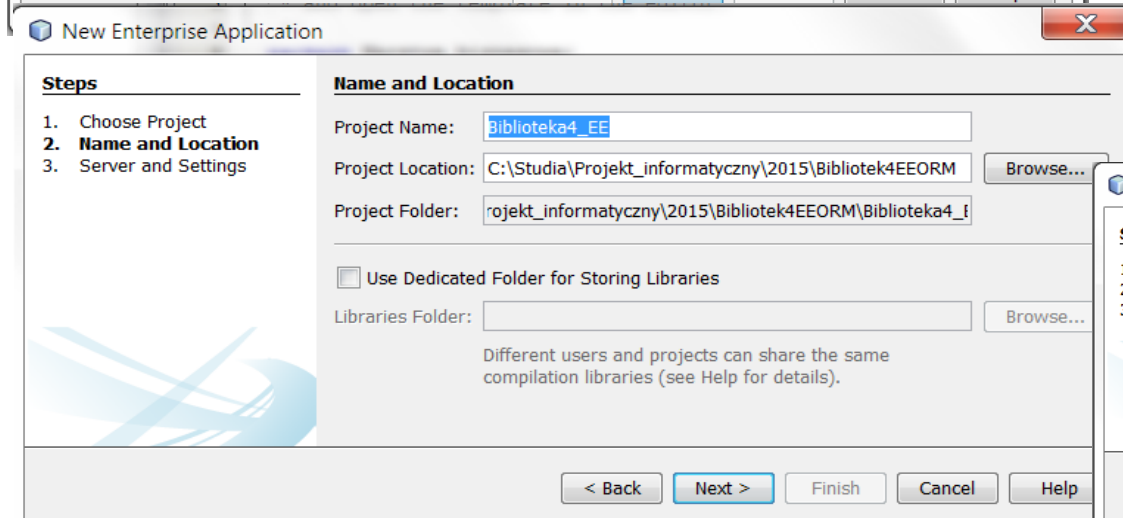
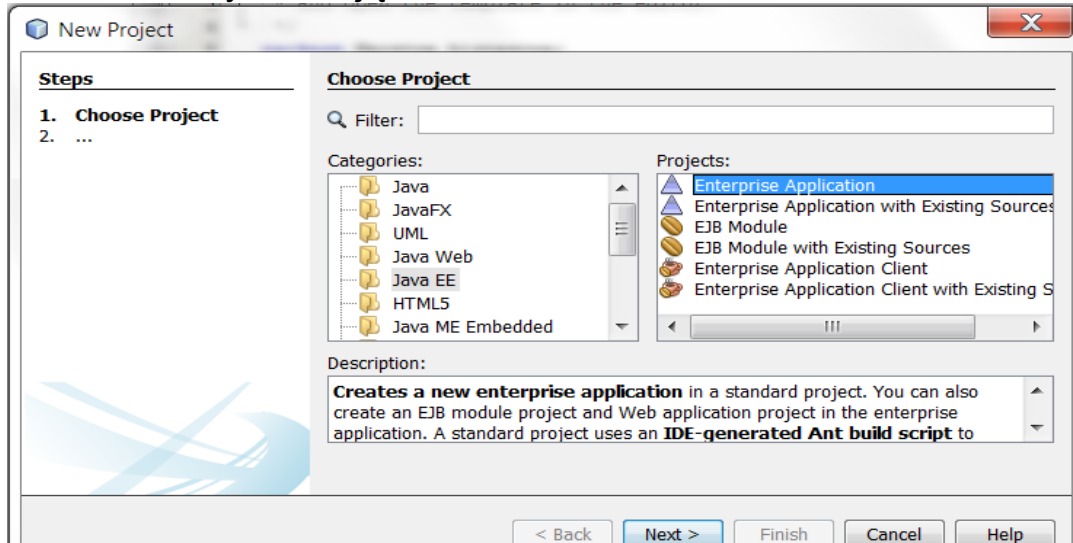
```
public ArrayList<String> tytul_y() {
    ArrayList<String> tytul_y = new ArrayList<String>();
    Iterator<Tytul_książki> it = tytul_y_książek.iterator();
    while (it.hasNext()) {
        tytul_y.add(it.next().toString());
    }
    return tytul_y;
}
public ArrayList<String> książki() {
    ArrayList<String> książki = new ArrayList<String>();
    Iterator<Tytul_książki> it = tytul_y_książek.iterator();
    while (it.hasNext()) {
        książki.addAll(it.next().książki());
    }
    return książki;
}
public void uaktualnij_dane(Tytul_książki tytul_y[], Książka książki[]) {
    tytul_y_książek.clear();
    for (int i = 0; i < tytul_y.length; i++) {
        tytul_y_książek.add(tytul_y[i]);
    }
}
```

Ten parametr nie jest wykorzystany w tej wersji programu, mimo, że na stronie 34 pokazano kod, z którego wynika, że ta tablica zawiera dane. W przypadku, gdyby nie działał mechanizm **@OneToMany(mappedBy = "mTytul_książki", cascade = CascadeType.ALL)** **private Collection<Książka> mKsiążka;**

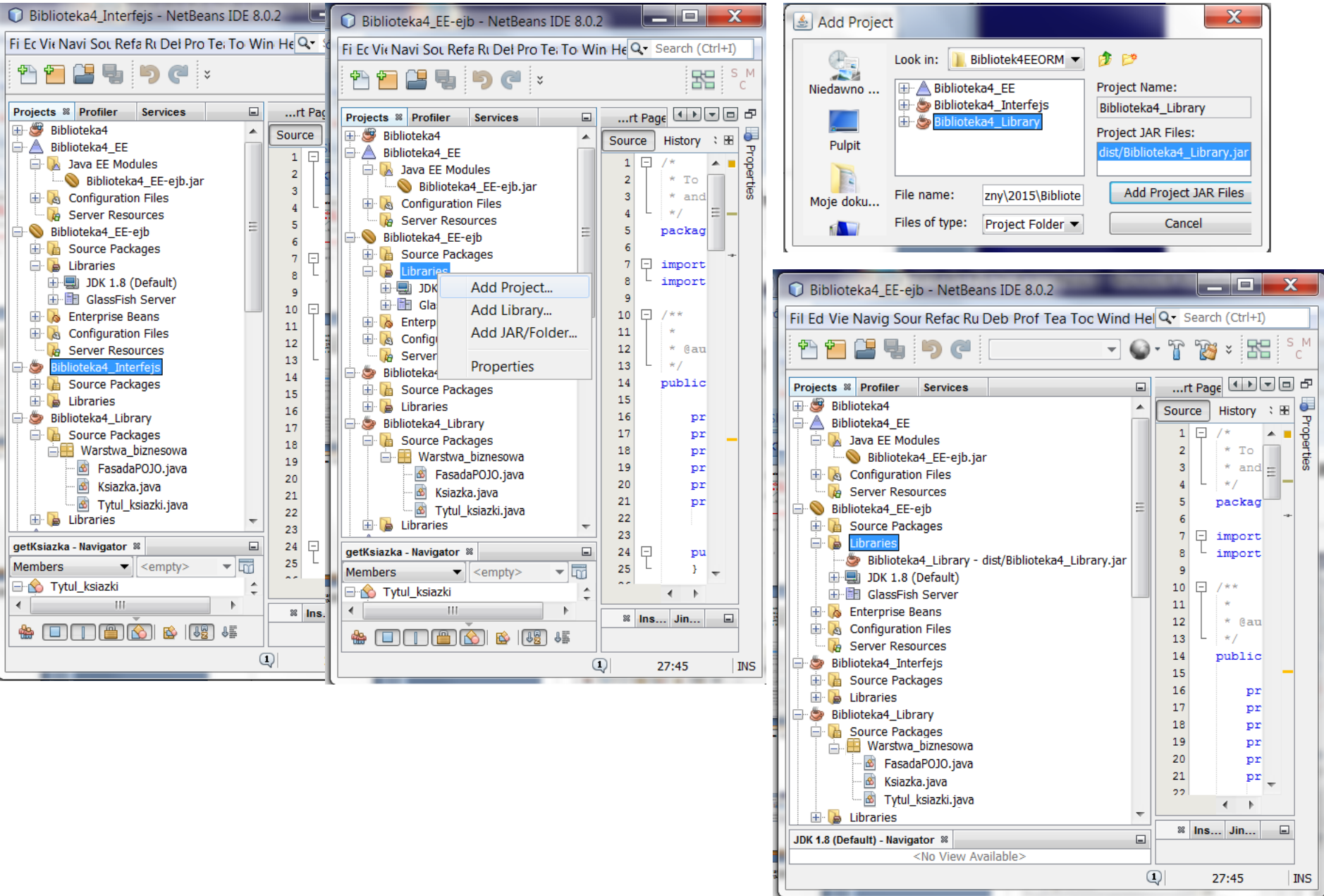
w klasie `Tytul_książki`, wtedy należałoby wykorzystać tablicę `książki`, wiążąc dane z obu tablic w dwukierunkowej relacji 1...wiele. Jeśli nie wystąpią problemy z tym mechanizmem **cascade**, można ten parametr `książki` pominąć. Na stronie 34 w metodzie `uaktualnij_dane()` - można wtedy usunąć również wywołanie metody `uaktualnij_książki()` i uaktualnić nagłówek wywołanej metody `fasada.uaktualnij_dane(tytul_y)`.

4. Wykonanie projektu EE

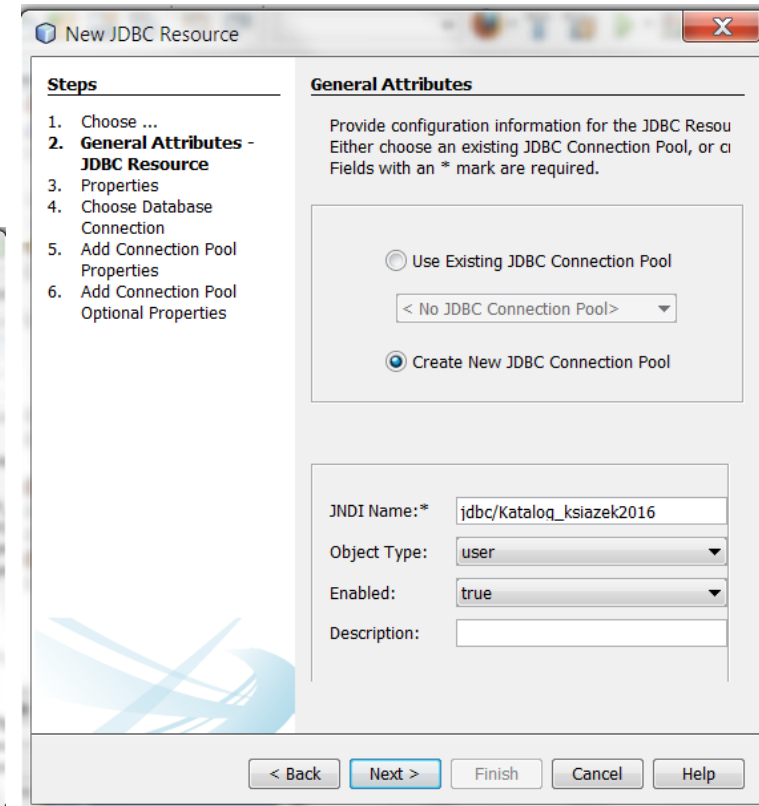
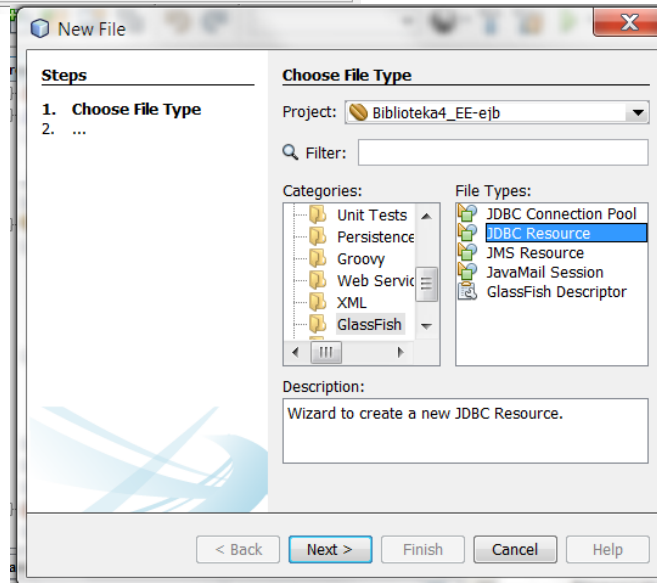
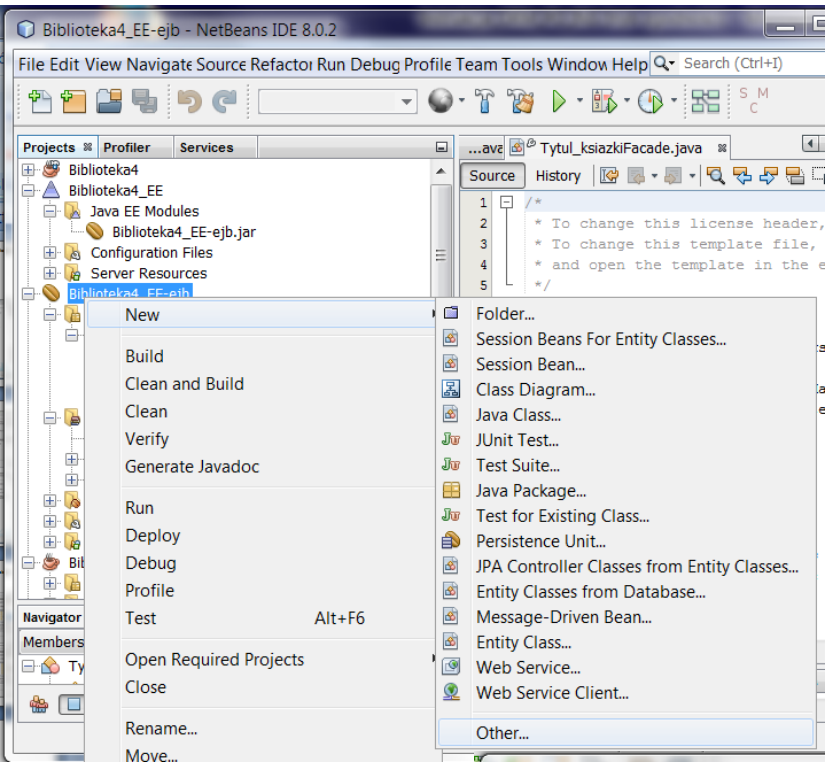
- 4.1. Wykonanie kolejnego projektu typu **Java Class Library** o nazwie **Biblioteka4_Interfejs** (wg p.2.1)
- 4.2. Wykonanie projektu **Biblioteka4_EE** typu **Java EE (File/New Project/Java/Java EE/Enterprise Application** i nacisnąć klawisz **Next**). Następnie w polu **Project Name** wpisać nazwę projektu **Biblioteka4_EE** po wyborze katalogu w polu **Project Location** i nacisnąć klawisz **Next**. W kolejnym formularzu wybrać server aplikacji **GlassFish**, **Java EE 7** oraz zaznaczyć moduł **EJB**. Następnie należy zakończyć klikając na klawisz **Finish**



4.3. Dodanie projektu **Biblioteka4_Library** do folderu **Libraries** modułu EJB **Biblioteka4_EE-ejb**



4.4. Wykonanie JDBC Resources dla serwera aplikacji GlassFish, dotyczące modułu Biblioteka4_EE-ejb



4.4. Wykonanie JDBC Resources dla serwera aplikacji GlassFish cd

New JDBC Resource

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

Additional Properties

Add additional configuration information for the resource jdbc/Katalog_ksiazek2016. Hit the Enter key to save values in the Properties table.

Properties:

Name	Value
------	-------

New JDBC Resource

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)

New JDBC Resource

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/Katalog_ksiazek2016
serverName	localhost
PortNumber	1527
DatabaseName	Katalog_ksiazek2016

New JDBC Resource

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:

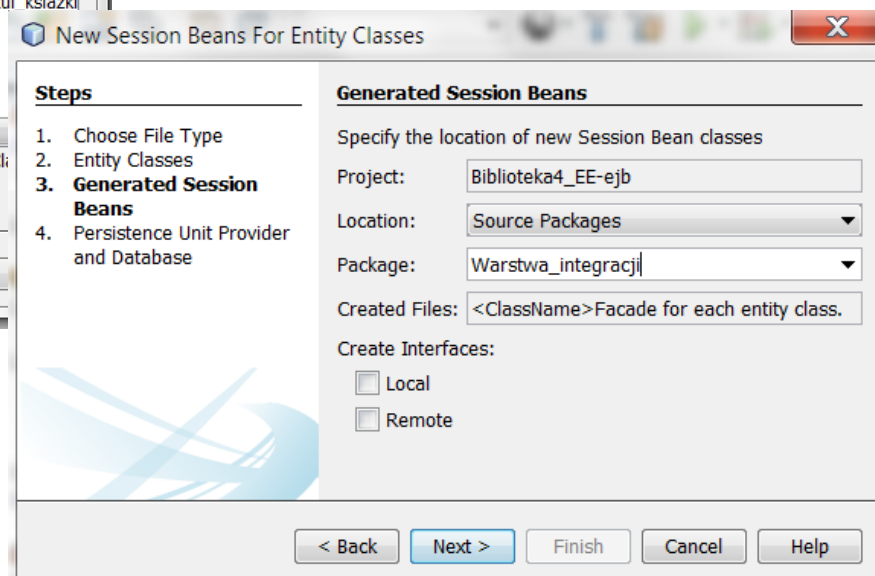
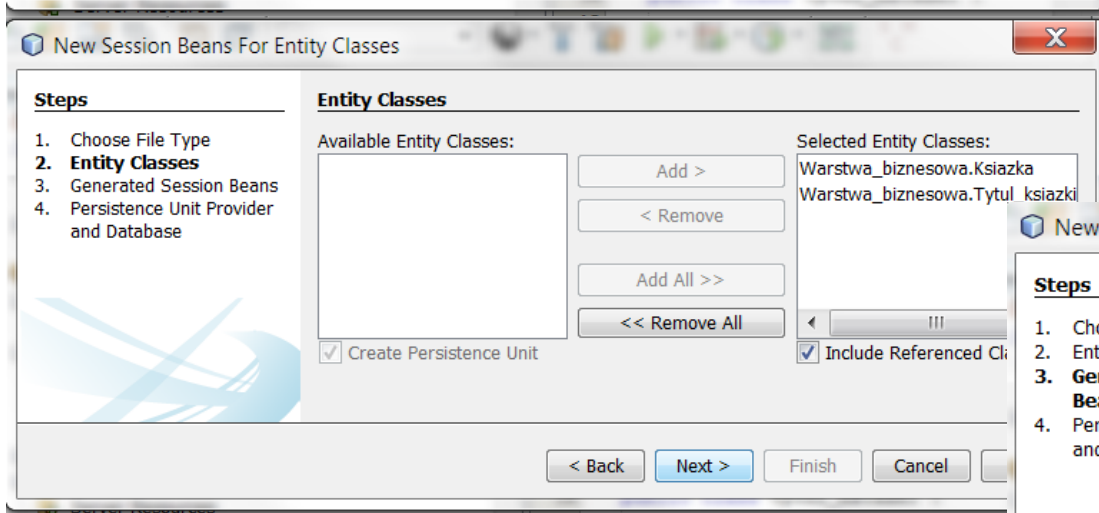
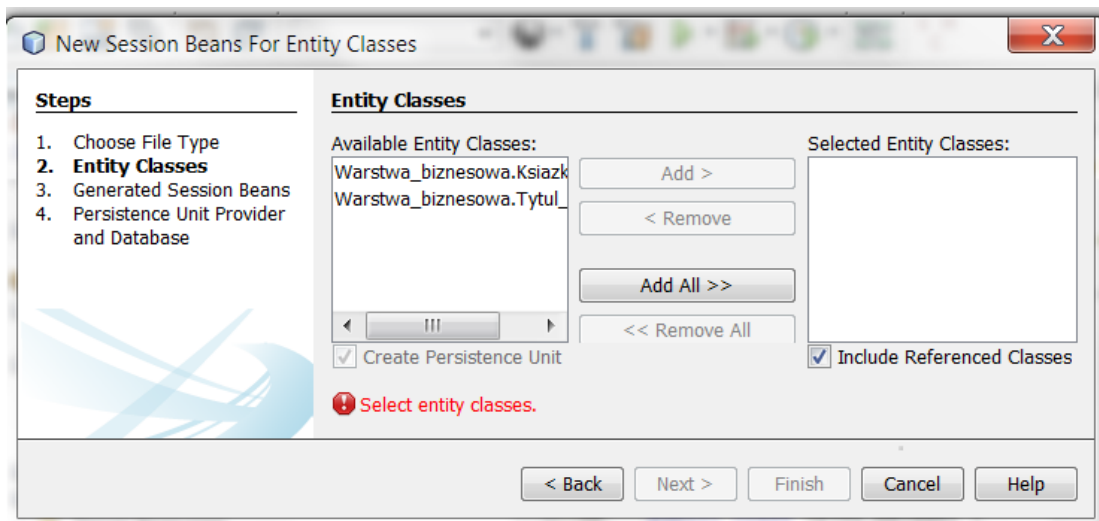
Non Transactional Connections:

Allow Non Component Callers:

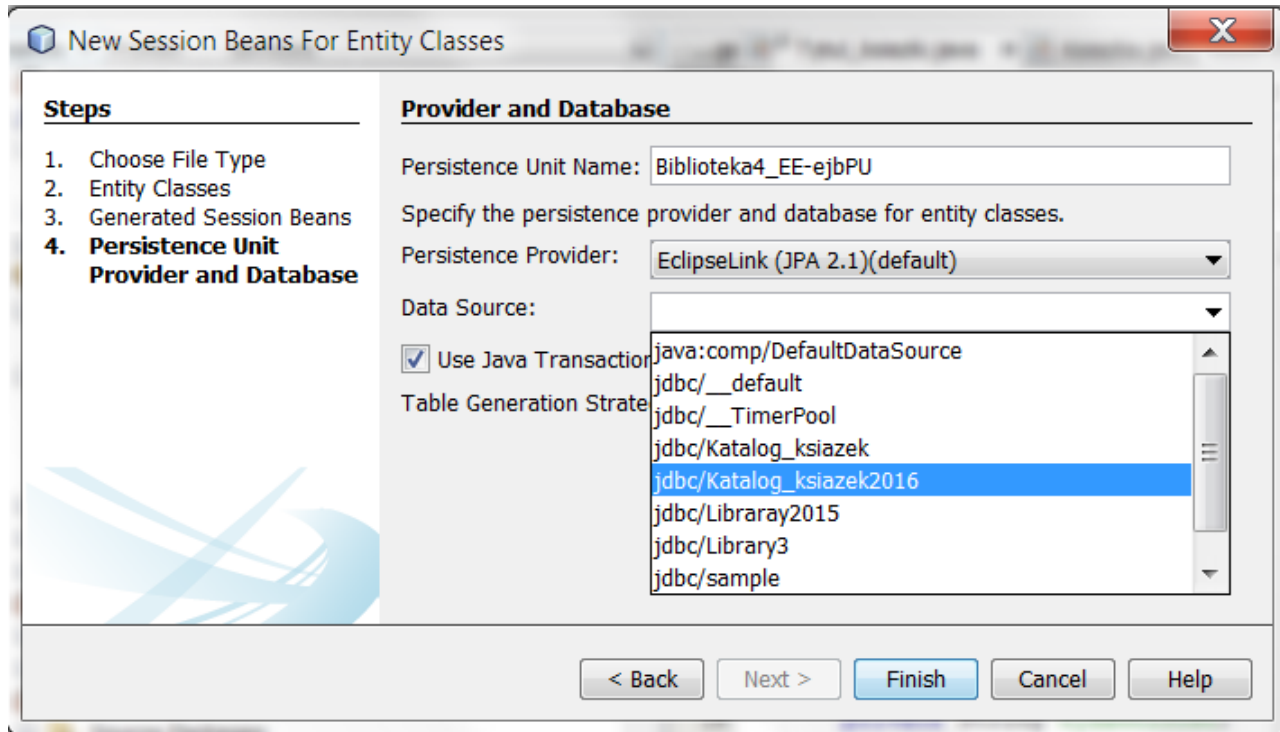
4.5. Wykonanie automatyczne kontrolerów ORM dla klas typu Entity: Tytul_ksiazki i Ksiazka w projekcie Biblioteka4_EE-ejb

The screenshot displays the NetBeans IDE 8.0.2 interface for the project 'Biblioteka4_EE-ejb'. The main editor shows the source code for 'Tytuł_ksiazki.java' and 'Ksiazka.java' in the 'Warstwa_biznesowa' package. A context menu is open over the project tree, listing various actions such as 'New', 'Build', 'Clean and Build', 'Run', and 'Test'. The 'New' option is selected, and a sub-menu is visible with 'Session Beans For Entity Classes' highlighted. Two 'New File' dialog boxes are overlaid on the IDE. The top dialog shows the 'Choose File Type' step with 'Enterprise JavaBeans' selected in the categories list and 'Session Beans For Entity Classes' selected in the file types list. The bottom dialog shows the same step with 'Session Beans For Entity Classes' selected in the file types list. The description for the selected file type reads: '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.'

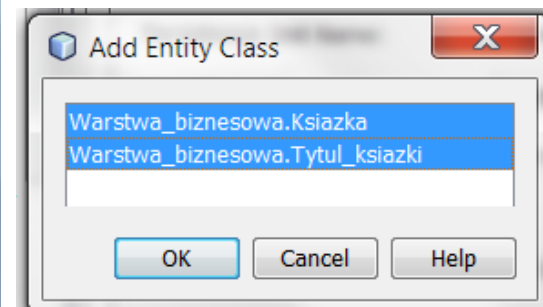
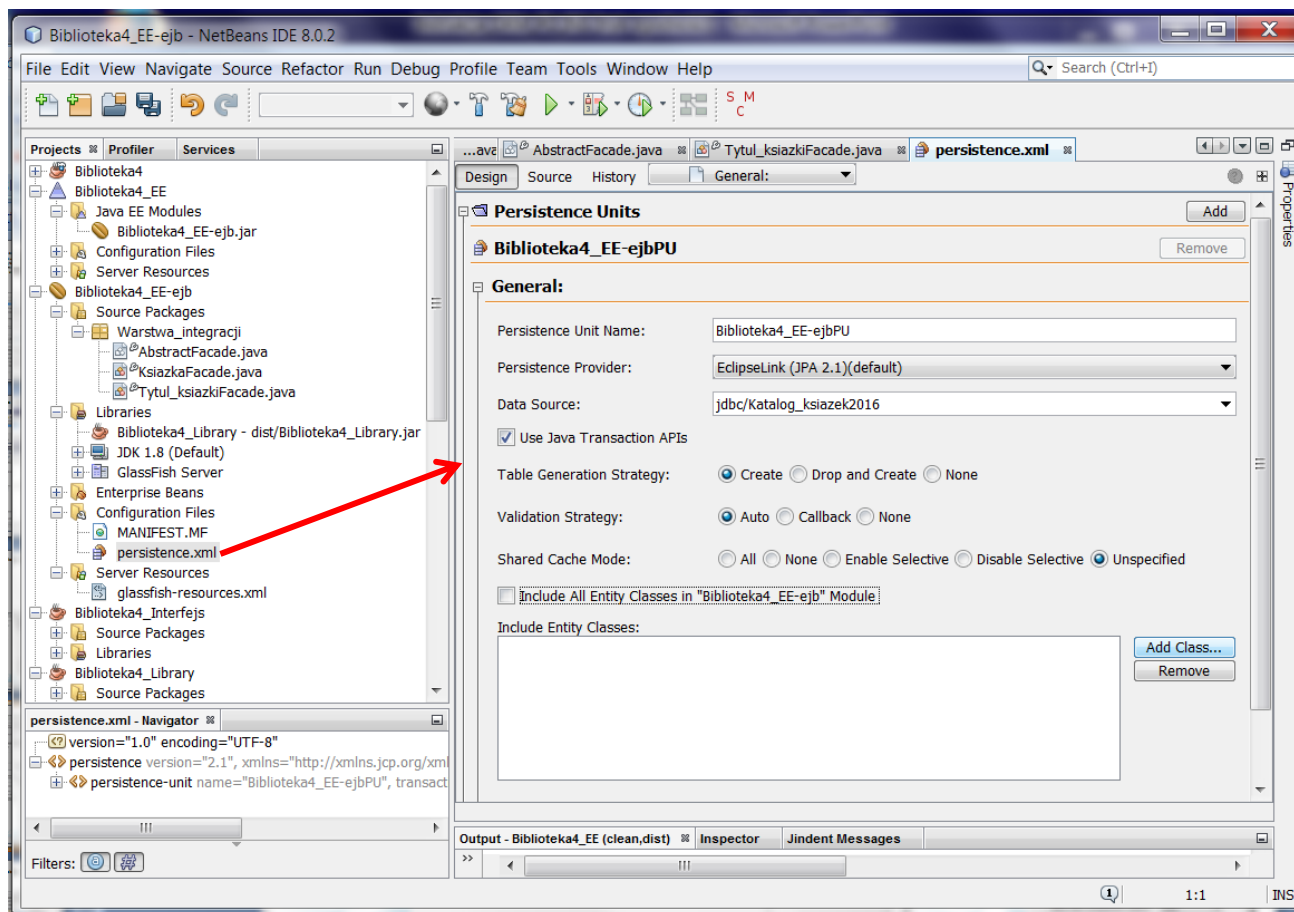
4.6. Należy wybrać encje przez **AddAll**, nacisnąć klawisz **Next**, i podać nazwę pakietu **Warstwa_integracji**, w którym należy umieścić nowe kontrolery typu EJB o dostępie domyślnym (lokalnym)



4.7. Należy wybrać identyfikator bazy danych, zdefiniowany w p.4.4 jako **JNDI Name**



4.8. W utworzonym automatycznie pliku **persistence.xml** umieścić jawnie klasy typu **Entity** przez anulowanie ustawienia **Include All EntityClasses** in „**Biblioteka4_EE-ejb**” i za pomocą przycisku **Add Class...** wybrać w formularzu wszystkie klasy typu **Entity** (**Tytuł_książki** i **Książka**)



4.8. cd

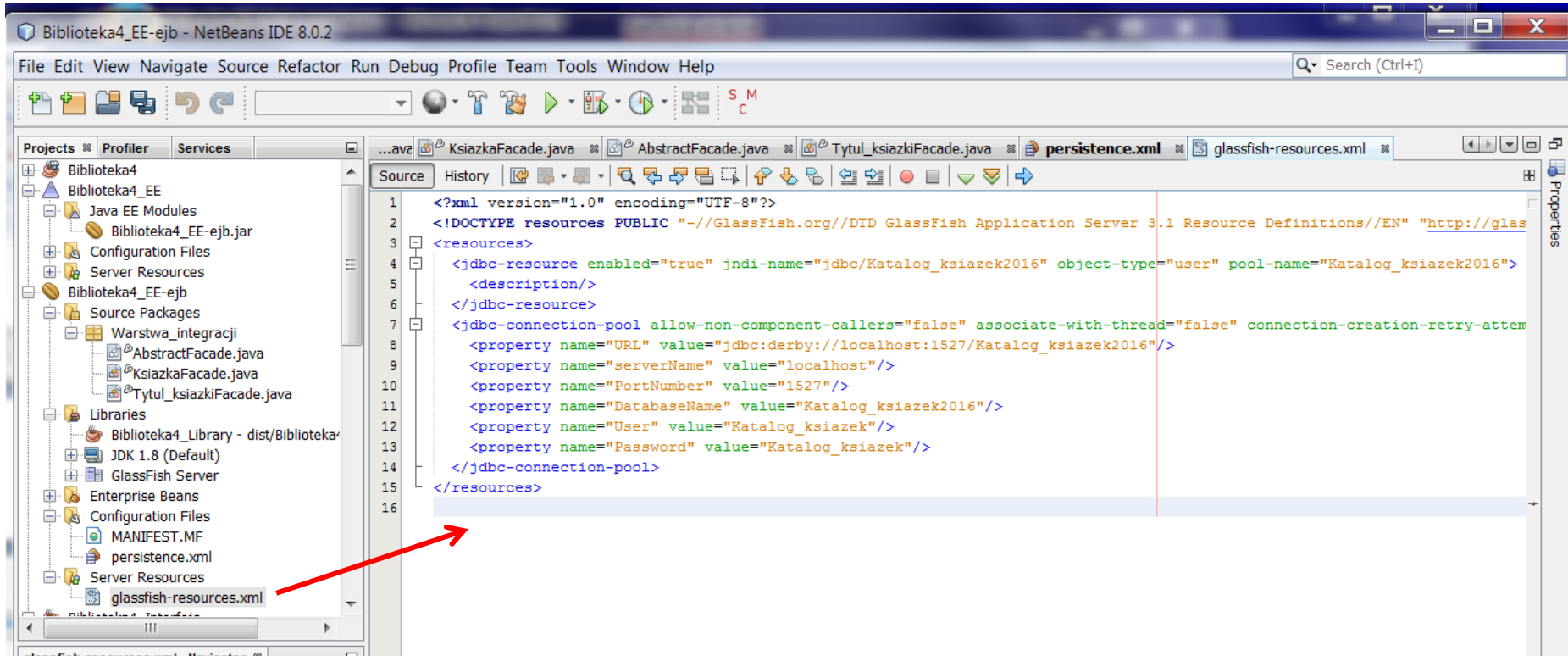
The screenshot displays the NetBeans IDE interface for a project named 'Biblioteka4_EE-ejb'. The main window shows the 'Persistence Units' configuration for 'Biblioteka4_EE-ejbPU'. The 'General' tab is selected, and the following settings are visible:

- Persistence Unit Name: Biblioteka4_EE-ejbPU
- Persistence Provider: EclipseLink (JPA 2.1)(default)
- Data Source: jdbc/Katalog_ksiazek2016
- Use Java Transaction APIs
- Table Generation Strategy: Create Drop and Create None
- Validation Strategy: Auto Callback None
- Shared Cache Mode: All None Enable Selective Disable Selective Unspecified
- Include All Entity Classes in "Biblioteka4_EE-ejb" Module
- Include Entity Classes: Warstwa_biznesowa.Ksiazka, Warstwa_biznesowa.Tytul_ksiazki

A red arrow points to the 'Use Java Transaction APIs' checkbox. The 'persistence.xml' file is open in the Navigator, showing the following XML snippet:

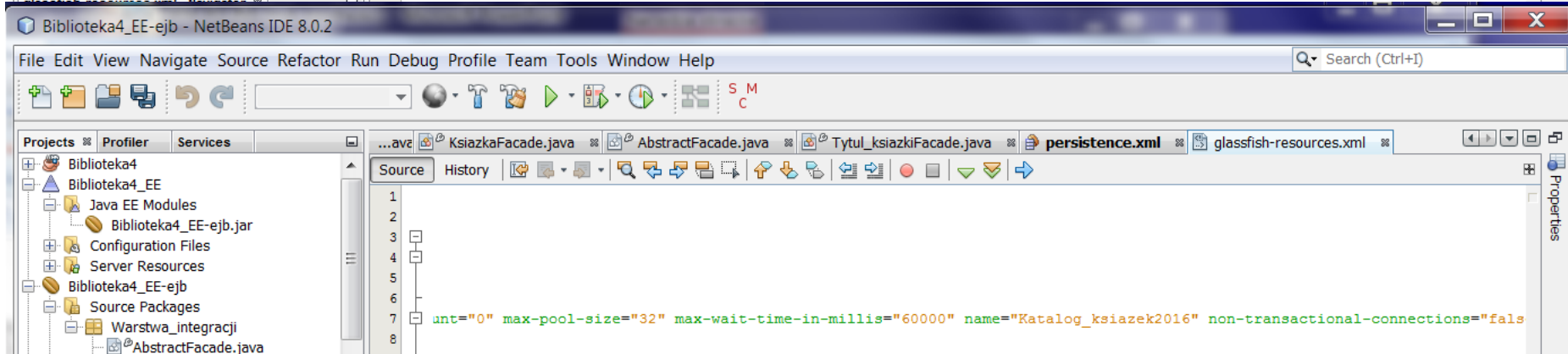
```
<?xml version="1.0" encoding="UTF-8"
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml
<persistence-unit name="Biblioteka4_EE-ejbPU", transact
```


4.9. Zawartość pliku **glassfish-resources.xml**, utworzonego razem z plikiem **persistence.xml**



The screenshot shows the NetBeans IDE interface with the `glassfish-resources.xml` file open in the editor. The Project Explorer on the left shows the file's location within the `Biblioteka4` project. A red arrow points from the file in the Project Explorer to the editor.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE resources PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 Resource Definitions//EN" "http://glas
3
4 <resources>
5   <jdbc-resource enabled="true" jndi-name="jdbc/Katalog_ksiazek2016" object-type="user" pool-name="Katalog_ksiazek2016">
6     <description/>
7   </jdbc-resource>
8   <jdbc-connection-pool allow-non-component-callers="false" associate-with-thread="false" connection-creation-retry-attem
9     <property name="URL" value="jdbc:derby://localhost:1527/Katalog_ksiazek2016"/>
10    <property name="serverName" value="localhost"/>
11    <property name="PortNumber" value="1527"/>
12    <property name="DatabaseName" value="Katalog_ksiazek2016"/>
13    <property name="User" value="Katalog_ksiazek"/>
14    <property name="Password" value="Katalog_ksiazek"/>
15  </jdbc-connection-pool>
16 </resources>
```



This screenshot shows the same NetBeans IDE interface, but the editor view is partially obscured, showing only the bottom part of the XML file.

```
unt="0" max-pool-size="32" max-wait-time-in-millis="60000" name="Katalog_ksiazek2016" non-transactional-connections="fals
```

4.10. Utworzona automatycznie klasa abstrakcyjna **AbstractFacade** reprezentująca klasę bazową kontrolerów ORM typu EJB

```
package Warstwa_integracji;
import java.util.List;
import javax.persistence.EntityManager;
/**...4 lines */
public abstract class AbstractFacade<T> {
    private Class<T> entityClass;

    public AbstractFacade(Class<T> entityClass) {
        this.entityClass = entityClass;
    }

    protected abstract EntityManager getEntityManager();

    public void create(T entity) {
        getEntityManager().persist(entity);
    }

    public void edit(T entity) {
        getEntityManager().merge(entity);
    }

    public void remove(T entity) {
        getEntityManager().remove(getEntityManager().merge(entity));
    }

    public T find(Object id) {
        return getEntityManager().find(entityClass, id);
    }

    public List<T> findAll() {
        javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
        cq.select(cq.from(entityClass));
        return getEntityManager().createQuery(cq).getResultList();
    }

    public List<T> findRange(int[] range) {
        javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
        cq.select(cq.from(entityClass));
        javax.persistence.Query q = getEntityManager().createQuery(cq);
        q.setMaxResults(range[1] - range[0] + 1);
        q.setFirstResult(range[0]);
        return q.getResultList();
    }

    public int count() {
        javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();
        javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
        cq.select(getEntityManager().getCriteriaBuilder().count(rt));
        javax.persistence.Query q = getEntityManager().createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    }
}
```

4.11. Utworzony automatycznie kontroler EJB dla klasy typu Entity dla klas: **Tytul_książki** (na lewo) oraz po wstawieniu kodu (na prawo) do dodania zbioru obiektów typu **Tytul_książki** do bazy danych (**dodaj_tytuly**) oraz pobranie tych danych z bazy danych do wyświetlenia (**tytuly_**)

```
...5 lines
package Warstwa_integracji;

import Warstwa_biznesowa.Tytul_książki;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**...4 lines */
@Stateless
public class Tytul_książkiFacade extends AbstractFacade<Tytul_książki> {
    @PersistenceContext(unitName = "Biblioteka4_EE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public Tytul_książkiFacade() {
        super(Tytul_książki.class);
    }
}
```

```
package Warstwa_integracji;
import Warstwa_biznesowa.Tytul_książki;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**...4 lines */
@Stateless
public class Tytul_książkiFacade extends AbstractFacade<Tytul_książki> {
    @PersistenceContext(unitName = "Biblioteka4_EE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public Tytul_książkiFacade() {
        super(Tytul_książki.class);
    }

    public ArrayList<String> tytuly_() {
        ArrayList<String> tytuly = new ArrayList();
        Tytul_książki[] tytuly_ = this.findAll().toArray(new Tytul_książki[0]);
        for (Tytul_książki t : tytuly_) {
            tytuly.add(t.toString());
        }
        return tytuly;
    }

    public void dodaj_tytuly(List<Tytul_książki> tytuly) {
        for (Tytul_książki tytul : tytuly) {
            if (tytul.getIdTytul() == null) {
                getEntityManager().persist(tytul);
            }
        }
    }
}
```

4.12. Utworzony automatycznie kontroler EJB dla klasy typu Entity dla klas: **Ksiazka** (na lewo) oraz po wstawieniu kodu (na prawo) do dodania zbioru obiektów typu **Ksiazka** do bazy danych (**dodaj_ksiazki**) oraz pobranie tych danych z bazy danych do wyświetlenia (**ksiazki_**)

```
package Warstwa_integracji;

import Warstwa_biznesowa.Ksiazka;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**...4 lines */
@Stateless
public class KsiazkaFacade extends AbstractFacade<Ksiazka> {
    @PersistenceContext(unitName = "Biblioteka4_EE-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public KsiazkaFacade() {
        super(Ksiazka.class);
    }
}
```

```
package Warstwa_integracji;
import Warstwa_biznesowa.Ksiazka;
import Warstwa_biznesowa.Tytul_ksiazki;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**...4 lines */
@Stateless
public class KsiazkaFacade extends AbstractFacade<Ksiazka> {
    @PersistenceContext(unitName = "Biblioteka4_EE-ejbPU")
    private EntityManager em;

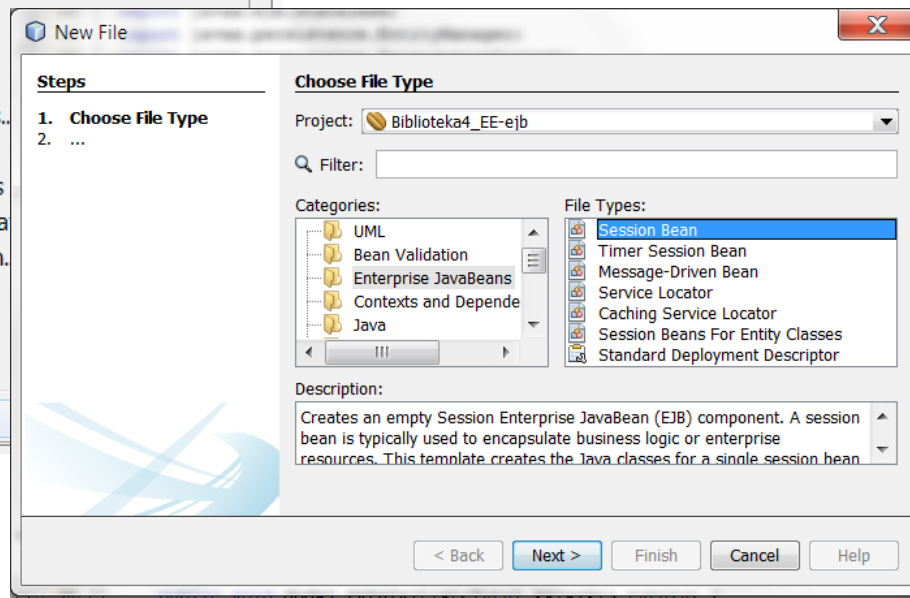
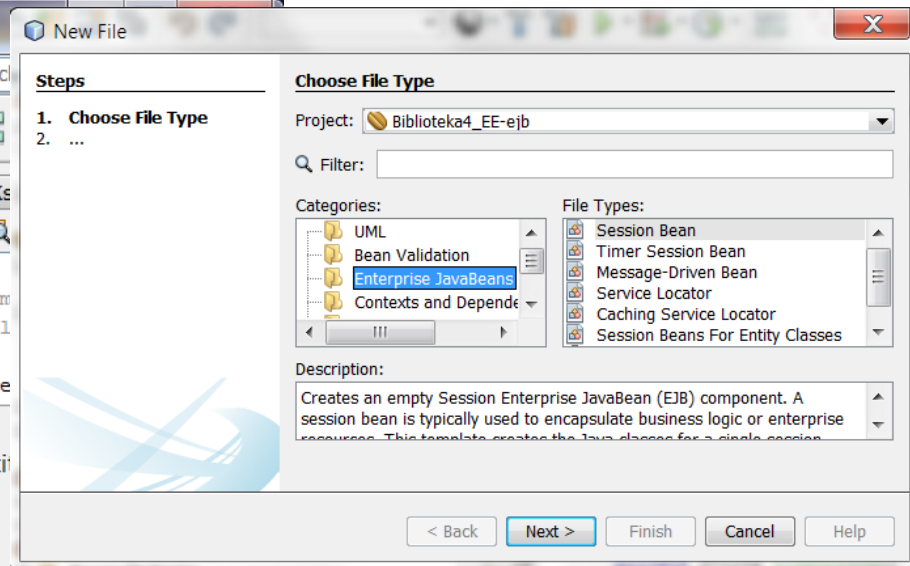
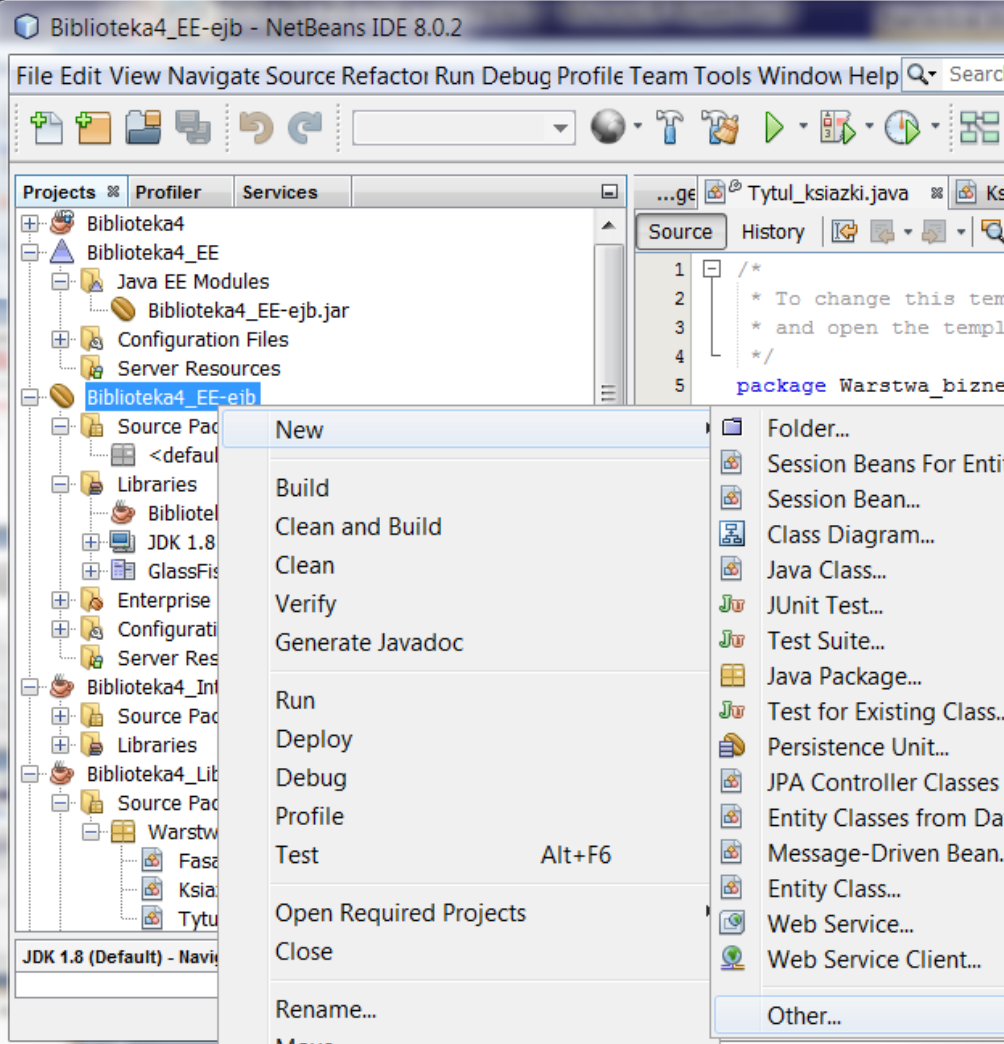
    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public KsiazkaFacade() {
        super(Ksiazka.class);
    }

    public ArrayList<String> ksiazki_() {
        ArrayList<String> ksiazki = new ArrayList();
        Ksiazka[] ksiazki_ = this.findAll().toArray(new Ksiazka[0]);
        for (Ksiazka k : ksiazki_) {
            ksiazki.add(k.toString());
        }
        return ksiazki;
    }

    public void dodaj_ksiazki(List<Tytul_ksiazki> tytuly) {
        for (Tytul_ksiazki tytul : tytuly) {
            if (tytul.getIdTytul() == null) {
                continue;
            }
            for (Ksiazka ksiazka : tytul.getMKsiazka()) {
                if (ksiazka.getIdKsiazka() == null) {
                    getEntityManager().persist(ksiazka);
                }
            }
        }
    }
}
```

4.13. Dodanie klasy **Fasadaejb** typu EJB (Session Bean) o dostępie Remote do modułu **Biblioteka4_EE-ejb** do obsługi komunikacji typu Klient-Serwer



4.13. Dodanie klasy **Fasadaejb** typu EJB (Session Bean) o dostępie Remote do modułu **Biblioteka4_EE-ejb** do obsługi komunikacji typu Klient-Server cd

The image shows the NetBeans IDE interface with two main windows. On the left is the 'New Session Bean' dialog, and on the right is the IDE workspace showing the project structure and the source code of the 'Fasadaejb.java' file.

New Session Bean Dialog:

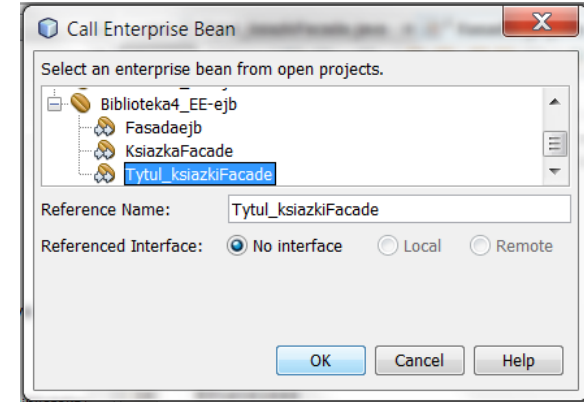
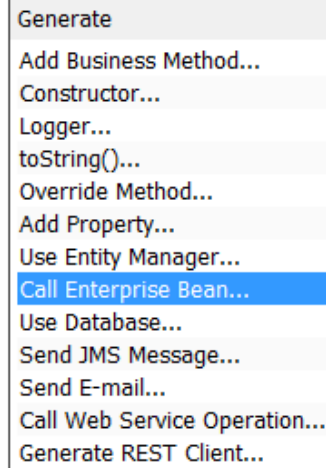
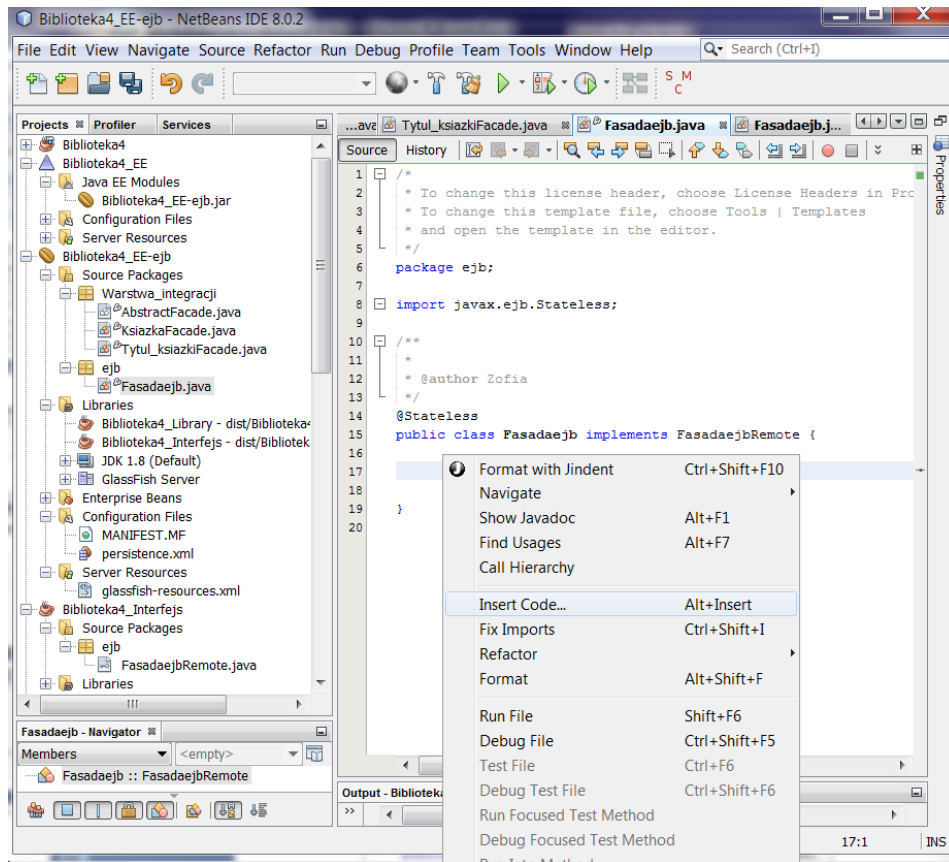
- Steps:**
 1. Choose File Type
 2. Name and Location
- Name and Location:**
 - EJB Name: Fasadaejb
 - Project: Biblioteka4_EE-ejb
 - Location: Source Packages
 - Package:.ejb
- Session Type:**
 - Stateless
 - Stateful
 - Singleton
- Create Interface:**
 - Local
 - Remote in project: Biblioteka4_Interfejs

IDE Workspace:

- Project Structure:** Biblioteka4_EE-ejb.jar, Configuration Files, Server Resources, Biblioteka4_EE-ejb, Source Packages, Warstwa_integracji, AbstractFacade.java, KsiazkaFacade.java, Tytul_ksiazkiFacade.java,.ejb, Fasadaejb.java, Libraries, Biblioteka4_Library - dist/Biblioteka4, Biblioteka4_Interfejs - dist/Biblioteka4, JDK 1.8 (Default), GlassFish Server, Enterprise Beans, Configuration Files, MANIFEST.MF, persistence.xml, Server Resources, glassfish-resources.xml, Biblioteka4_Interfejs, Source Packages,.ejb, FasadaejbRemote.java, Libraries, Biblioteka4_Library, Source Packages, Libraries.
- Source Code (Fasadaejb.java):**

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates in the IDE window.
4  * and open the template in the editor.
5  */
6  package.ejb;
7
8  import javax.ejb.Stateless;
9
10 /**
11 *
12 * @author Zofia
13 */
14 @Stateless
15 public class Fasadaejb implements FasadaejbRemote {
16
17     // Add business logic below. (Right-click in editor to see more options)
18     // "Insert Code > Add Business Method"
19 }
20
```

4.14. Wykonanie połączenia z kontrolerami ORM typu EJB dla encji **Tytuł_książki** i **Książka** w klasie **Fasadejb** o dostępie Remote za pomocą tzw adnotacji.



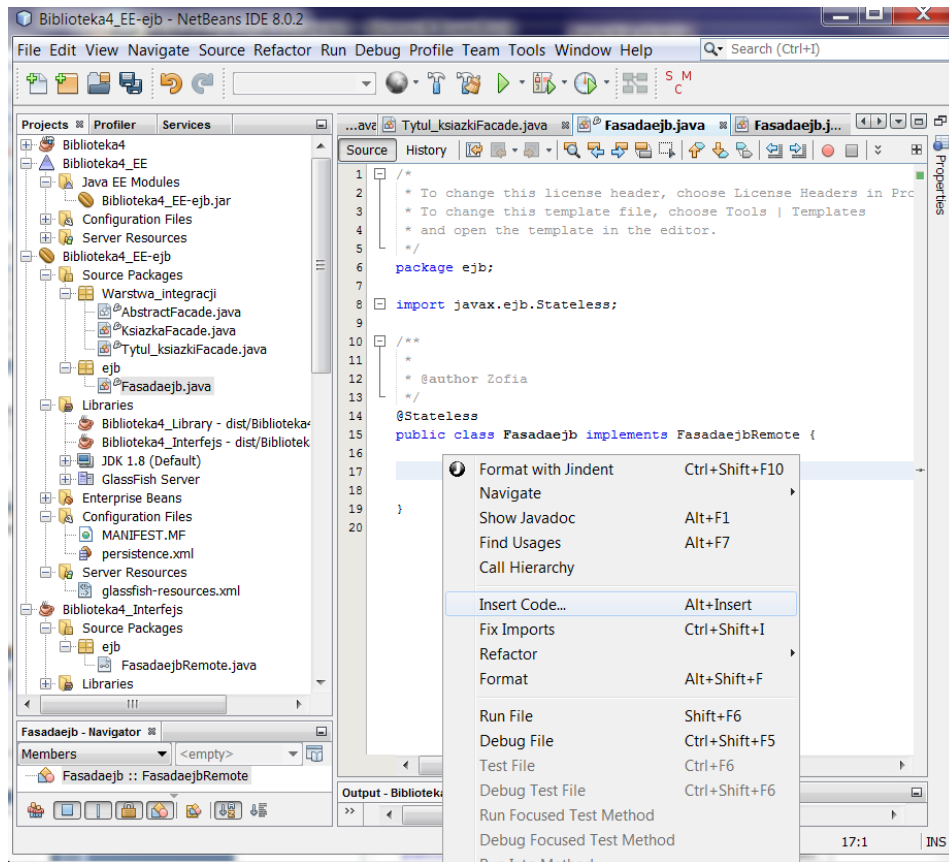
```
+ |...5 lines |
package ejb;

- |import Warstwa_integracji.Tytuł_książkiFacade;
  |import javax.ejb.EJB;
  |import javax.ejb.Stateless;

+ |/**...4 lines */|
@Stateless
public class Fasadaejb implements FasadaejbRemote {
    @EJB
    private Tytuł_książkiFacade tytuł_książkiFacade;

}
```

4.14. Wykonanie połączenia z kontrolerami ORM typu EJB dla encji Tytul_książki i Książka w klasie Fasadejeb o dostępie Remote za pomocą tzw adnotacji cd.

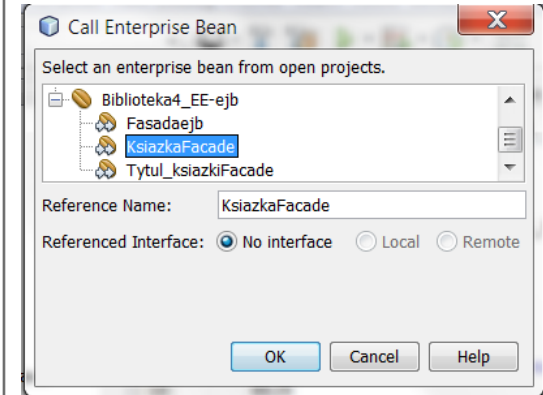


- Generate
- Add Business Method...
- Constructor...
- Logger...
- toString()...
- Override Method...
- Add Property...
- Use Entity Manager...
- Call Enterprise Bean...
- Use Database...
- Send JMS Message...
- Send E-mail...
- Call Web Service Operation...
- Generate REST Client...

```
package.ejb;

import Warstwa_integracji.KsiazkaFacade;
import Warstwa_integracji.Tytul_ksiazkiFacade;
import javax.ejb.EJB;
import javax.ejb.Stateless;

/**...4 lines */
@Stateless
public class Fasadaejb implements FasadaejbRemote {
    @EJB
    private KsiazkaFacade ksiazkaFacade;
    @EJB
    private Tytul_ksiazkiFacade tytul_ksiazkiFacade;
}
```



4.15. Wykonanie deklaracji metod w interfejsie **FasadaejbRemote** klasy **Fasadaejb** w projekcie **Biblioteka4_Interfejs**, wykonanym w p.4.1, typu **Java Class Library** (konieczny dla ziaren EJB typu Remote)

```
package ejb1;

import java.util.ArrayList;
import javax.ejb.Remote;

/**...4 lines */
@Remote
public interface FasadaejbRemote {
    //operacje na danych przechowywanych w pamieci
    public void dodaj_tytul(String dane_tytul[]);

    public void dodaj_ksiazke(String dane[]);

    public ArrayList<String> tytuly();

    public ArrayList<String> ksiazki();

    //.....
    //operacje na bazie danych

    public void uaktualnij_tytuly() throws Exception;

    public void uaktualnij_ksiazki() throws Exception;

    public void uaktualnij_dane() throws Exception;

    public void dodaj_tytuly() throws Exception;

    public void dodaj_ksiazki() throws Exception;

    public ArrayList<String> ksiazki_() throws Exception;

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

4.16. Implementacja metod w klasie **Fasadaejb**, zadeklarowanych w interfejsie **FasadaejbRemote** w projekcie **Biblioteka4_Interfejs**

```
package ejb;
import Warstwa_biznesowa.FasadaPOJO;
import Warstwa_biznesowa.Ksiazka;
import Warstwa_biznesowa.Tytul_ksiazki;
import Warstwa_integracji.KsiazkaFacade;
import Warstwa_integracji.Tytul_ksiazkiFacade;
import java.util.ArrayList;
import javax.annotation.PostConstruct;
import javax.ejb.EJB;
import javax.ejb.Stateless;

/**...4 lines */
@Stateless
public class Fasadaejb implements FasadaejbRemote {
    @EJB
    private KsiazkaFacade ksiazkaFacade;
    @EJB
    private Tytul_ksiazkiFacade tytul_ksiazkiFacade;

    final private FasadaPOJO fasada = new FasadaPOJO();
    private Tytul_ksiazki tytuly[];
    private Ksiazka ksiazki[];

    @PostConstruct
    public void init() {
        try {
            uaktualnij_dane();
        } catch (Exception e) {
        }
    }

    public void dodaj_tytul(String dane_tytul[]) {
        fasada.dodaj_tytul(dane_tytul);
    }

    public void dodaj_ksiazke(String dane[]) {
        fasada.dodaj_ksiazke(dane);
    }

    public ArrayList<String> tytuly() {
        return fasada.tytuly();
    }

    public ArrayList<String> ksiazki() {
        return fasada.ksiazki();
    }
}
```

```
public void uaktualnij_tytuly() throws Exception {
    tytuly = tytul_ksiazkiFacade.findAll().toArray(new Tytul_ksiazki[0]);
}

public void uaktualnij_ksiazki() throws Exception {
    ksiazki = ksiazkaFacade.findAll().toArray(new Ksiazka[0]);
}

public void uaktualnij_dane() throws Exception {
    uaktualnij_tytuly();
    uaktualnij_ksiazki();
    fasada.uaktualnij_dane(tytuly, ksiazki);
}

public void dodaj_tytuly() throws Exception {
    tytul_ksiazkiFacade.dodaj_tytuly(fasada.getTytuly_ksiazek());
}

public void dodaj_ksiazki() throws Exception {
    ksiazkaFacade.dodaj_ksiazki(fasada.getTytuly_ksiazek());
}

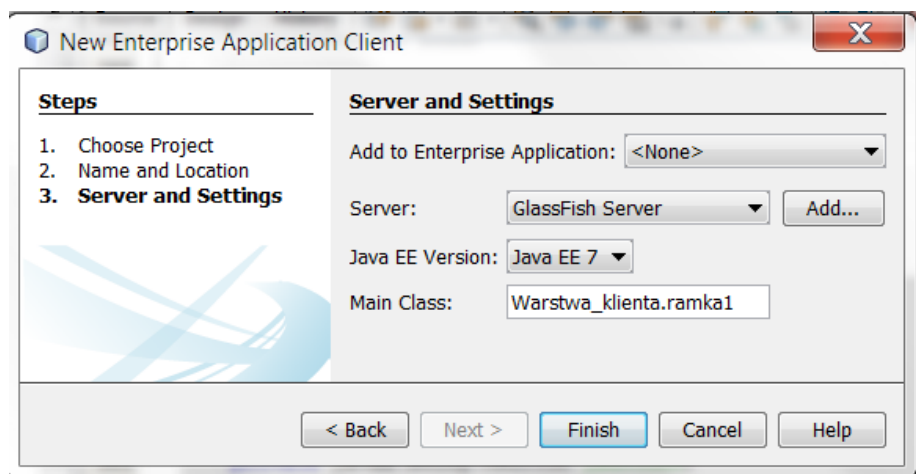
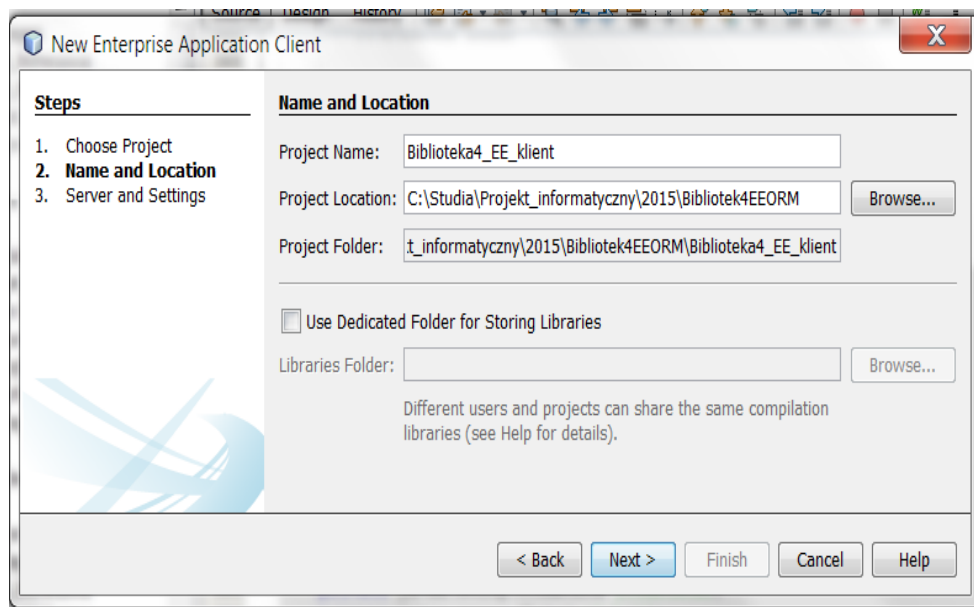
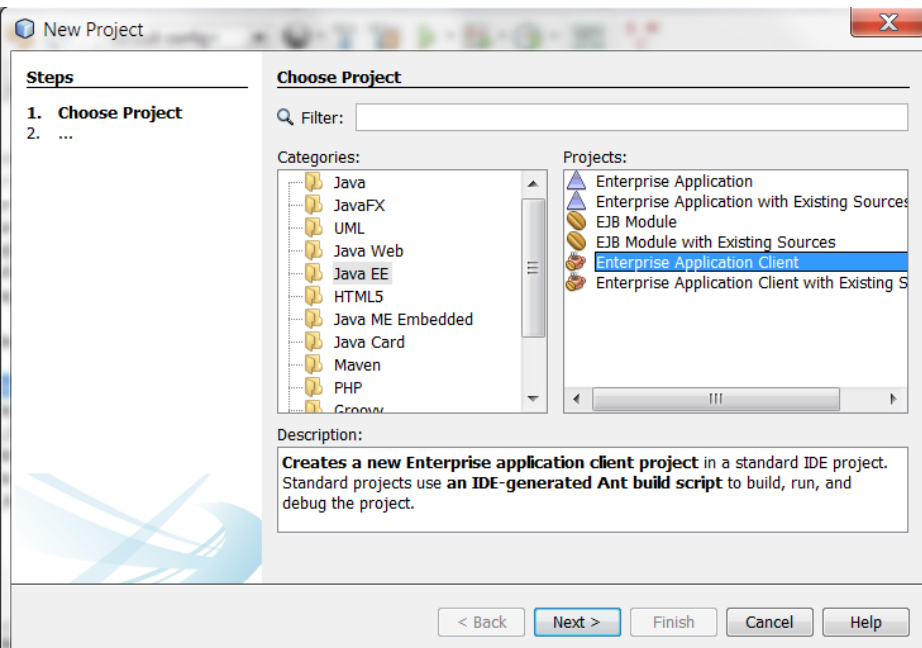
public ArrayList<String> ksiazki_() throws Exception {
    return ksiazkaFacade.ksiazki_();
}

public ArrayList<String> tytuly_() throws Exception {
    return tytul_ksiazkiFacade.tytuly_();
}
}
```

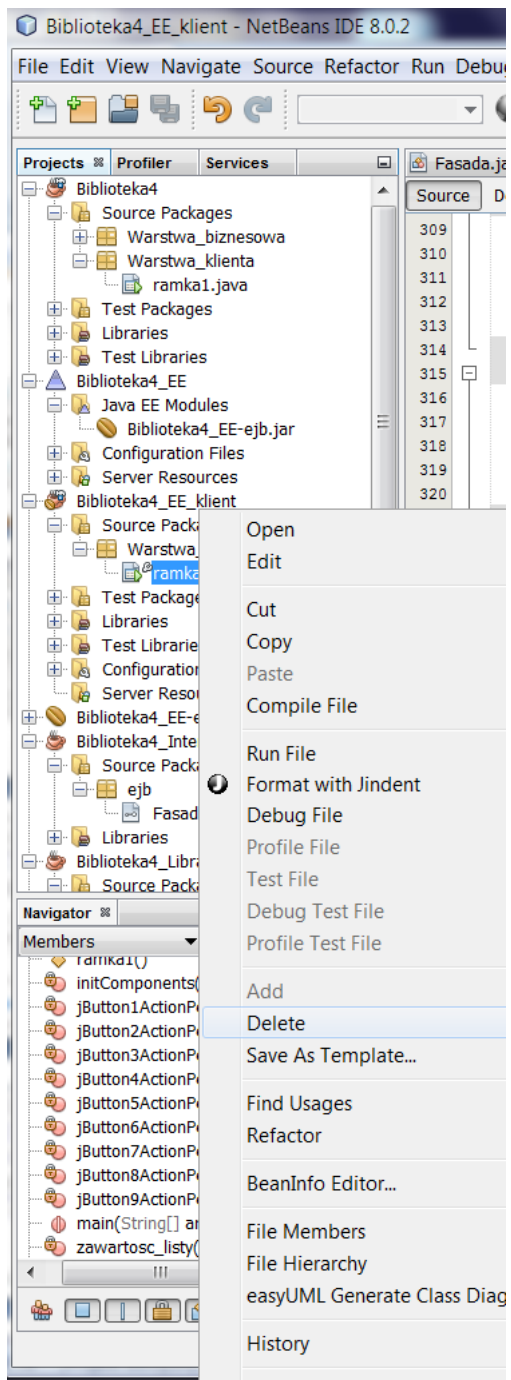
4.17. Dodanie adnotacji **Override** do wykonanych metod w klasie **Fasadaejb**

```
}  
@Override  
public void dodaj_tytul(String dane_tytul[]) {...2 lines }  
  
@Override  
public void dodaj_książke(String dane[]) {...2 lines }  
  
@Override  
public ArrayList<String> tytuly() {...2 lines }  
  
@Override  
public ArrayList<String> książki () {...2 lines }  
  
@Override  
public void uaktualnij_tytuly() throws Exception {...3 lines }  
  
@Override  
public void uaktualnij_książki() throws Exception {...3 lines }  
  
@Override  
public void uaktualnij_dane() throws Exception {...5 lines }  
  
@Override  
public void dodaj_tytuly() throws Exception {...3 lines }  
  
@Override  
public void dodaj_książki() throws Exception {...3 lines }  
  
@Override  
public ArrayList<String> książki_() throws Exception {...3 lines }  
  
@Override  
public ArrayList<String> tytuly_() throws Exception {...3 lines }  
}
```

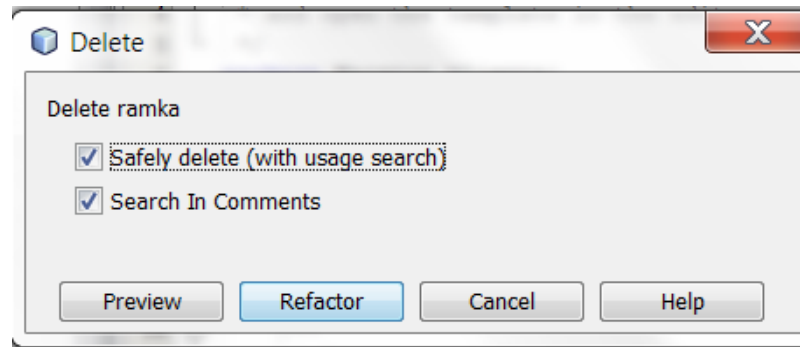
4.18. Dodanie projektu EE Biblioteka4_EE_klient typu Enterprise Application Client



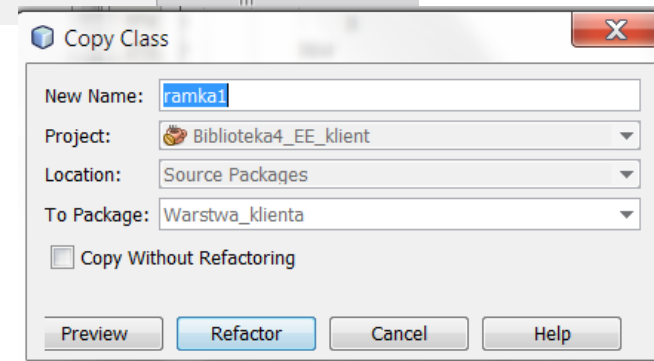
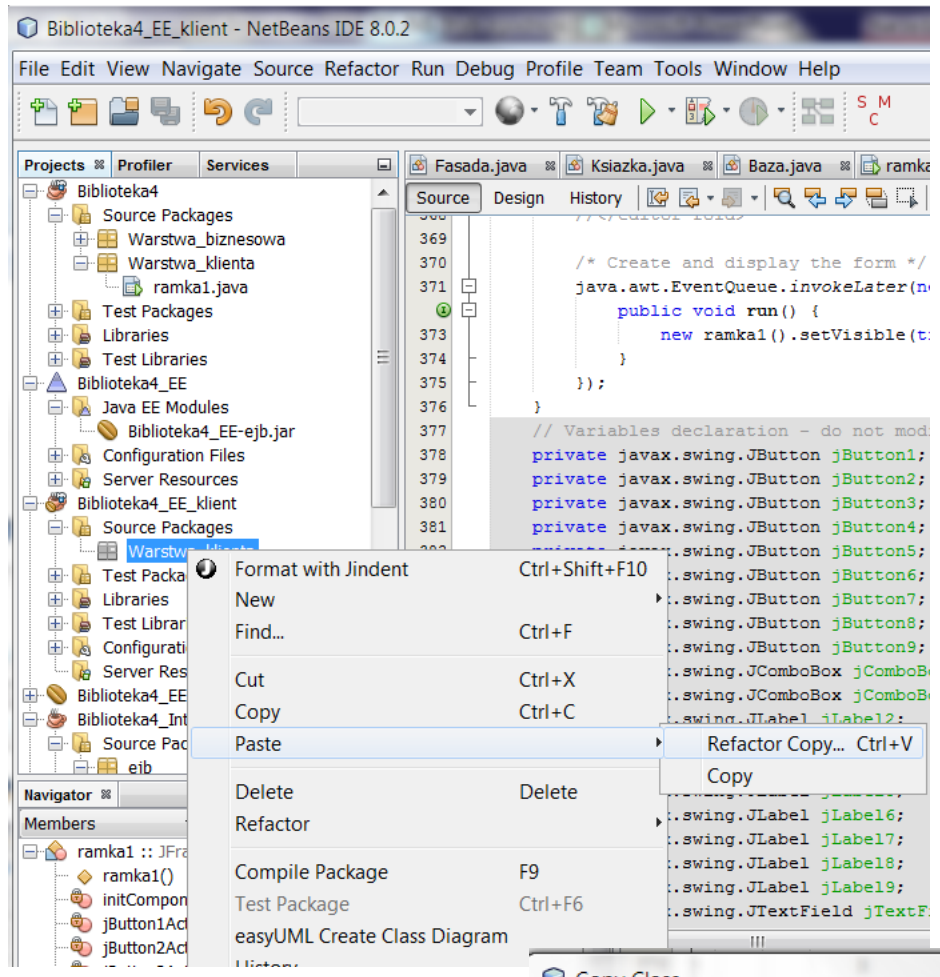
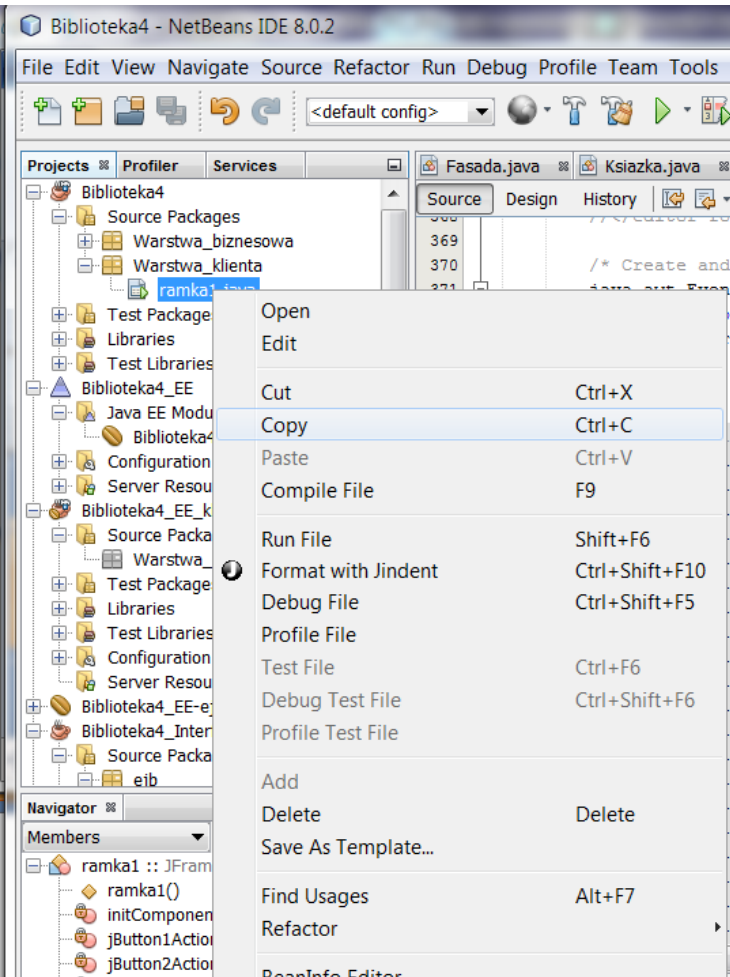
Definicja klasy o nazwie pakietowej takiej samej, jaką w projekcie SE posiada klasa reprezentująca GUI



4.18. Bezpieczne usunięcie wykonanej klasy o nazwie pakietowej takiej samej, jak w projekcie SE posiada klasa reprezentująca GUI -cd



4.19. Skopiowanie klasy ramka1 do obsługi GUI z projektu SE (Biblioteka4) do projektu Biblioteka4_EE_klient



4.20. Przygotowanie klasy **ramka1** do wstawienia adnotacji typu EJB do obiektu klasy **Fasadaejb**, umożliwiającej zdalny dostęp do logiki biznesowej i bazy danych

```
+ ...4 lines
package Warstwa_klienta;

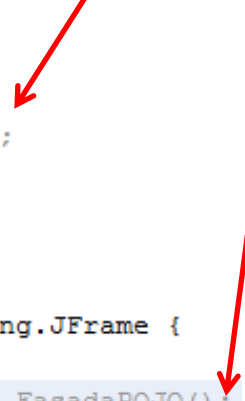
//import Warstwa_biznesowa.FasadaPOJO;
- import java.util.ArrayList;
  import javax.swing.JComboBox;

+ /**...4 lines */
public class ramka1 extends javax.swing.JFrame {

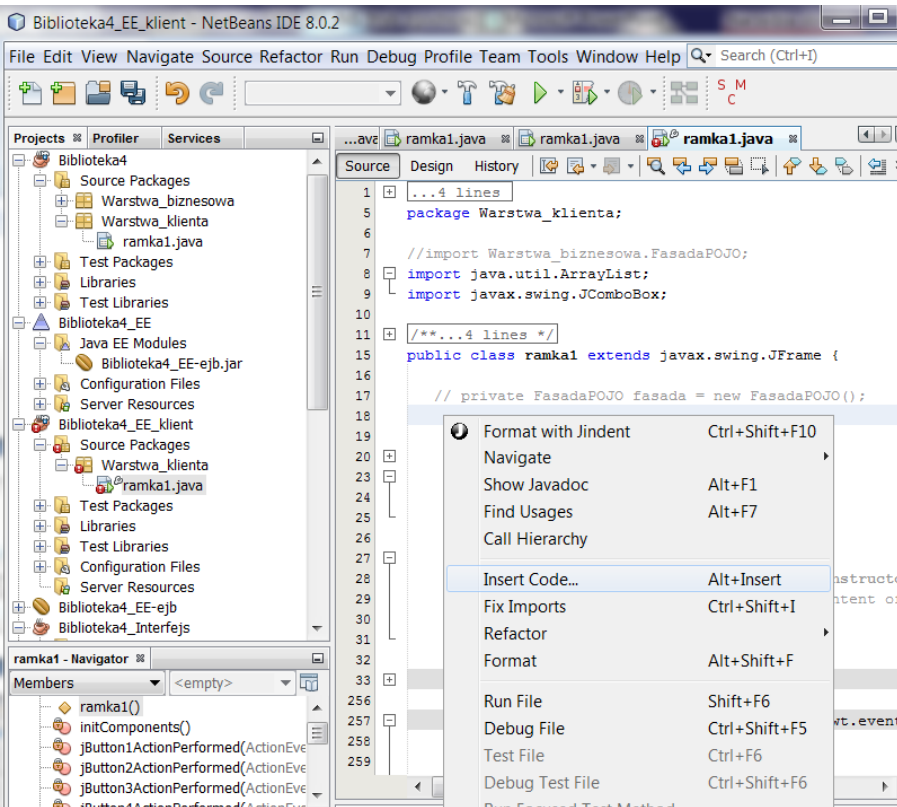
    // private FasadaPOJO fasada = new FasadaPOJO();

    + /** Creates new form ramka1 ...3 lines */
    - public ramka1() {
      initComponents();
    }

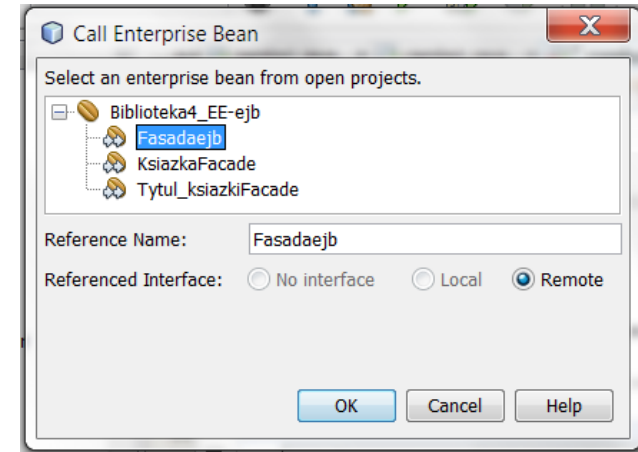
    - /**
      * This method is called from within the constr
      * WARNING: Do NOT modify this code. The content
      * regenerated by the Form Editor.
      */
      @SuppressWarnings("unchecked")
    + Generated Code
```



4.20. Wstawienie adnotacji do obiektu zdalnego typu Fasadaejb w klasie ramka1 cd



- Generate
- Constructor...
- Logger...
- Getter...
- Setter...
- Getter and Setter...
- equals() and hashCode()...
- toString()...
- Delegate Method...
- Override Method...
- Add Property...
- Call Enterprise Bean...
- Use Database...
- Send JMS Message...
- Send E-mail...
- Call Web Service Operation...
- Generate REST Client...



```
package Warstwa_klienta;

//import Warstwa_biznesowa.FasadaPOJO;
import ejb.FasadaejbRemote;
import java.util.ArrayList;
import javax.ejb.EJB;
import javax.swing.JComboBox;

/**...4 lines */
public class ramka1 extends javax.swing.JFrame {
    @EJB
    private static FasadaejbRemote fasada;

    // private FasadaPOJO fasada = new FasadaPOJO();
}
```


4.21. Usunięcie komentarzy w metodach obsługujących dostęp do bazy danych: `jButton5ActionPerformed`, `jButton6ActionPerformed`, `jButton7ActionPerformed`, `jButton8ActionPerformed`

```
package Warstwa_klienta;
//import Warstwa_biznesowa.FasadaPOJO;
import ejb.FasadaejbRemote;
import java.util.ArrayList;
import javax.ejb.EJB;
import javax.swing.JComboBox;

/**...4 lines */
public class ramka1 extends javax.swing.JFrame {
    @EJB
    private static FasadaejbRemote fasada;
    // private FasadaPOJO fasada = new FasadaPOJO();

    /** Creates new form ramka1 ...3 lines */
    public ramka1() {
        initComponents();
    }

    /** This method is called from within the constructor to initialize the form ...5 lines */
    @SuppressWarnings("unchecked")
    Generated Code

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String s1, s2, s3, s4, s5;
        Object zrodlo = evt.getSource();
        if (zrodlo == jButton1) {
            s1 = jTextField1.getText();
            s2 = jTextField2.getText();
            s3 = jTextField4.getText();
            s4 = jTextField5.getText();
            s5 = jTextField6.getText();
            String[] tytul = {s1, s2, s3, s4, s5};
            if (!s1.equals("") && !s2.equals("") && !s3.equals("")
                && !s4.equals("") && !s5.equals("")) {
                fasada.dodaj_tytul(tytul);
            }
        }
    }
}
```

4.21. Usunięcie komentarzy w metodach obsługujących dostęp do bazy danych: jButton5ActionPerformed, jButton6ActionPerformed, jButton7ActionPerformed, jButton8ActionPerformed cd

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    String s1, s2;  
    Object zrodlo = evt.getSource();  
    if (zrodlo == jButton2) {  
        s1 = jTextField5.getText();  
        s2 = jTextField7.getText();  
        String[] ksiazka = {s1, s2};  
        if (!s1.equals("") && !s2.equals("")) {  
            fasada.dodaj_ksiazke(ksiazka);  
        }  
    }  
}  
  
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    zawartosc_listy(fasada.tytuly(), jComboBox1);  
}  
  
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    zawartosc_listy(fasada.ksiazki(), jComboBox2);  
}  
  
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        fasada.dodaj_tytuly();  
    } catch (Exception e) {  
        System.out.println(e);  
    }  
}  
  
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        fasada.dodaj_ksiazki();  
    } catch (Exception e) {  
        System.out.println(e);  
    }  
}  
  
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        zawartosc_listy(fasada.tytuly_(), jComboBox1);  
    } catch (Exception e) {  
        System.out.println(e);  
    }  
}  
}
```

4.21. Usunięcie komentarzy w metodach obsługujących dostęp do bazy danych: `jButton5ActionPerformed`, `jButton6ActionPerformed`, `jButton7ActionPerformed`, `jButton8ActionPerformed` cd

```
private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        zawartosc_listy(fasada.ksiazki_(), jComboBox2);
    } catch (Exception e) {
        System.out.println(e);
    }
}

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
    jButton7ActionPerformed(evt);
    jButton8ActionPerformed(evt);
}

private void zawartosc_listy(ArrayList<String> kol, JComboBox lista) {
    lista.removeAllItems();
    for (String s : kol) {
        lista.addItem(s);
    }
}

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)
    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new ramka1().setVisible(true);
        }
    });
}
```

```
// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JButton jButton6;
private javax.swing.JButton jButton7;
private javax.swing.JButton jButton8;
private javax.swing.JButton jButton9;
private javax.swing.JComboBox jComboBox1;
private javax.swing.JComboBox jComboBox2;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField4;
private javax.swing.JTextField jTextField5;
private javax.swing.JTextField jTextField6;
private javax.swing.JTextField jTextField7;
// End of variables declaration
```

```
}
```

5. Sposób uruchomienia aplikacji EE

1. Należy kolejno wykonać Clean and Build projektu **Biblioteka4_Library** (biblioteka klas biznesowych)
2. Należy kolejno wykonać Clean and Build projektu **Biblioteka4_Interfejs** (interfejs zawierający deklaracje metod o zdalnym dostępie)
3. Należy kolejno wykonać Clean and Build projektu **Biblioteka4_EE-ejb** (moduł EJB)
4. Należy kolejno wykonać Clean and Build projektu **Biblioteka4_EE_klient** (aplikacja kliencka)
5. Należy kolejno wykonać Clean and Build projektu **Biblioteka4_EE** (aplikacja główna EE)
6. Należy kolejno wykonać **Deploy** projektu **Biblioteka4_EE** (aplikacja główna EE)
7. Należy uruchomić za pomocą **Run** kilka egzemplarzy aplikacji klienckiej **Biblioteka4_EE_klient**
8. W przypadku dokonywanych zmian w kodzie należy zatrzymać aplikacje EE (główną i kliencką) za pomocą **Undeploy** wg p. 8, a po zakończeniu zmian powtórzyć kroki 1-7. W przypadku zakończenia poprawek wystarczy uruchomić aplikację za pomocą p.6-7.

6. Projekt GUI warstwy klienta (program II z lab. 4) – dodano przyciski: **Tytuły do bazy** (zapis tytułów do bazy danych), **Książki do bazy** (zapis książek do bazy), **Tytuły z bazy** (odczyt tytułów z bazy danych i wyświetlenie w liście – Tytuły książek), **Książki z bazy** (odczyt książek z bazy danych i wyświetlenie w liście – Książki), **Dane z bazy** (wykonuje czynności przycisków **Tytuły z bazy** oraz **Książki z bazy**)

The screenshot shows a GUI for a book database application. It consists of the following elements:

- Input Fields:** Six text input fields for entering book details: "Tytuł książki", "Nazwisko autora książki", "Imię autora książki", "ISBN tytułu książki", "Wydawnictwo książki", and "Numer książki".
- Action Buttons:** A grid of buttons for performing database operations:
 - Top row: "Zapisz tytuł", "Zapisz książkę", "Wyświetl tytuły", "Wyświetl książki".
 - Bottom row: "Tytuły do bazy", "Książki do bazy", "Tytuły z bazy", "Książki z bazy", "Dane z bazy".
- Data Display:** Two dropdown menus at the bottom for displaying data:
 - "Tytuły książek" with a dropdown menu showing "Item 1".
 - "Książki" with a dropdown menu showing "Item 1".

6.1. Wyświetlenie tytułów z aplikacji

The screenshot shows a window with the following elements:

- Input fields for: Tytuł książki, Nazwisko autora książki, Imię autora książki, ISBN tytułu książki, Wydawnictwo książki, Numer książki.
- Buttons: Zapisz tytuł, Zapisz książkę, Wyświetl tytuły, Wyświetl książki, Tytuły do bazy, Książki do bazy, Tytuły z bazy, Książki z bazy, Dane z bazy.
- A dropdown menu for "Tytuły książek" with the following items:
 - Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1
 - Tytuł: 2 Autor: 2 2 ISBN: 22 Wydawnictwo: 2
 - Tytuł: 3 Autor: 3 3 ISBN: 33 Wydawnictwo: 3
 - Tytuł: 4 Autor: 4 4 ISBN: 44 Wydawnictwo: 4
 - Tytuł: 5 Autor: 5 5 ISBN: 55 Wydawnictwo: 5

6.2. Wyświetlenie książek z aplikacji

The screenshot shows a software application window with a title bar containing standard Windows window controls (minimize, maximize, close). The main area of the window is a form for managing books. It features several text input fields for book details: 'Tytuł książki', 'Nazwisko autora książki', 'Imię autora książki', 'ISBN tytułu książki', 'Wydawnictwo książki', and 'Numer książki'. Below these fields is a grid of buttons for saving and displaying data: 'Zapisz tytuł', 'Zapisz książkę', 'Wyświetl tytuły', 'Wyświetl książki', 'Tytuły do bazy', 'Książki do bazy', 'Tytuły z bazy', 'Książki z bazy', and 'Dane z bazy'. At the bottom, there are two dropdown menus. The first, labeled 'Tytuły książek', shows a selected item: 'Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1'. The second dropdown, labeled 'Książki', is open, displaying a list of items with their full details: 'Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1 Numer: 1' through 'Tytuł: 5 Autor: 5 5 ISBN: 55 Wydawnictwo: 5 Numer: 1'. The first item in this list is highlighted in blue.

Tytuł książki

Nazwisko autora książki

Imię autora książki

ISBN tytułu książki

Wydawnictwo książki

Numer książki

Zapisz tytuł Zapisz książkę Wyświetl tytuły Wyświetl książki

Tytuły do bazy Książki do bazy Tytuły z bazy Książki z bazy Dane z bazy

Tytuły książek Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1 ▼

Książki Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1 Numer: 1 ▼

- Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1 Numer: 1
- Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1 Numer: 2
- Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1 Numer: 3
- Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1 Numer: 4
- Tytuł: 2 Autor: 2 2 ISBN: 22 Wydawnictwo: 2 Numer: 2
- Tytuł: 3 Autor: 3 3 ISBN: 33 Wydawnictwo: 3 Numer: 1
- Tytuł: 4 Autor: 4 4 ISBN: 44 Wydawnictwo: 4 Numer: 1
- Tytuł: 5 Autor: 5 5 ISBN: 55 Wydawnictwo: 5 Numer: 1

6.3. Wyświetlenie tytułów z bazy danych

The screenshot shows a software application window with a light blue border and standard Windows window controls (minimize, maximize, close) in the top right corner. The main area is divided into several sections:

- Input Fields:** Six text boxes for entering book information, each with a label to its left:
 - Tytuł książki
 - Nazwisko autora książki
 - Imię autora książki
 - ISBN tytułu książki
 - Wydawnictwo książki
 - Numer książki
- Action Buttons:** A grid of buttons for performing operations:
 - Row 1: Zapisz tytuł, Zapisz książkę, Wyświetl tytuły, Wyświetl książki
 - Row 2: Tytuły do bazy, Książki do bazy, Tytuły z bazy, Książki z bazy, Dane z bazy
- Data Display:** A dropdown menu is open, showing a list of books. The list has two columns: 'Tytuły książek' and 'Książki'. The data in the list is as follows:

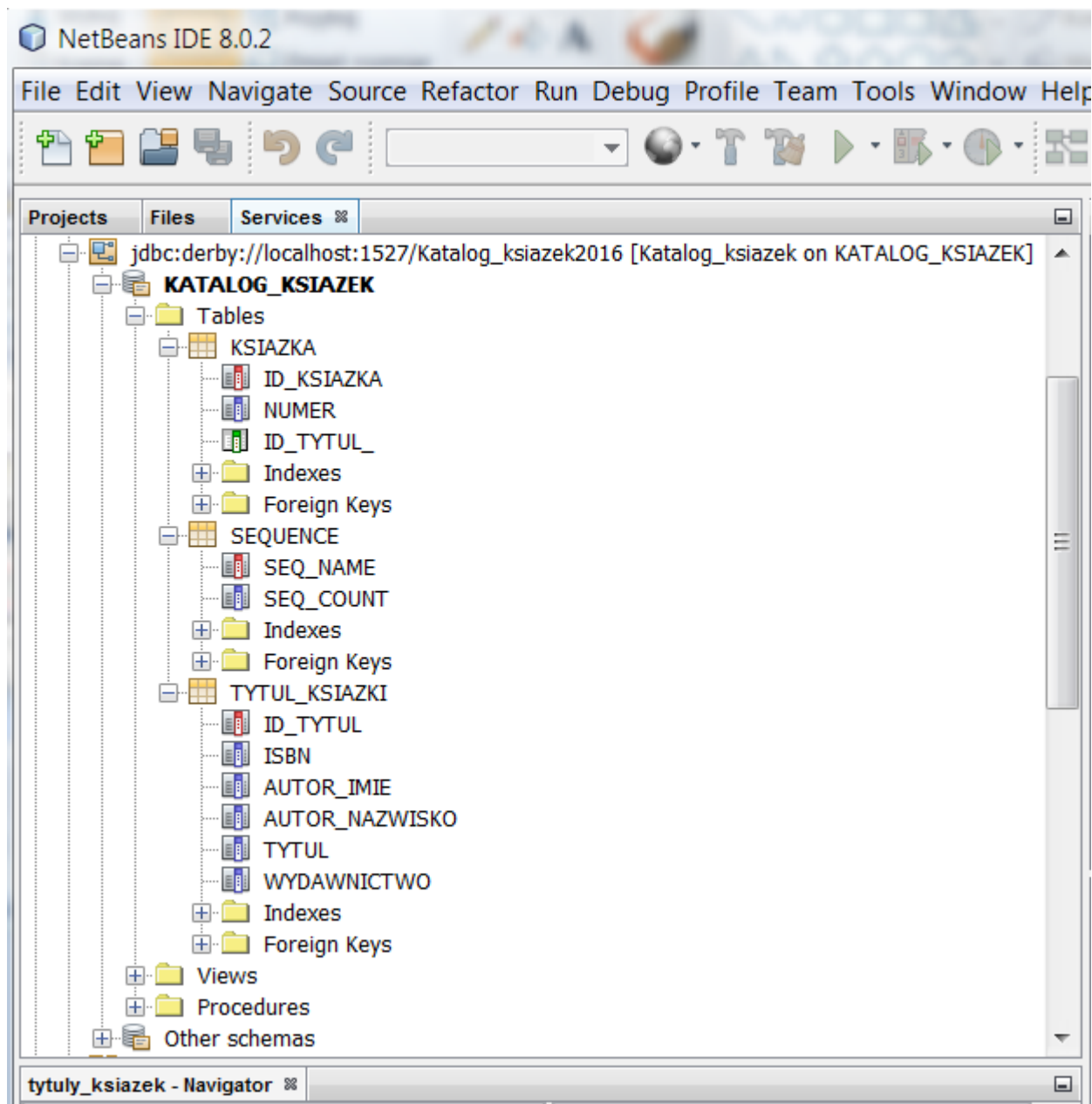
Tytuły książek	Książki
Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1	Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1
Tytuł: 2 Autor: 2 2 ISBN: 22 Wydawnictwo: 2	Tytuł: 2 Autor: 2 2 ISBN: 22 Wydawnictwo: 2
Tytuł: 3 Autor: 3 3 ISBN: 33 Wydawnictwo: 3	Tytuł: 3 Autor: 3 3 ISBN: 33 Wydawnictwo: 3
Tytuł: 4 Autor: 4 4 ISBN: 44 Wydawnictwo: 4	Tytuł: 4 Autor: 4 4 ISBN: 44 Wydawnictwo: 4
Tytuł: 5 Autor: 5 5 ISBN: 55 Wydawnictwo: 5	Tytuł: 5 Autor: 5 5 ISBN: 55 Wydawnictwo: 5

6.4. Wyświetlenie książek z bazy danych

The screenshot shows a software window with a title bar and standard window controls. The main area contains several input fields and buttons. The input fields are labeled: 'Tytuł książki', 'Nazwisko autora książki', 'Imię autora książki', 'ISBN tytułu książki', 'Wydawnictwo książki', and 'Numer książki'. Below these fields are two rows of buttons. The first row contains 'Zapisz tytuł', 'Zapisz książkę', 'Wyświetl tytuły', and 'Wyświetl książki'. The second row contains 'Tytuły do bazy', 'Książki do bazy', 'Tytuły z bazy', 'Książki z bazy', and 'Dane z bazy'. Below the buttons, there are two dropdown menus. The first is labeled 'Tytuły książek' and shows 'Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1'. The second is labeled 'Książki' and shows 'Tytuł: 1 Autor: 1 1 ISBN: 11 Wydawnictwo: 1 Numer: 1'. A list of book records is displayed below the 'Książki' dropdown, with the first record highlighted in blue. The records are as follows:

Tytuł	Autor	ISBN	Wydawnictwo	Numer
Tytuł: 1	Autor: 1 1	ISBN: 11	Wydawnictwo: 1	Numer: 1
Tytuł: 1	Autor: 1 1	ISBN: 11	Wydawnictwo: 1	Numer: 2
Tytuł: 2	Autor: 2 2	ISBN: 22	Wydawnictwo: 2	Numer: 2
Tytuł: 1	Autor: 1 1	ISBN: 11	Wydawnictwo: 1	Numer: 3
Tytuł: 1	Autor: 1 1	ISBN: 11	Wydawnictwo: 1	Numer: 4
Tytuł: 3	Autor: 3 3	ISBN: 33	Wydawnictwo: 3	Numer: 1
Tytuł: 4	Autor: 4 4	ISBN: 44	Wydawnictwo: 4	Numer: 1
Tytuł: 5	Autor: 5 5	ISBN: 55	Wydawnictwo: 5	Numer: 1

7. Widok struktury bazy danych, w której automatycznie zostały utworzone tabele



Projects Files Services

jdbc:derby://localhost:1527/Katalog_ksiazek2016 [Katalog_ksiazek2016]

KATALOG_KSIAZEK

- Tables
 - KSIAZKA
 - ID_KSIAZKA
 - NUMER
 - ID_TYTUL_
 - Indexes
 - Foreign Keys
 - SEQUENCE
 - SEQ_NAME
 - SEQ_COUNT
 - Indexes
 - Foreign Keys
 - TYTUL_KSIAZKI
 - ID_TYTUL
 - ISBN
 - AUTOR_IMIE
 - AUTOR_NAZWISKO
 - TYTUL
 - WYDAWNICTWO
 - Indexes
 - Foreign Keys
- Views
- Procedures
- Other schemas

jdbc:derby://localhost:1527/Dostawca [Dostawca o...]

jdbc:derby://localhost:1527/katalog11 [katalog1 on...]

jdbc:derby://localhost:1527/Katalog_ksiazek [Katalo...]

jdbc:derby://localhost:1527/Katalog_ksiazek2016 [Katalog_ksiazek2016]

KATALOG_KSIAZEK

- Tables
 - KSIAZKA
 - ID_KSIAZKA
 - NUMER
 - ID_TYTUL_
 - Indexes
 - Foreign Keys
 - SEQUENCE
 - SEQ_NAME
 - SEQ_COUNT
 - Indexes
 - Foreign Keys
 - TYTUL_KSIAZKI
 - ID_TYTUL
 - ISBN
 - AUTOR_IMIE
 - AUTOR_NAZWISKO
 - TYTUL
 - WYDAWNICTWO
 - Indexes
 - Foreign Keys

...ave FasadaPOJO.java SQL 10 [jdbc:derby://localhost:15...]

Connection: jdbc:derby://localhost:1527/Katalog_ksiazek2016 [Katalog_ksiazek...]

```
1 select * from KATALOG_KSIAZEK.TYTUL_KSIAZKI;
```

select * from KATALOG_KSI... *

Page Size: 20 Total Rows: 5 Page: 1 of 1 Matching Row

#	ID_TYTUL	ISBN	AUTOR_IMIE	AUTOR_NAZWISKO	TYTUL	WYDAWNICTWO
1		210 44	4	4	4	4
2		212 55	5	5	5	5
3		208 33	3	3	3	3
4		201 11	1	1	1	1
5		206 22	2	2	2	2

Output

Connection: jdbc:derby://localhost:1527/Katalog_ksiazek2016 [Katalog_ksiazek...]

```
1 select * from KATALOG_KSIAZEK.KSIAZKA;
```

select * from KATALOG_KSI... *

Page Size: 20 Total Rows: 8 Page: 1 of 1 M

#	ID_KSIAZKA	NUMER	ID_TYTUL_	
1		211	1	210
2		207	2	206
3		204	3	201
4		209	1	208
5		205	4	201
6		203	2	201
7		213	1	212
8		202	1	201

Output

7.1. Zawartość tabel TYTUL_KSIAZKI i KSIAZKA w wyniku działania aplikacji

7.2. Zawartość tabeli SEQUENCE wspierającej działanie narzędzia ORM (EclipseLink) w zakresie generowania kluczy głównych w wyniku działania aplikacji

The screenshot shows a database management tool interface. On the left, a tree view displays the database schema for 'KATALOG_KSIAZEK'. The 'SEQUENCE' table is highlighted, showing columns 'SEQ_NAME' and 'SEQ_COUNT'. The main window displays a SQL query: 'select * from KATALOG_KSIAZEK."SEQUENCE";'. Below the query, the results are shown in a table with the following data:

#	SEQ_NAME	SEQ_COUNT
1	SEQ_GEN	250

8. W zakładce **Services** należy zatrzymać aplikacje EE za pomocą mechanizmu **Undeploy**

