Projektowanie oprogramowania

Warstwa integracji z bazą danych oparta na technologii ORM

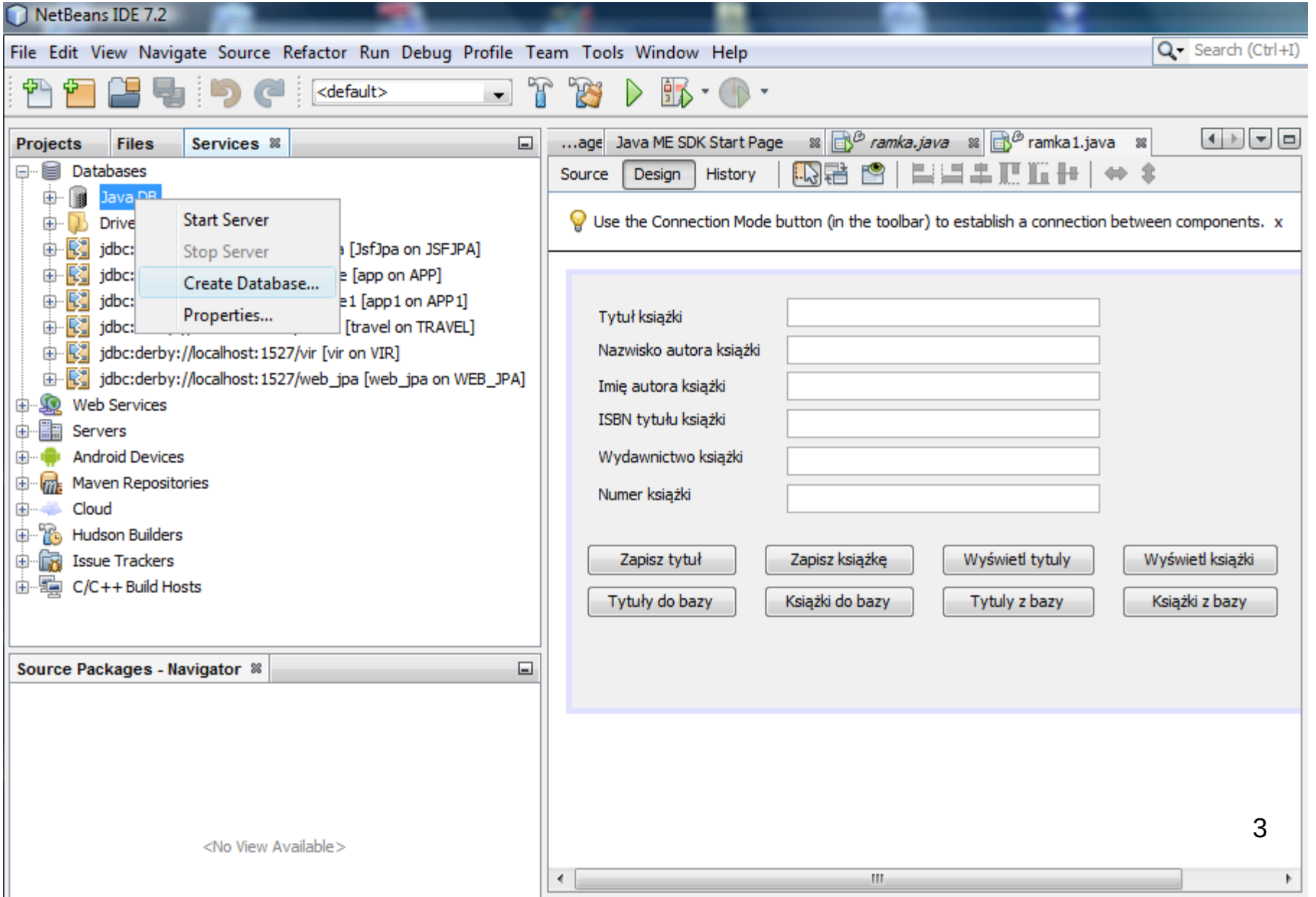
Platforma Java EE

Autor: Zofia Kruczkiewicz

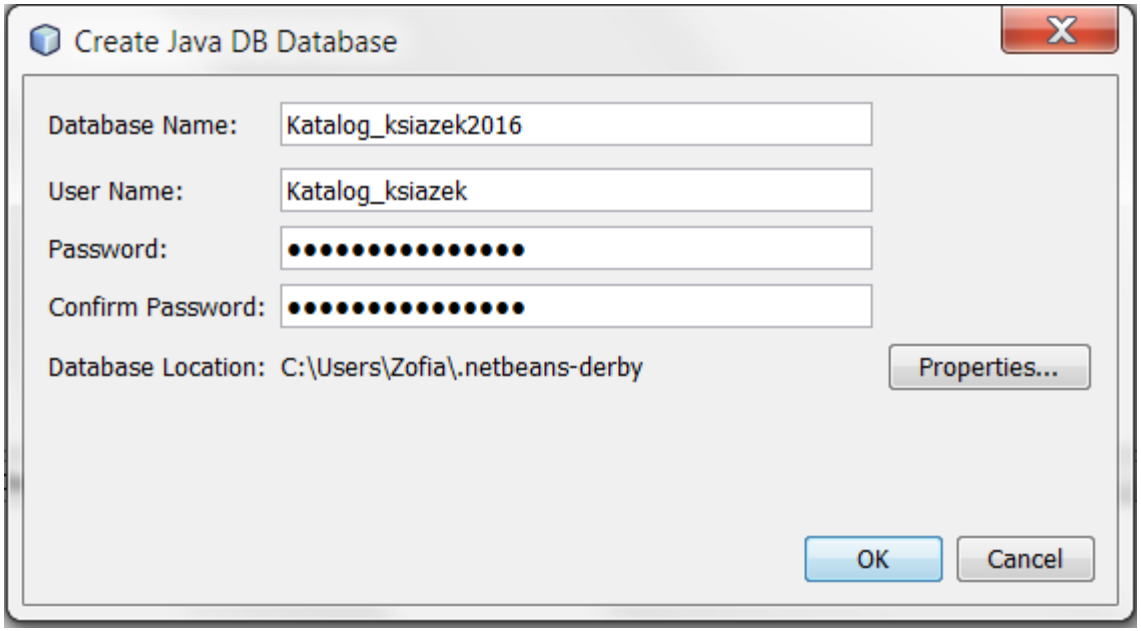
Wykonanie czterowarstwowej aplikacji EE z dostępem do bazy danych, opartym na wzorcu Domain Store

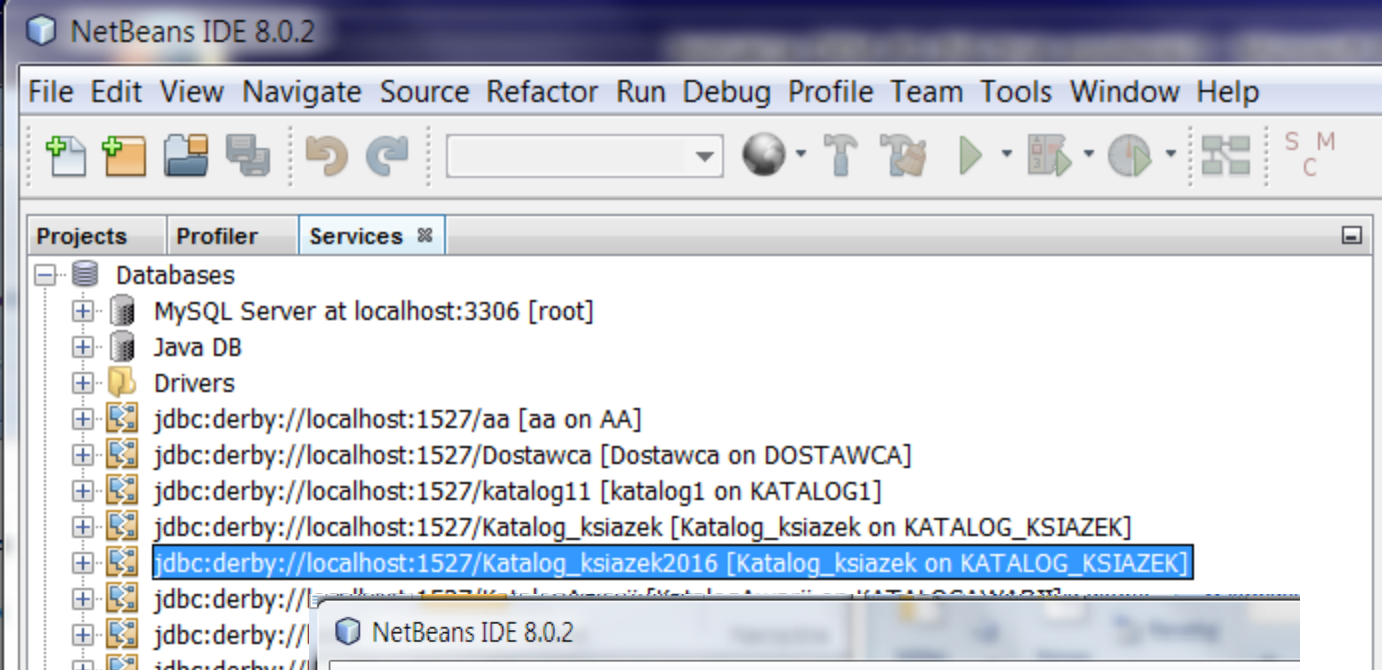
(http://corej2eepatterns.com/DomainStore.htm)

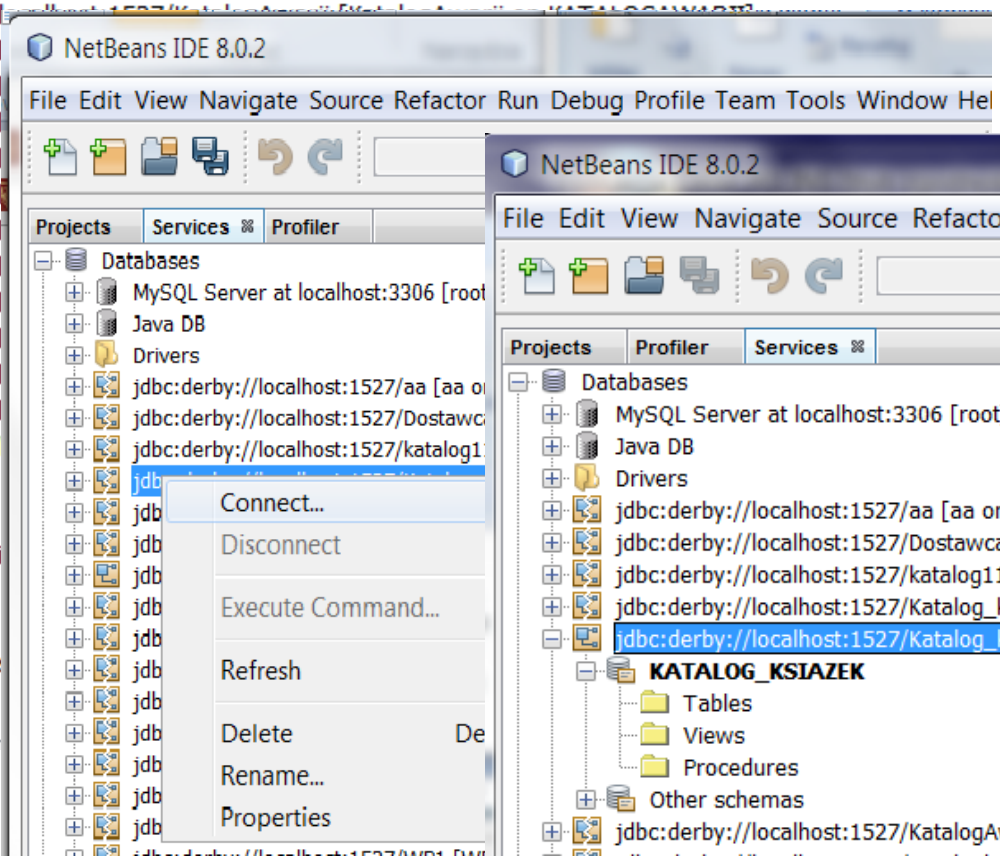
1. Utworzenie pustej bazy danych – w okienku Services należy prawym klawiszem myszy wybrać **Databases\Java DB\Create Database**

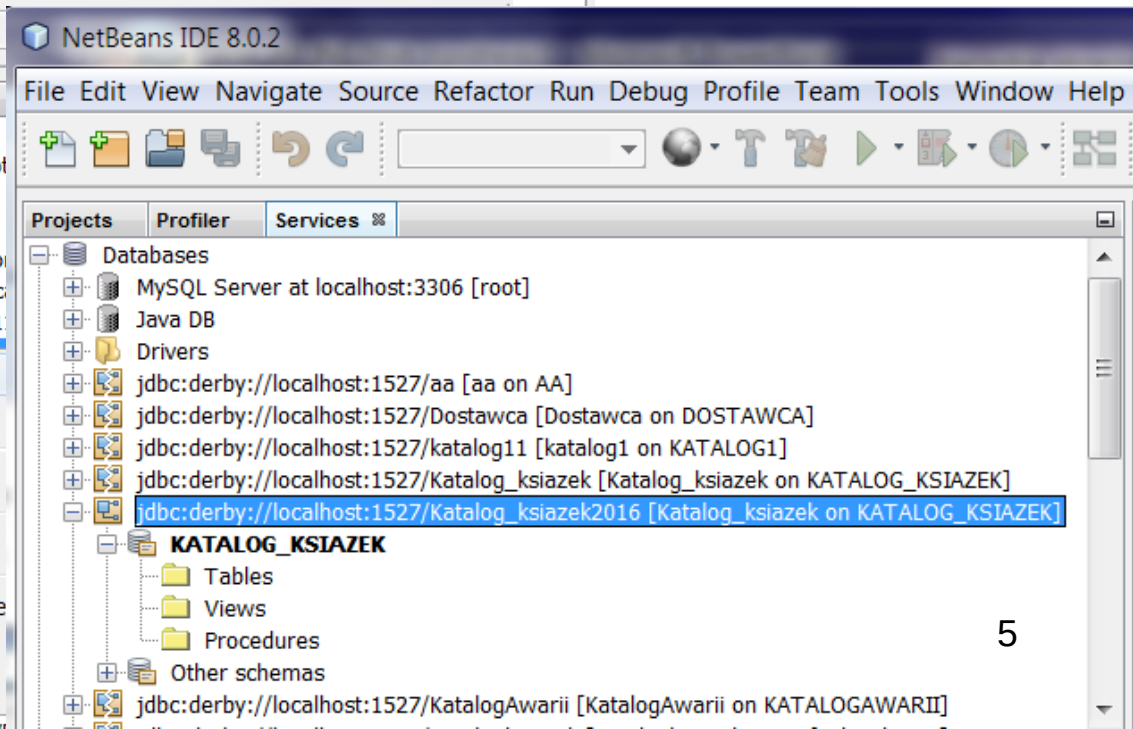


1.1. Wykonanie pustej bazy danych Katalog_książek2016
login: Katalog_ksiazek
haslo: Katalog_ksiazek





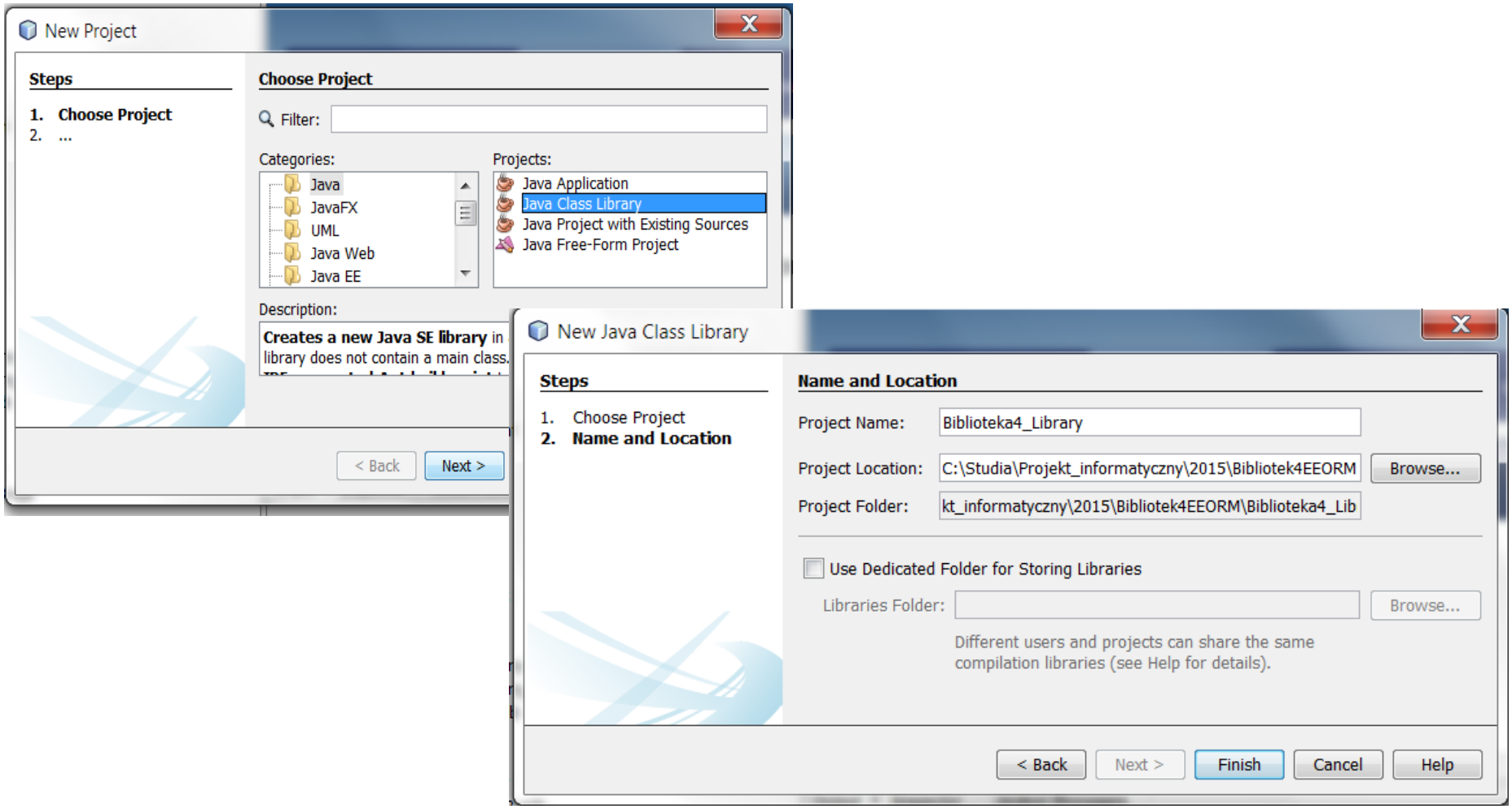




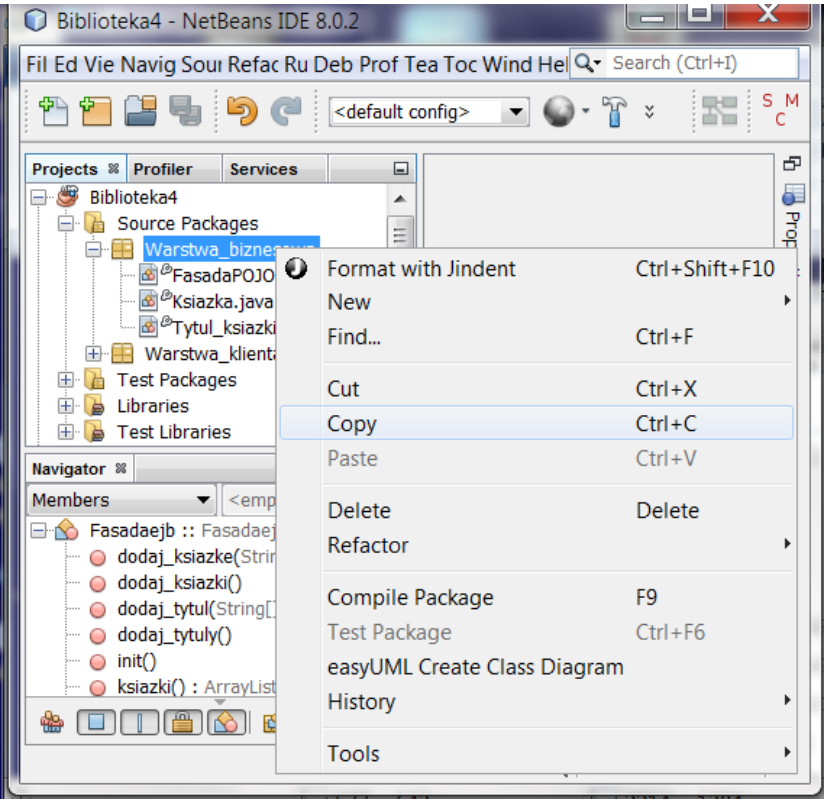
1.2. Połączenie z pustą bazą danych o nazwie Katalog_ksiazek2016

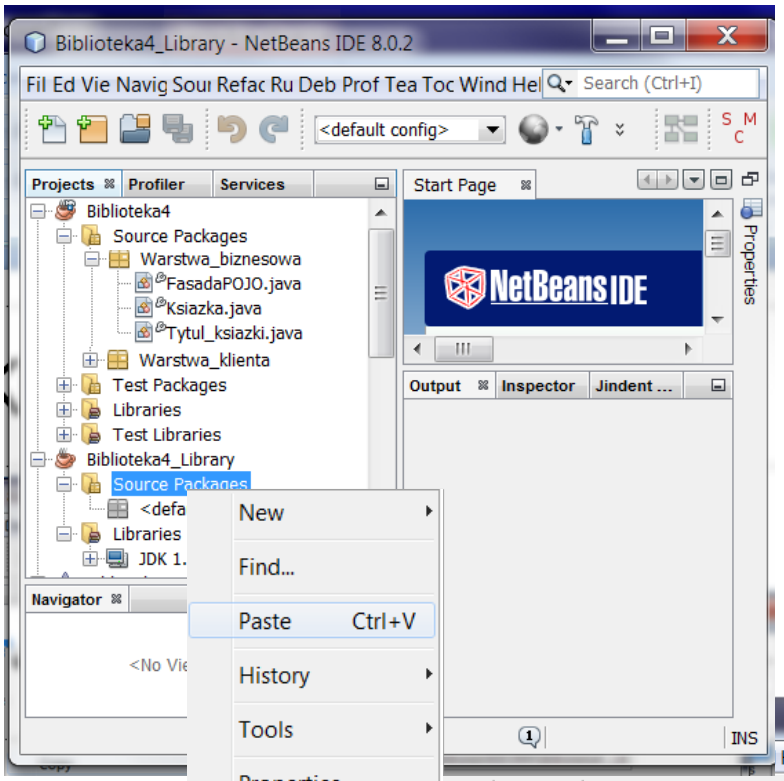
2. Wykonanie adnotacji w obiektowym modelu danych

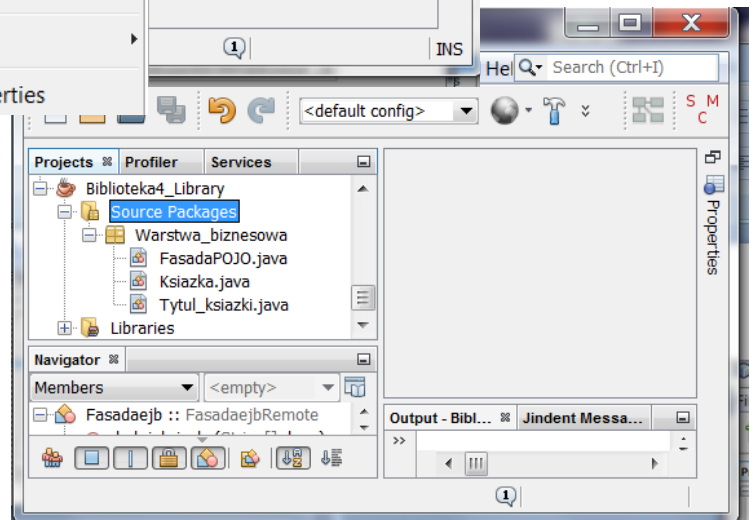
2.1. Należy wykonać projekt typu **Java Class Library** np. o nazwie **Biblioteka4_Library** (**File/New Project/Java/Java Class Library** i nacisnąć klawisz **Next)**. Następnie w polu **Project Name** wpisać nazwę projektu po wyborze katalogu w polu **Project Location** i nacisnąć klawisz **Finish**.



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



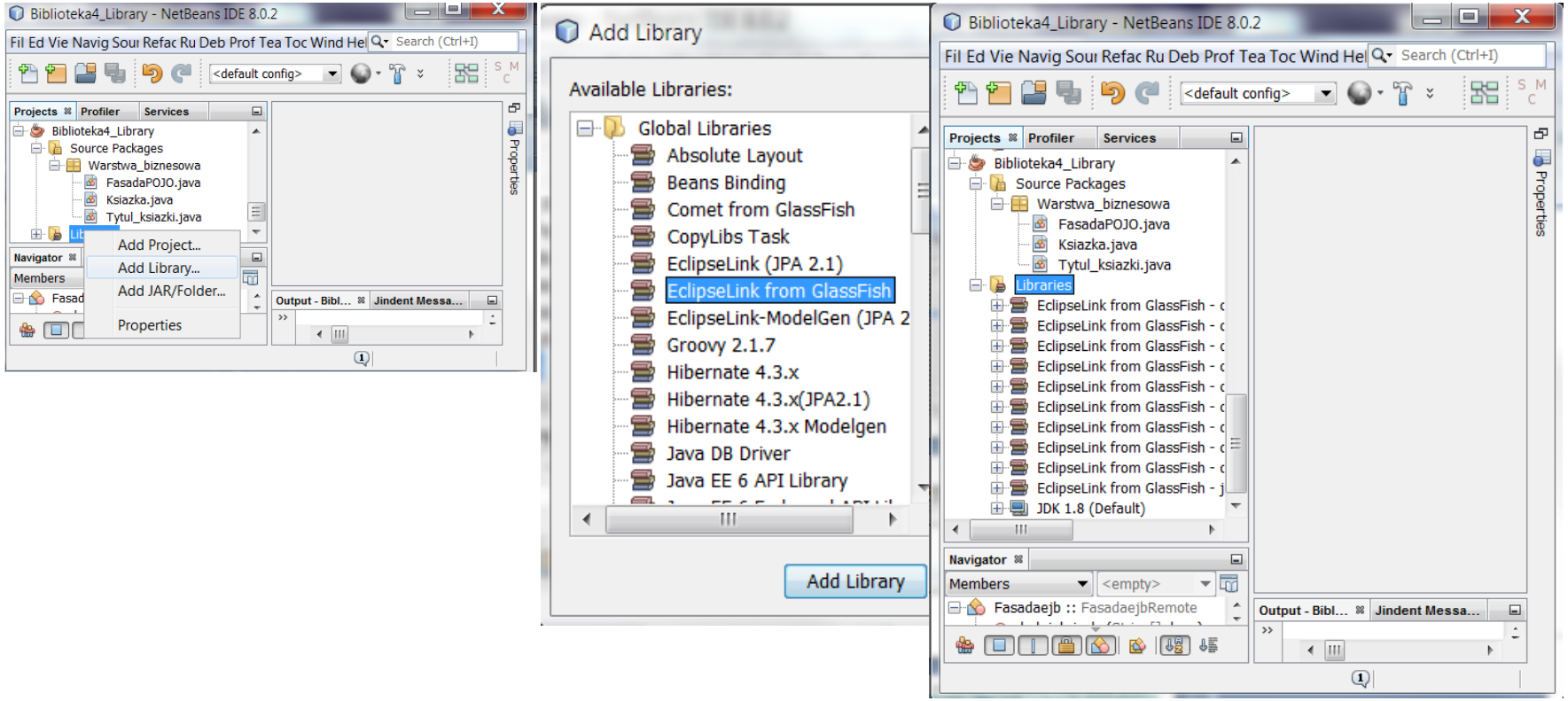




2.3. Wykonanie adnotacji w obiektowym modelu danych

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/javaee/7/tutorial/persistence-intro.htm#BNBPZ



2.3.3. Na kolejnych pokazano, jak przekształcić klasy Tytul_ksiazki oraz Ksiazka na klasy typu Entity

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

```
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 technologią ORM (JPA) - cd

```
public Tytul_ksiazki()
    {     }
public Tytul_ksiazki(int i)
    { mKsiazka=new ArrayList();
      idTytul=null;
    }
public Tytul_ksiazki(Long idTytul)              { this.idTytul = idTytul;   }
public Long getIdTytul()                        { return idTytul; }
public void setIdTytul(Long idTytul)            { this.idTytul = idTytul; }
public String getTytul()                        { return tytul; }
public void setTytul(String tytul)              { this.tytul = tytul; }
public String getNazwisko()                     { return nazwisko;  }
public void setNazwisko(String autorNazwisko)   { this.nazwisko = autorNazwisko;   }
public String getImie()                         {  return imie; }
public void setImie(String autorImie)           {  this.imie = autorImie; }
public String getISBN()                         { return ISBN;  }
public void setISBN(String isbn)                { this.ISBN = isbn;  }
public String getWydawnictwo()                  {  return wydawnictwo;  }
public void setWydawnictwo(String wydawnictwo) {   this.wydawnictwo = wydawnictwo;   }
public Collection<Ksiazka> getMKsiazka()
    {  return mKsiazka; }
public void setMKsiazka(Collection<Ksiazka> ksiazkaCollection)
    {  this.mKsiazka = ksiazkaCollection;  }
```

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

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_ksiazki) ob).getISBN();
    return ISBN.equals(isbn2); }
```

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

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

    public void addKsiazka(Ksiazka nowa) {
        if (!this.mKsiazka.contains(nowa)) {
            this.mKsiazka.add(nowa);
            nowa.setTytul_ksiazki(this);
        }
    }

    public ArrayList<String> ksiazki() {
        ArrayList<String> ksiazki = new ArrayList();
        Iterator<Ksiazka> it = mKsiazka.iterator();
        while (it.hasNext()) {
            ksiazki.add(it.next().toString());
        }
        return ksiazki;
    }
}
```

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; }
```

Metoda hashCode() powinna być dodana do dodaniu adnotacji

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

    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_ksiazki> tytuly_ksiazek;

    public FasadaPOJO() {
      tytuly_ksiazek = new ArrayList();
    }
    public ArrayList<Tytul_ksiazki> getTytuly_ksiazek() {
        return tytuly_ksiazek;
    }
    public void setTytuly_ksiazek(ArrayList<Tytul_ksiazki> val) {
        this.tytuly_ksiazek = val;
    }
    public void dodaj_tytul(String dane_tytul[]) {
        Tytul_ksiazki tytul_ksiazki = new Tytul_ksiazki(1);
        tytul_ksiazki.setTytul(dane_tytul[0]);
        tytul_ksiazki.setNazwisko(dane_tytul[1]);
        tytul_ksiazki.setImie(dane_tytul[2]);
        tytul_ksiazki.setISBN(dane_tytul[3]);
        tytul_ksiazki.setWydawnictwo(dane_tytul[4]);
        addtytul_ksiazki(tytul_ksiazki);
    }
    public void addtytul_ksiazki(Tytul_ksiazki val) {
        boolean czy_jest = tytuly_ksiazek.contains(val);
        if (!czy_jest) {
            tytuly_ksiazek.add(val);
        }
    }
    public void dodaj_ksiazke(String dane[]) // your code here
    {
        Tytul_ksiazki pom = new Tytul_ksiazki();
        pom.setISBN(dane[0]);
        int idx = tytuly_ksiazek.indexOf(pom);
        if (idx != -1) {
            Tytul_ksiazki pom1 = tytuly_ksiazek.get(idx);
            pom1.dodaj_ksiazke(dane);
        }
    }
```

```
    public ArrayList<String> tytuly() {
      ArrayList<String> tytuly = new ArrayList<String>();
      Iterator<Tytul_ksiazki> it = tytuly_ksiazek.iterator();
      while (it.hasNext()) {
          tytuly.add(it.next().toString());
      }
      return tytuly;
    }
    public ArrayList<String> ksiazki() {
      ArrayList<String> ksiazki = new ArrayList<String>();
      Iterator<Tytul_ksiazki> it = tytuly_ksiazek.iterator();
      while (it.hasNext()) {
          ksiazki.addAll(it.next().ksiazki());
      }
      return ksiazki;
    }
    public void uaktualnij_dane(Tytul_ksiazki tytuly[], Ksiazka ksiazki[]) {
      tytuly_ksiazek.clear();
      for (int i = 0; i < tytuly.length; i++) {
          tytuly_ksiazek.add(tytuly[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

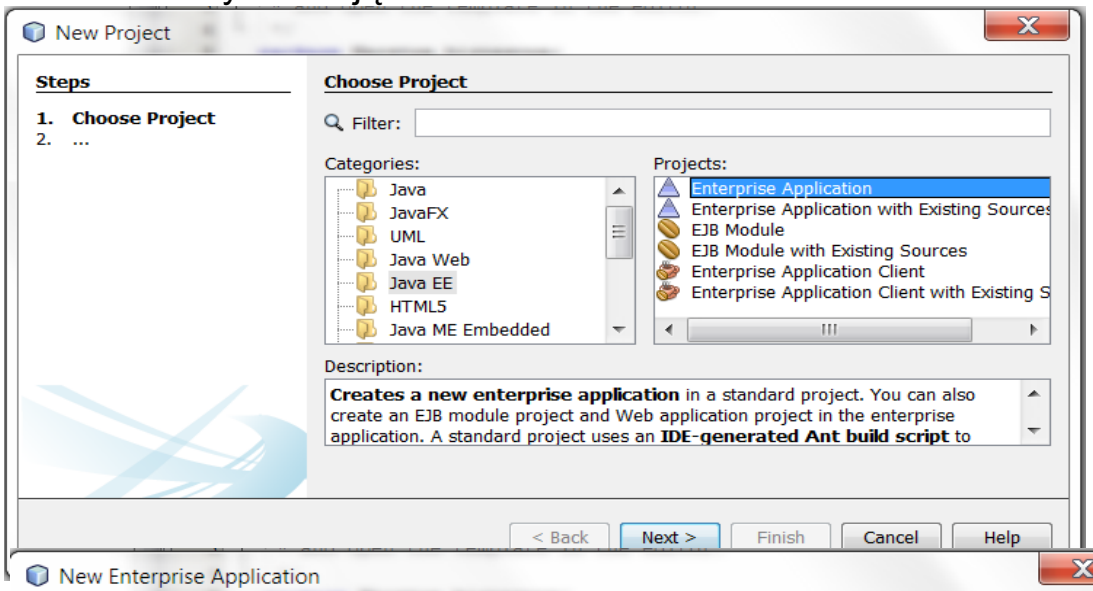
@OneToMany(mappedBy = "mTytul_ksiazki",
cascade = CascadeType.ALL)
private Collection<Ksiazka> mKsiazka;

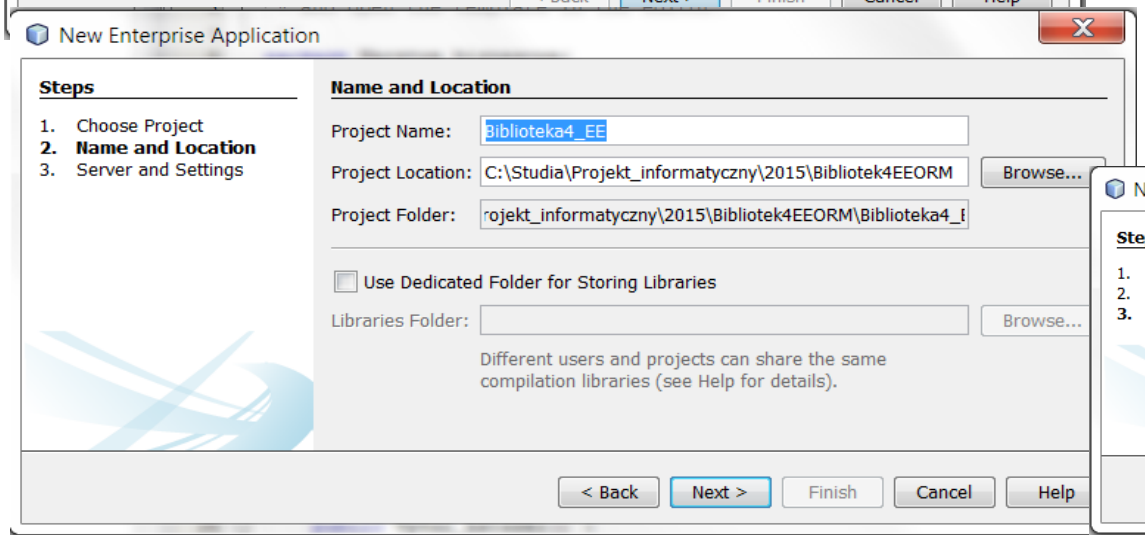
w klasie Tytul_ksiazki, wtedy należałoby wykorzystać tablicę ksiazki, wiążąc dane z obu tablic w dwukierunkowej relacji 1…wiele.
Jeśli nie wystąpią problemy z tym mechanizmem ***cascade****, można ten parametr ksiazki pominąć. Na stronie 34 w metodzie uaktualnij_dane() - można wtedy usunąć również wywołanie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

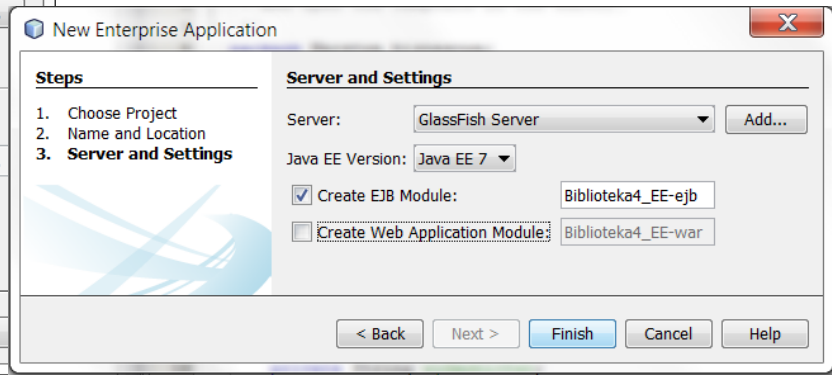
4. Wykonanie projektu EE

4.1. Wykonanie kolejnego projektu typu **Java Class Library** o nazwie **Biblioteka4_Interfejs** (wg p.2.1)

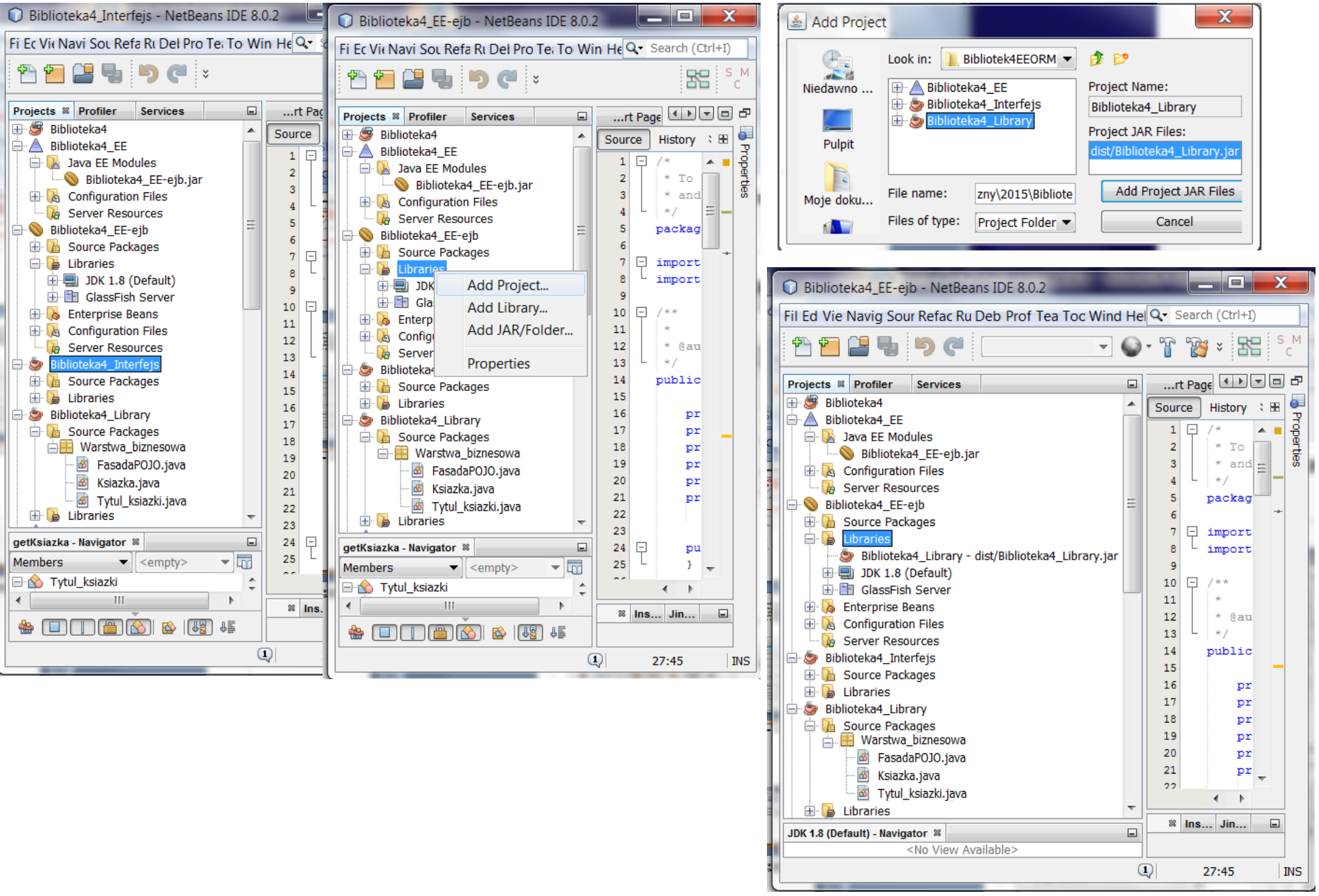
4.2. Wykonanie projektu **Biblioteka4_EE** typu **Java EE** (**File/New Project/Java/Java EE/Enterprise Application** i nacisnąć klawisz **Next**). Następnie w polu **Project Name** wpisać nazwę projektu **Biblioteka4_EE** po wyborze katalogu w polu **Project Location** i nacisnąć klawisz **Next**. W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



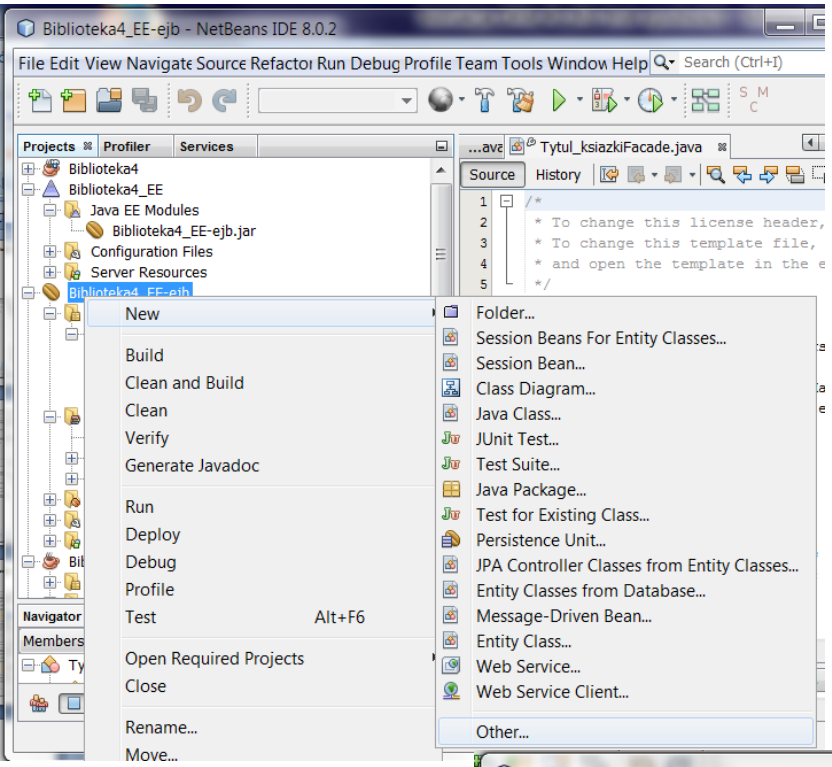


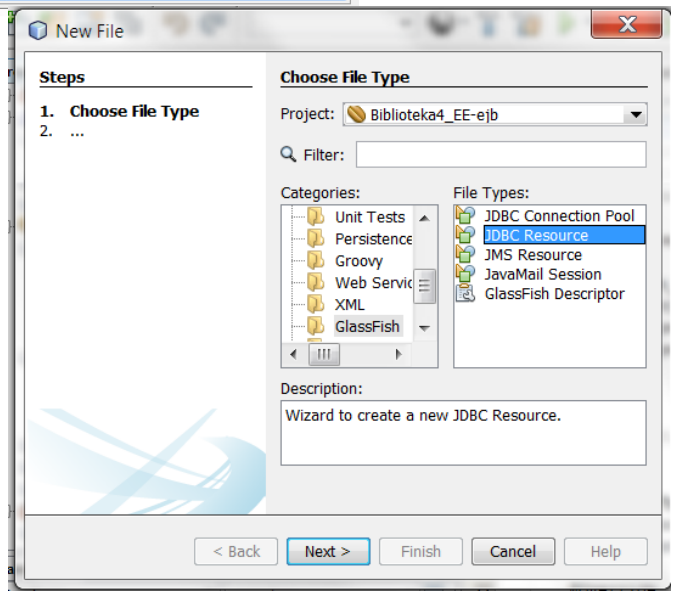


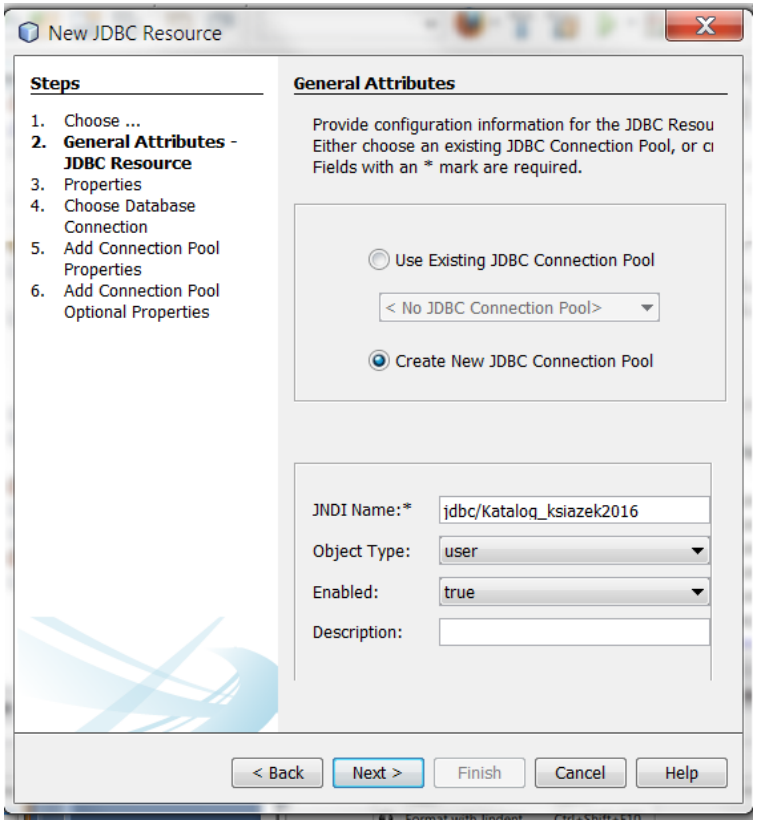
4.3. Dodanie projektu **Biblioteka4_Library** do folderu **Libraries** modułu EJB **Biblioteka4_EE-ejb**



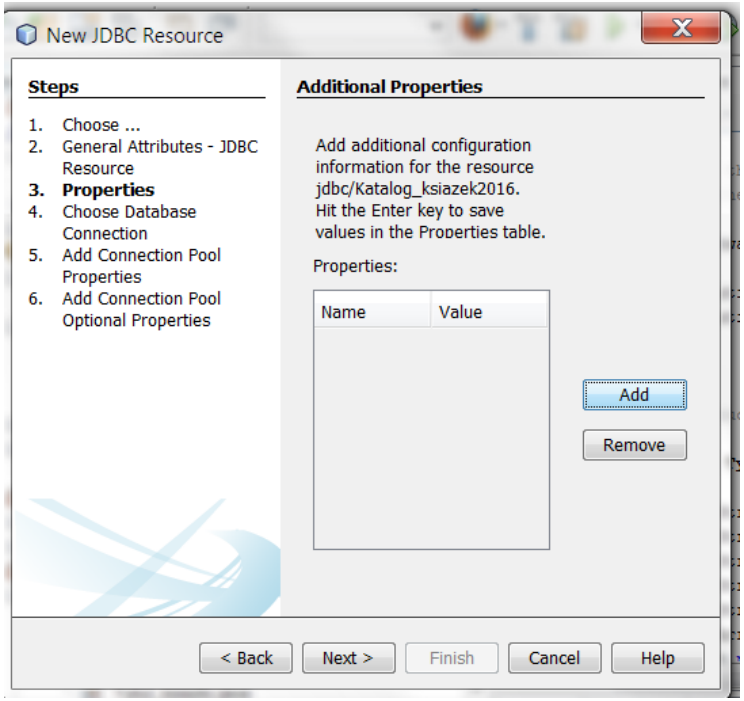
4.4. Wykonanie **JDBC Resources** dla serwera aplikacji GlassFish, dotyczące modułu **Biblioteka4_EE-ejb**

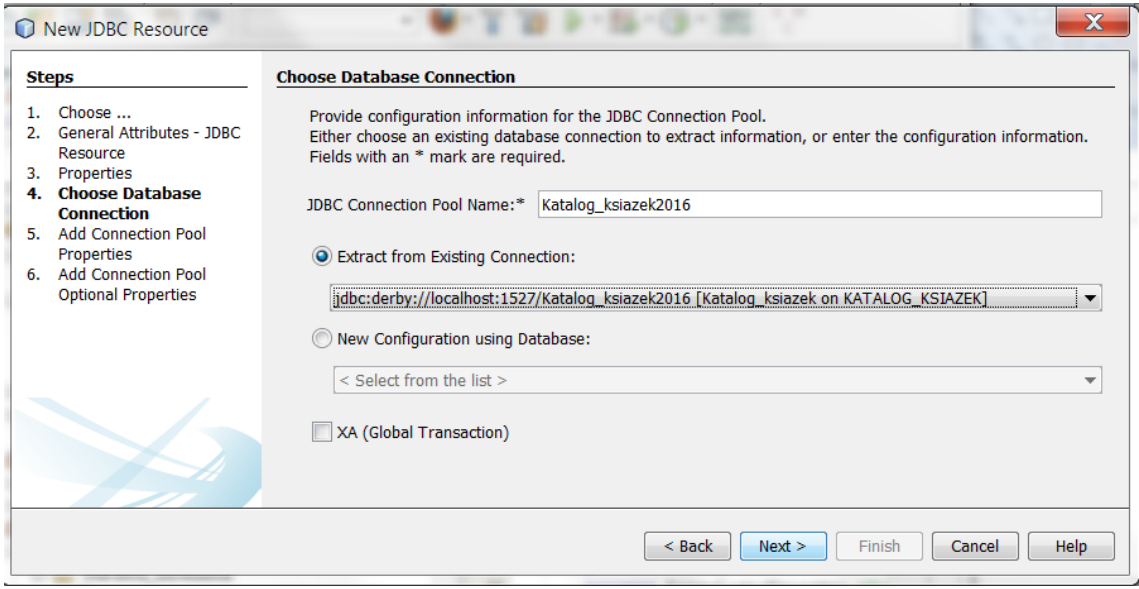


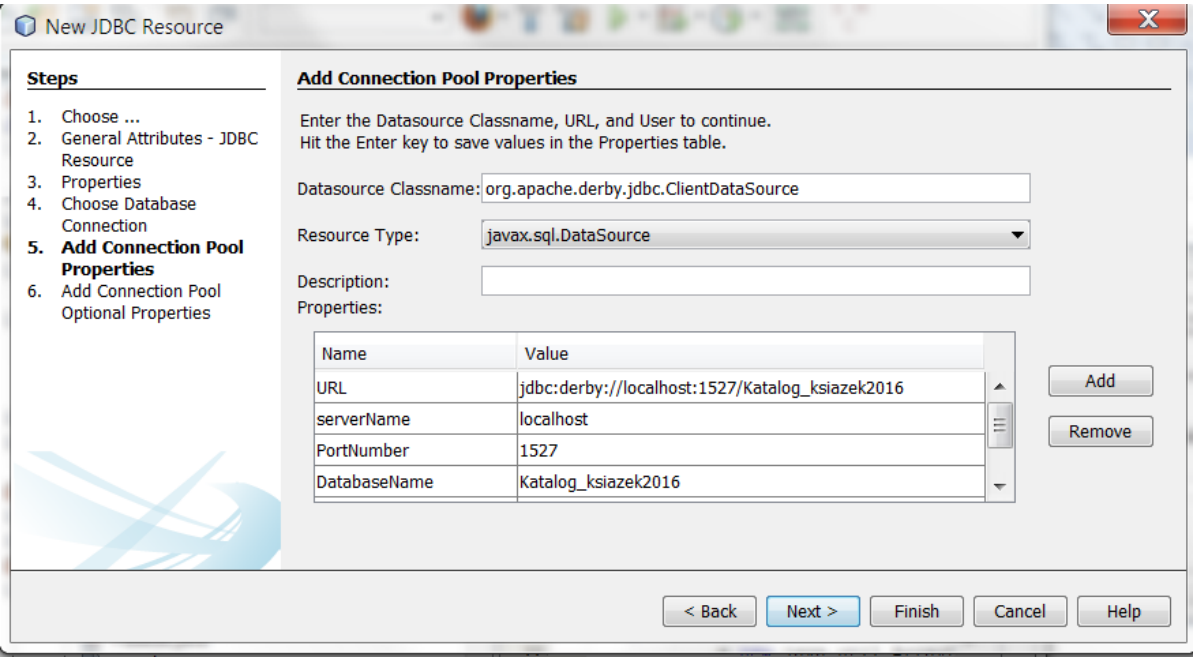


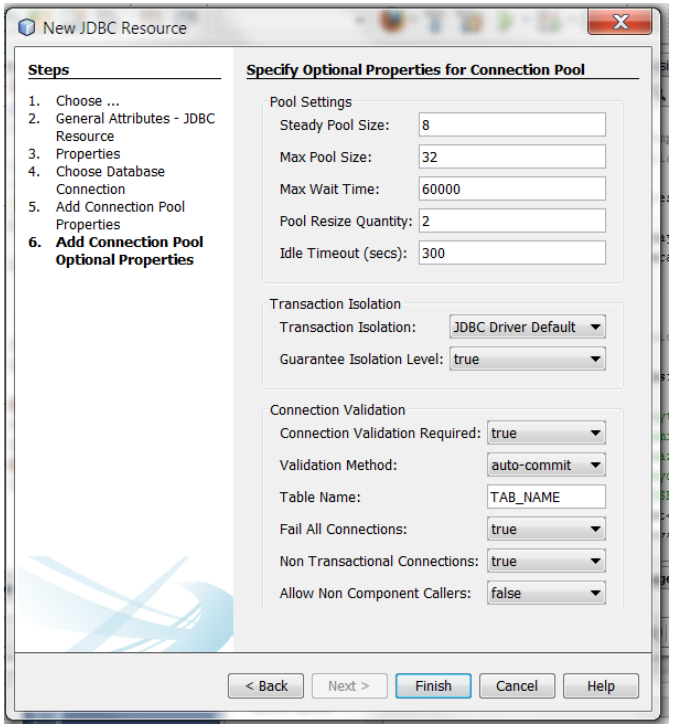


4.4. Wykonanie **JDBC Resources** dla serwera aplikacji GlassFish cd

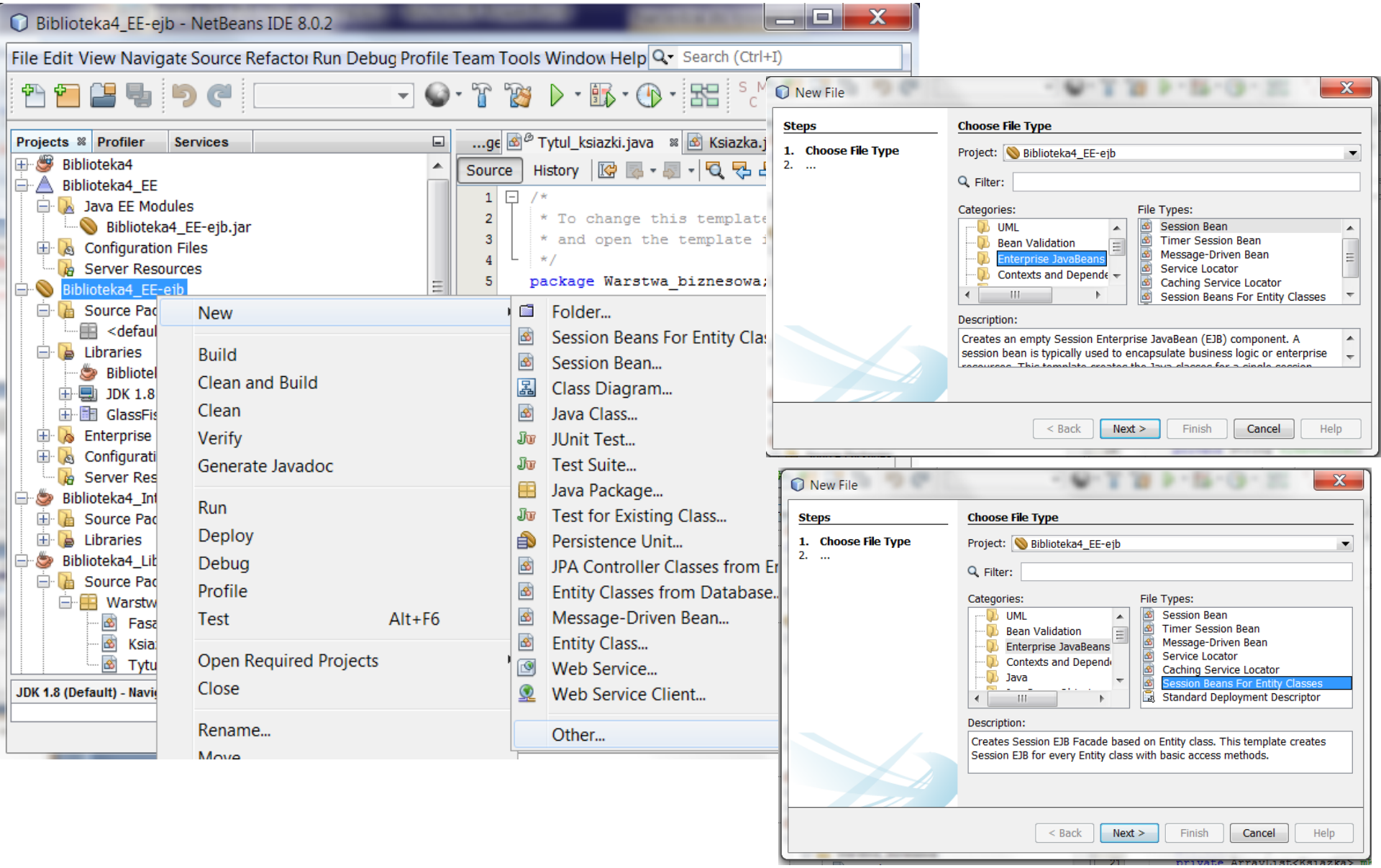




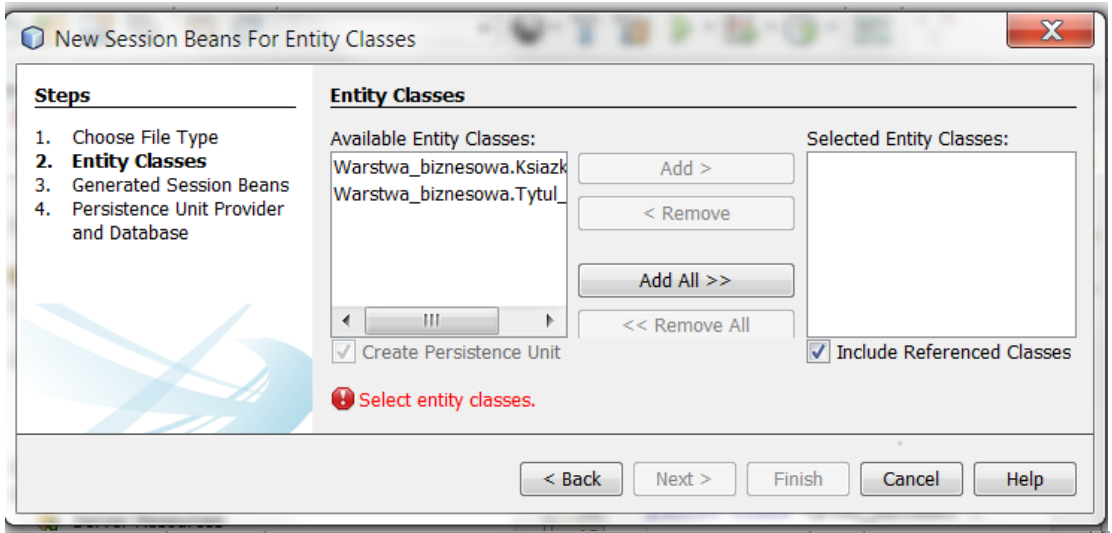


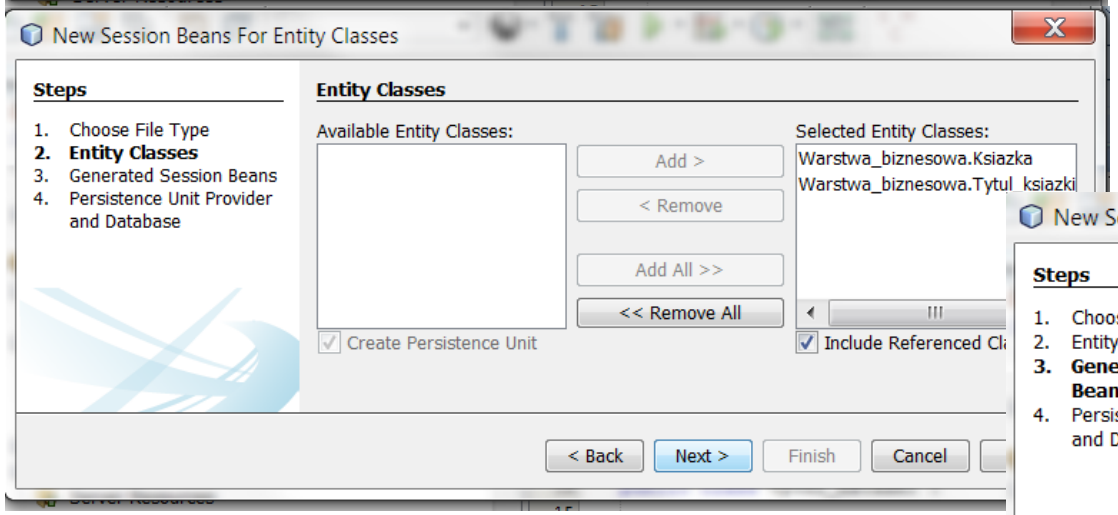


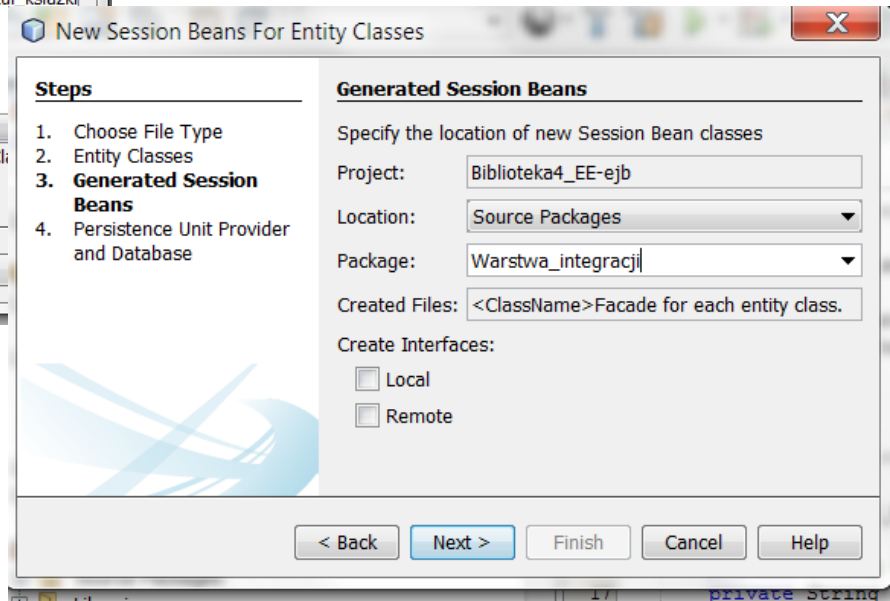
4.5. Wykonanie automatyczne kontrolerów ORM dla klas typu Entity: Tytul_ksiazki i Ksiazka w projekcie **Biblioteka4_EE-ejb**



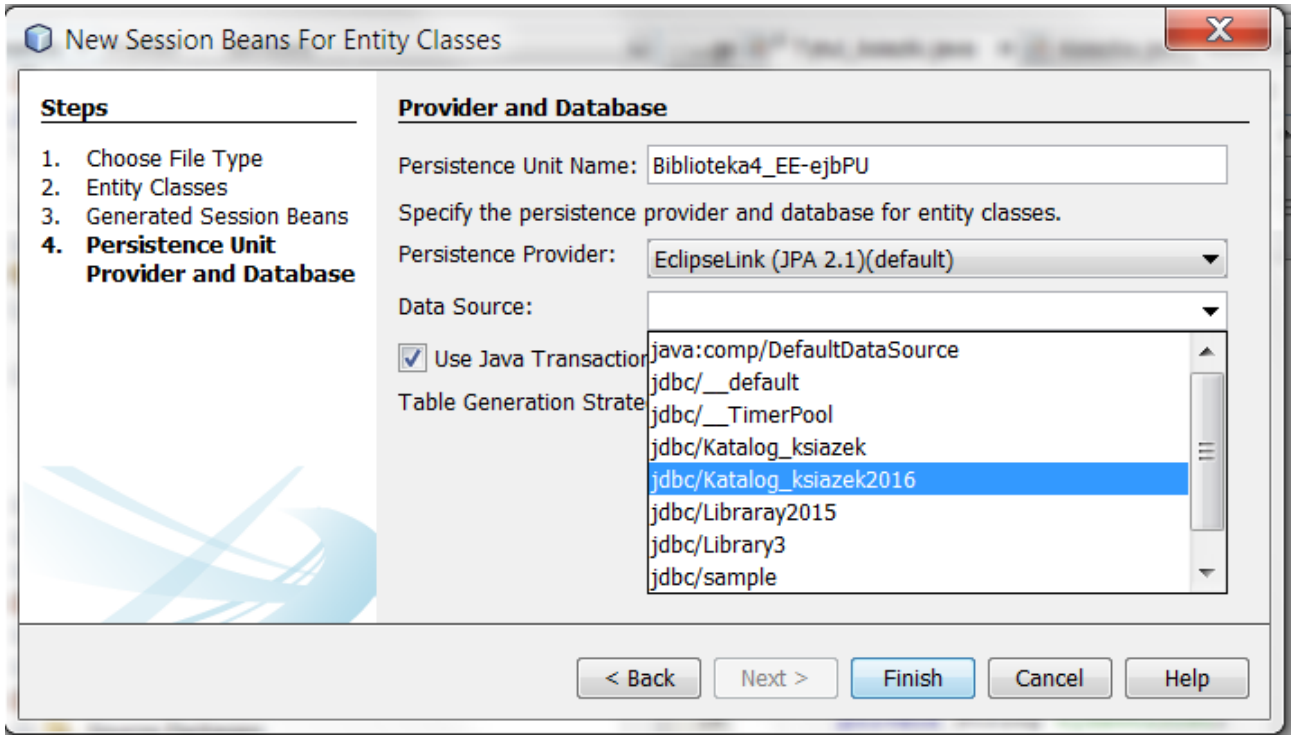
4.6. Należy wybrać encje przez **AddAll**, nacisnąć klawisz **Next**, i podać nazwę pakietu **Warstwa_integracji**, w którym należy umieścić nowe kontrolery typu EJB o dostępie domyślnym (lokalnym)



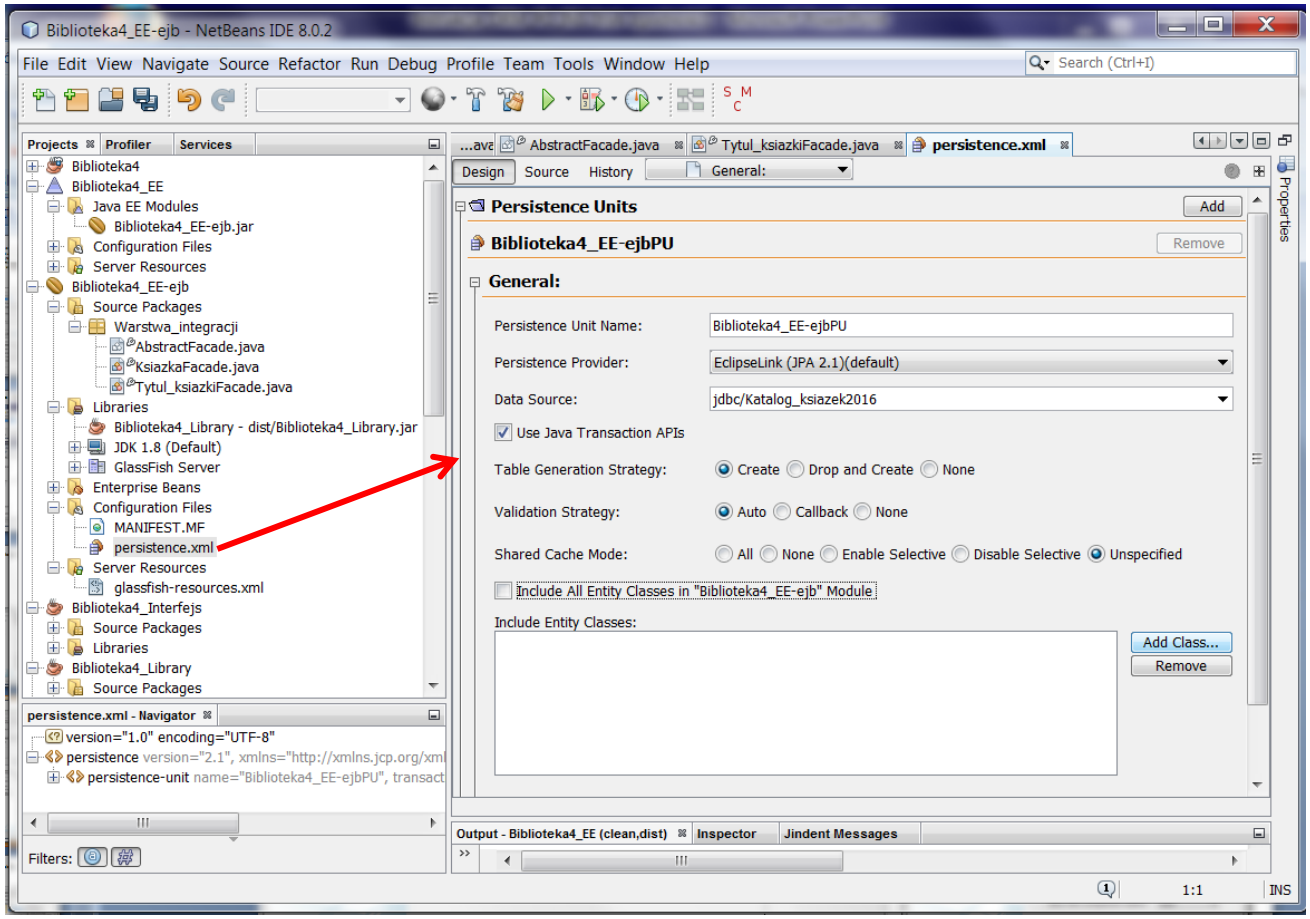


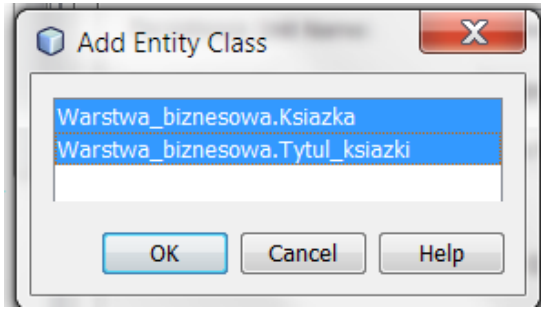


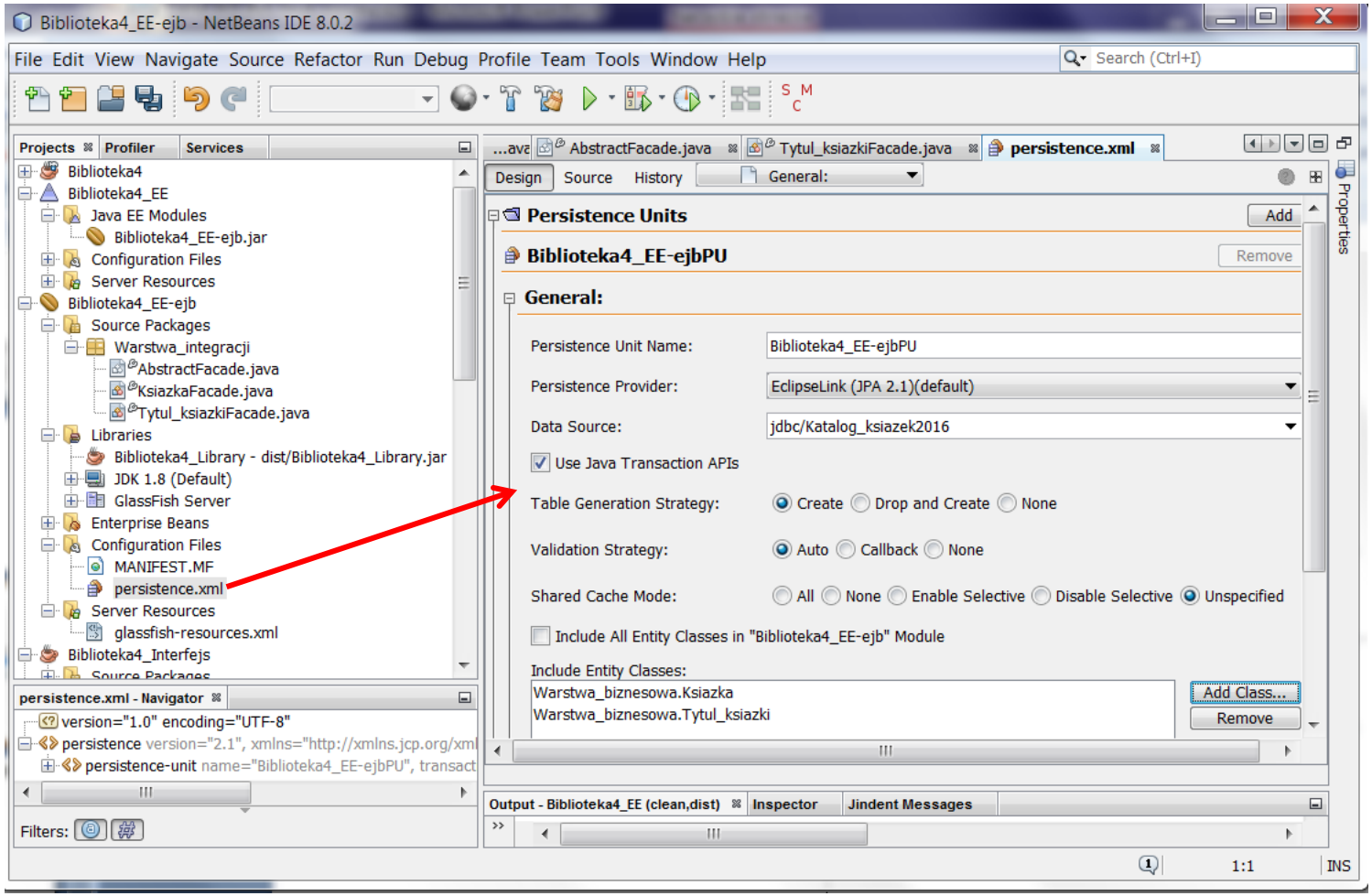
4.7. Należy wybrać identyfikator bazy danych, zdefiniowany w p.4.4 jako **JNDI Name**



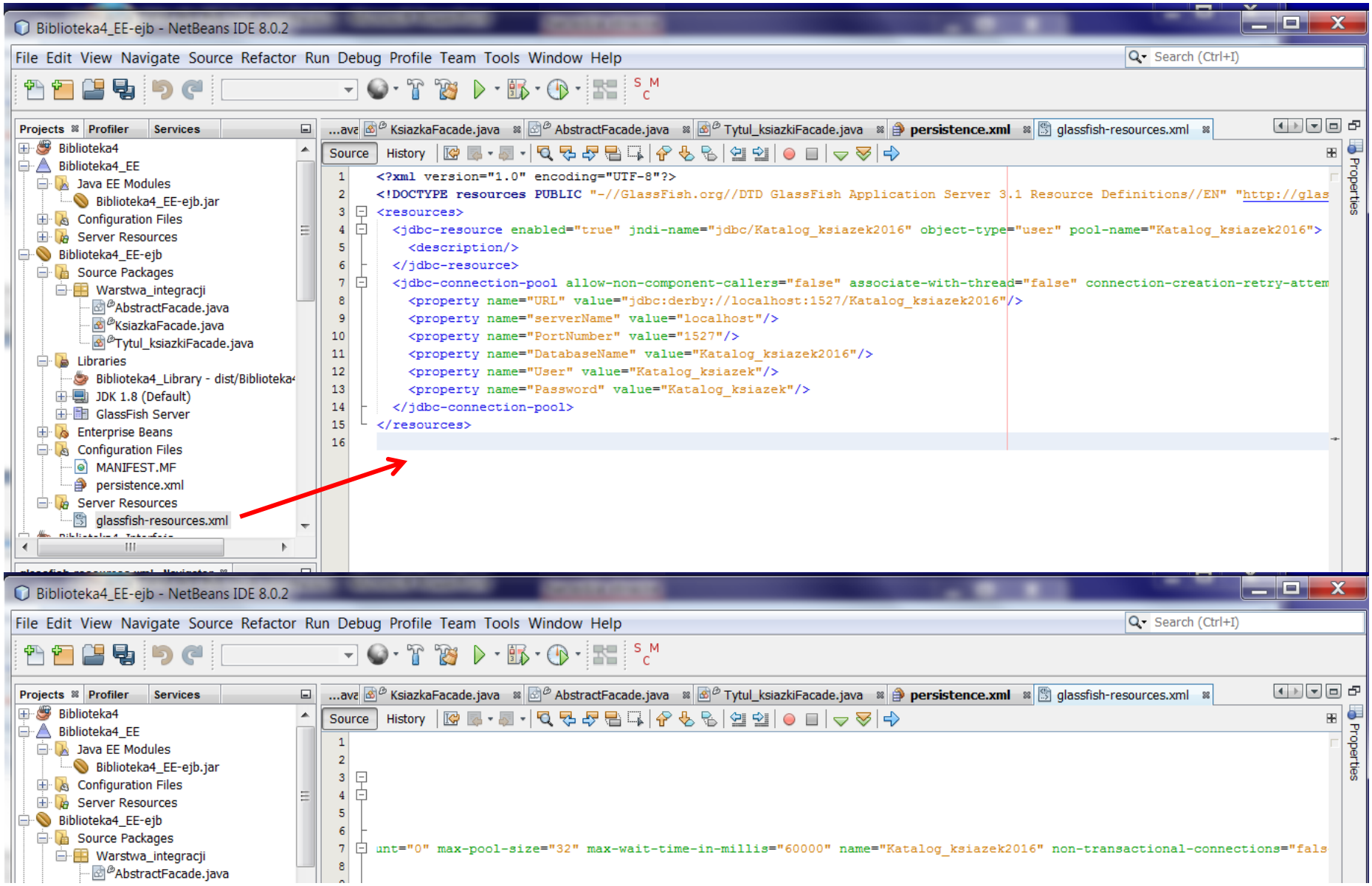
4.8. W utworzonym automatycznie pliku **persistence.xml** umieścić jawnie klasy typu **Entity** przez anulowanie ustawienia Include **All EntityClasses** in „**Biblioteka4_EE-ejb**” i za pomocą przycisku **Add Class**… wybrać w formularzu wszystkie klasy typu **Entity** (**Tytul_ksiazki** i **Ksiazka**)






Biblioteka4_EE-ejb - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Profiler
Services
Biblioteka4
Biblioteka4_EE
Java EE Modules
Biblioteka4_EE-ejb.jar
Configuration Files
Server Resources
Biblioteka4_EE-ejb
Source Packages
Warstwa_integracji
AbstractFacade.java
KsiazkaFacade.java
Tytul_ksiazkiFacade.java
Libraries
Biblioteka4_Library - dist/Biblioteka4_Library.jar
JDK 1.8 (Default)
GlassFish Server
Enterprise Beans
Configuration Files
MANIFEST.MF
persistence.xml
Server Resources
glassfish-resources.xml
Biblioteka4_Interfejs
persistence.xml - Navigator
version="1.0" encoding="UTF-8"
persistence version="2.1", xmlns="http://xmlns.jcp.org/xml
persistence-unit name="Biblioteka4_EE-ejbPU", transact
Filters:
AbstractFacade.java
Tytul_ksiazkiFacade.java
persistence.xml
Design
Source
History
General:
Persistence Units
Add
Biblioteka4_EE-ejbPU
Remove
General:
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
Add Class...
Remove
Properties
Output - Biblioteka4_EE (clean,dist)
Inspector
Jindent Messages
1:1
INS

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



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_ksiazki** (na lewo) oraz po wstawieniu kodu (na prawo) do dodania zbioru obiektów typu **Tytul_ksiazki** do bazy danych (**dodaj_tytuly**) oraz pobranie tych danych z bazy danych do wyświetlenia (**tytuly_)**

```
...5 lines
package Warstwa_integracji;

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

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

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    public Tytul_ksiazkiFacade() {
        super(Tytul_ksiazki.class);
    }
}
```

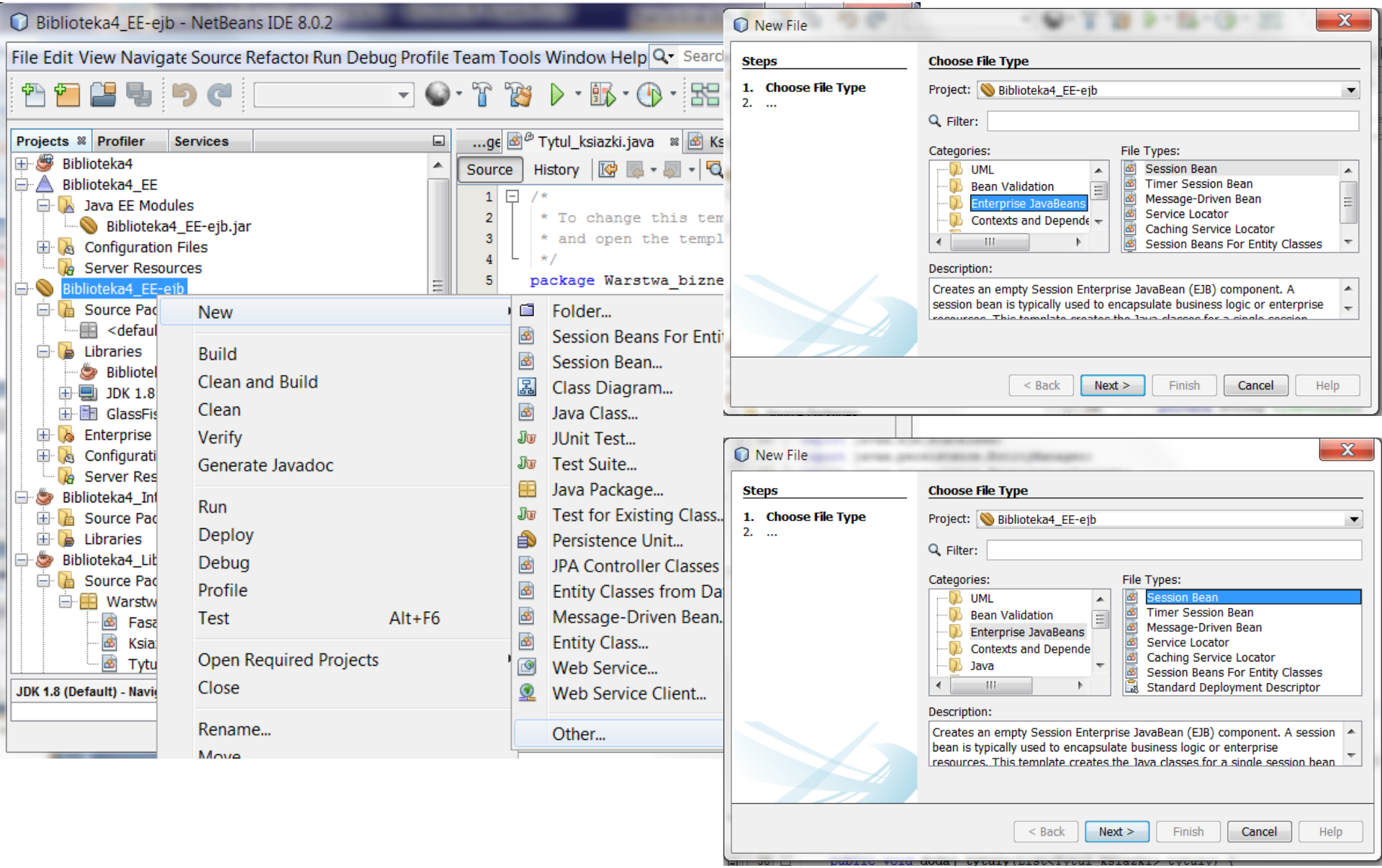
```
package Warstwa_integracji;
import Warstwa_biznesowa.Tytul_ksiazki;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

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

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }
    public Tytul_ksiazkiFacade() {
        super(Tytul_ksiazki.class);
    }
     public ArrayList<String> tytuly_() {
        ArrayList<String> tytuly = new ArrayList();
        Tytul_ksiazki[] tytuly_ = this.findAll().toArray(new Tytul_ksiazki[0]);
        for (Tytul_ksiazki t : tytuly_) {
            tytuly.add(t.toString());
        }
        return tytuly;
    }
    public void dodaj_tytuly(List<Tytul_ksiazki> tytuly) {
        for (Tytul_ksiazki 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_)**

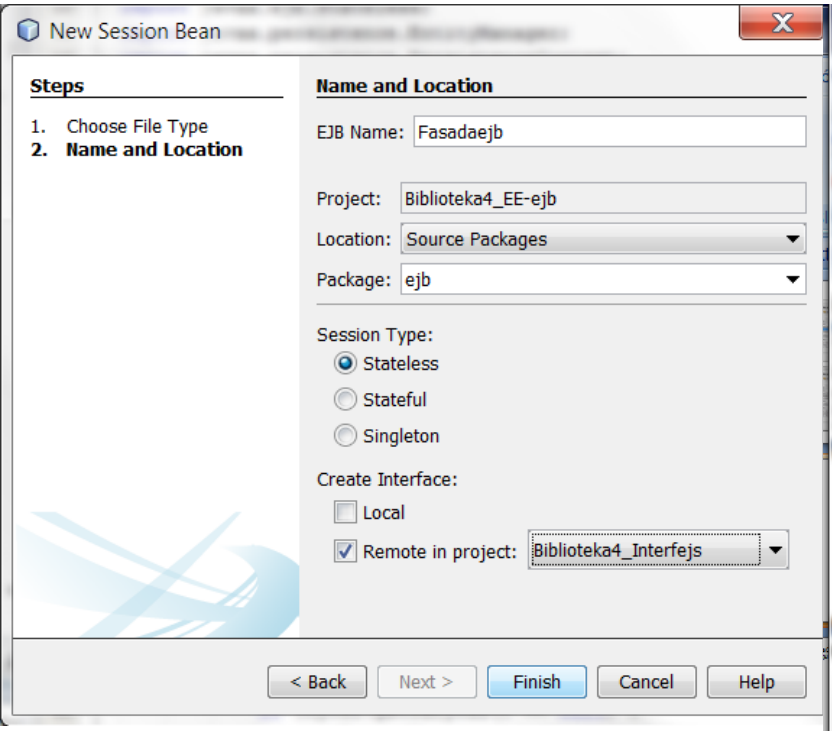
```
package Warstwa_integracji;

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);
    }
}
```

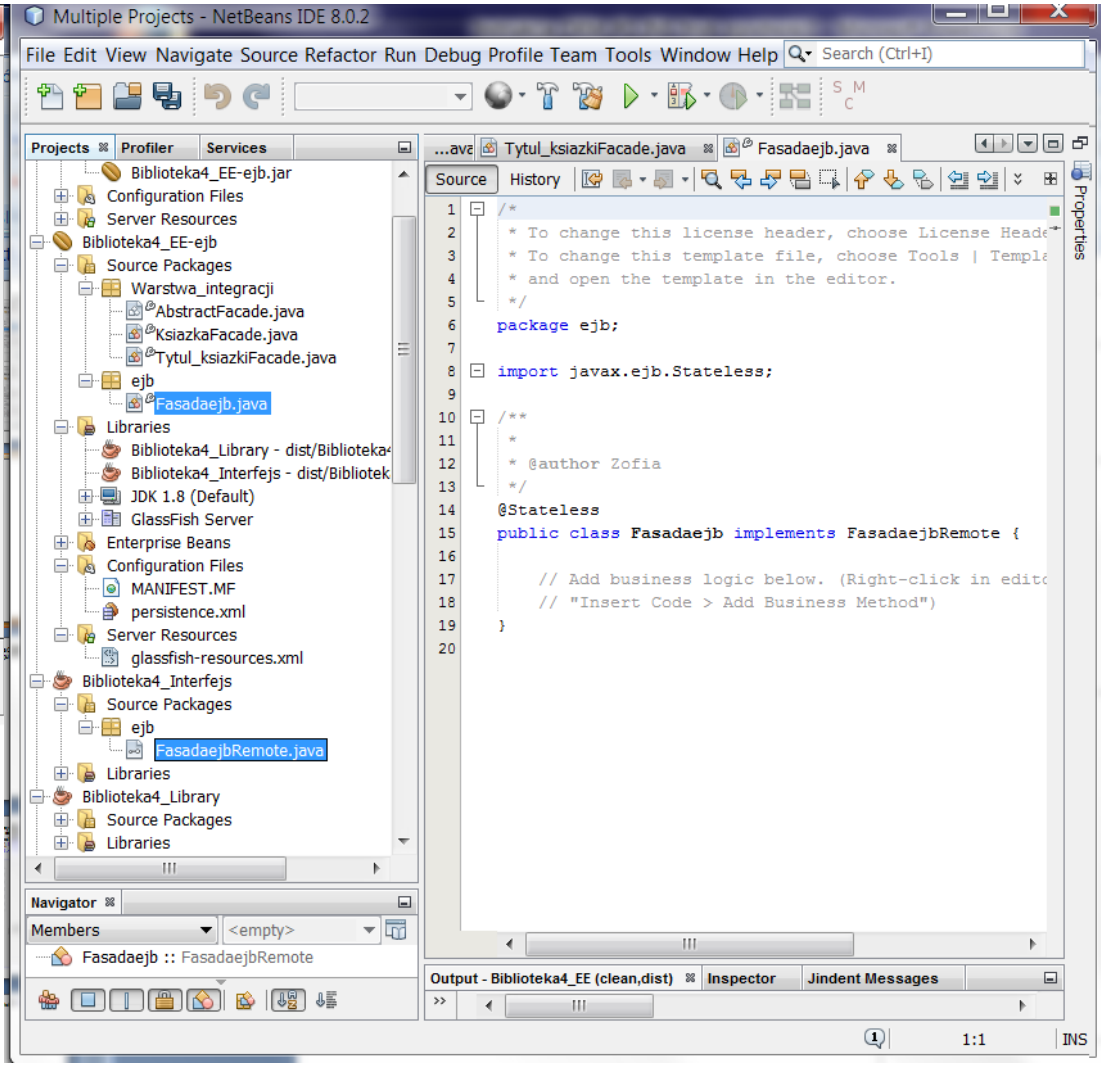
```
package Warstwa_integracji;
import Warstwa_biznesowa.Ksiazka;
import Warstwa_biznesowa.Tytul_ksiazki;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**...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

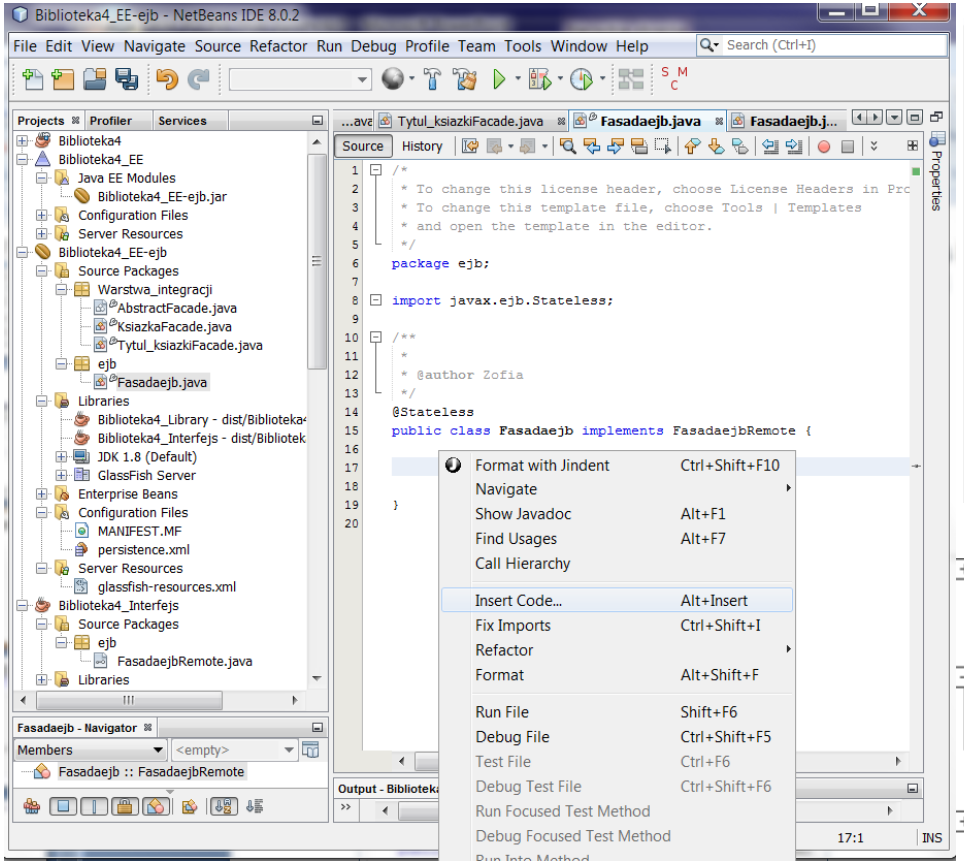


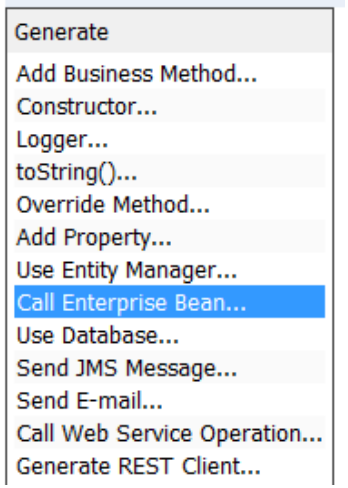
4.13. Dodanie klasy **Fasadaejb** typu EJB (Session Bean) o dostępie Remote do modułu **Biblioteka4_EE-ejb** do obsługi komunikacji typu Klient-Serwer cd

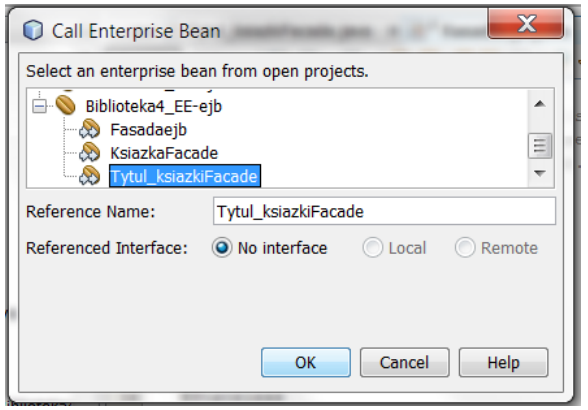




4.14. Wykonanie połączenia z kontrolerami ORM typu EJB dla encji **Tytul_ksiazki** i **Ksiazka** w klasie **Fasadejb** o dostępie Remote za pomocą tzw adnotacji.

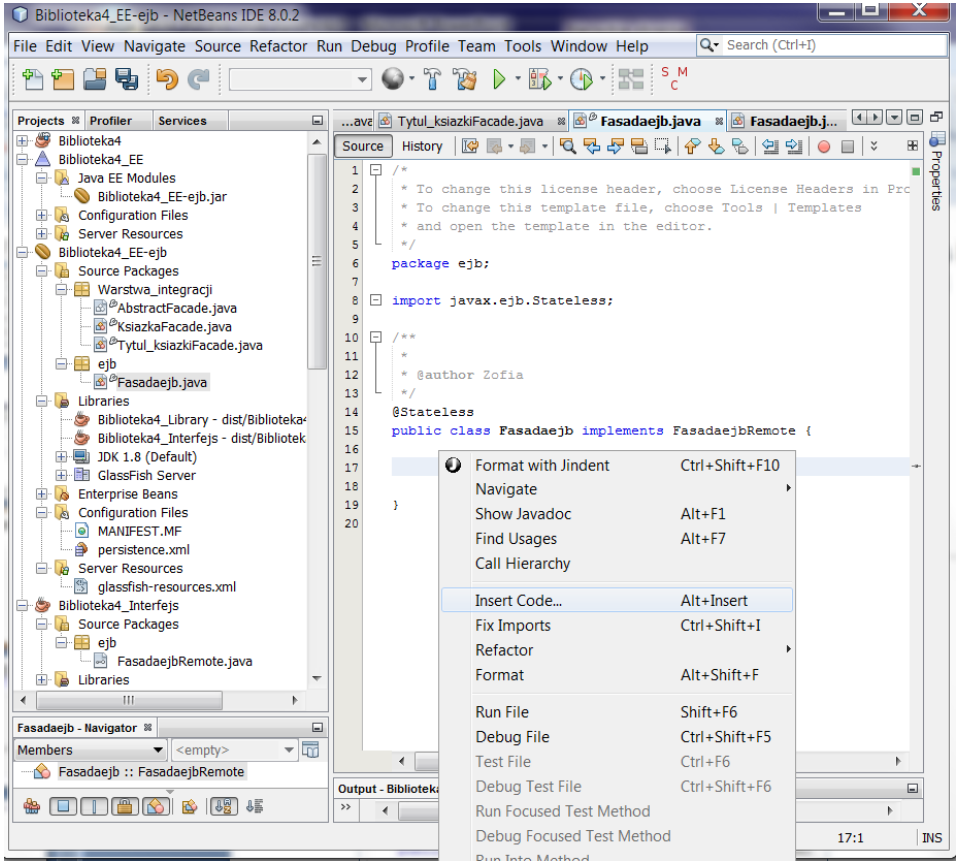


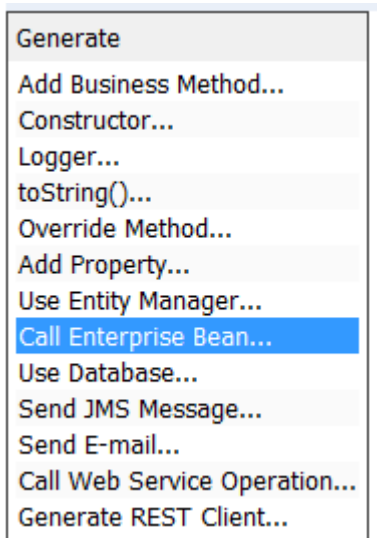


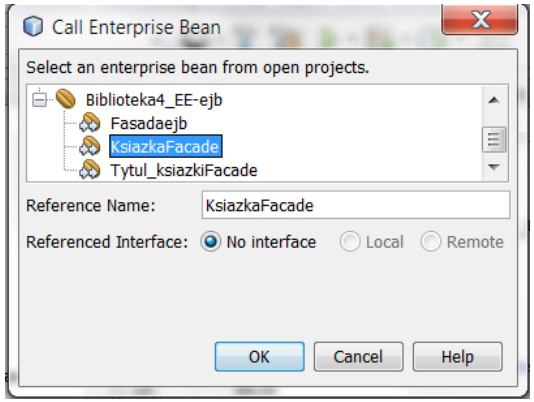



```
...5 lines
package ejb;

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

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


}
```

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








```
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 pamięci
    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**

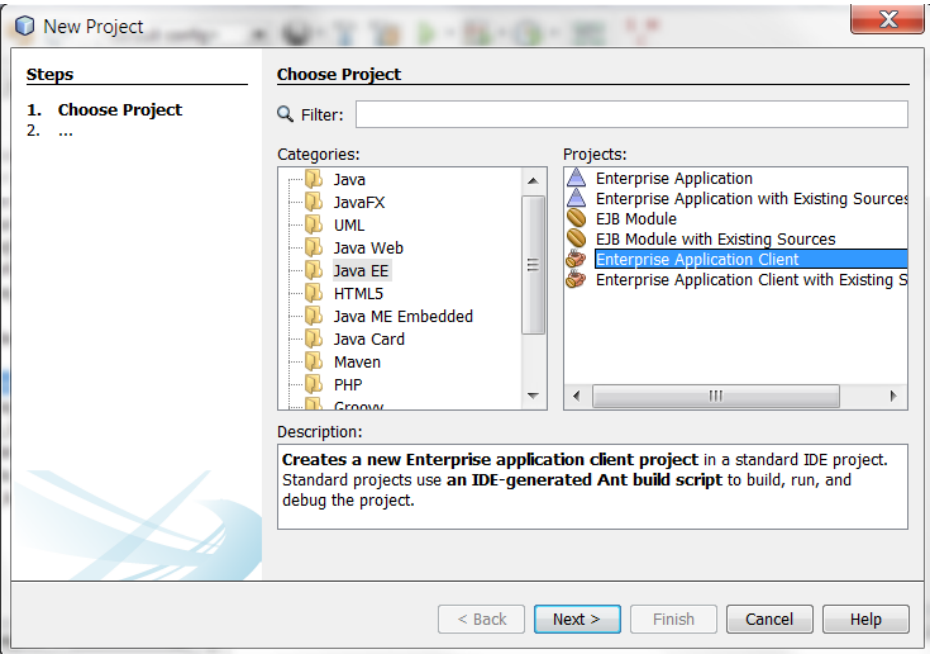
```
package ejb;
import Warstwa_biznesowa.FasadaPOJO;
import Warstwa_biznesowa.Ksiazka;
import Warstwa_biznesowa.Tytul_ksiazki;
import Warstwa_integracji.KsiazkaFacade;
import Warstwa_integracji.Tytul_ksiazkiFacade;
import java.util.ArrayList;
import javax.annotation.PostConstruct;
import javax.ejb.EJB;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

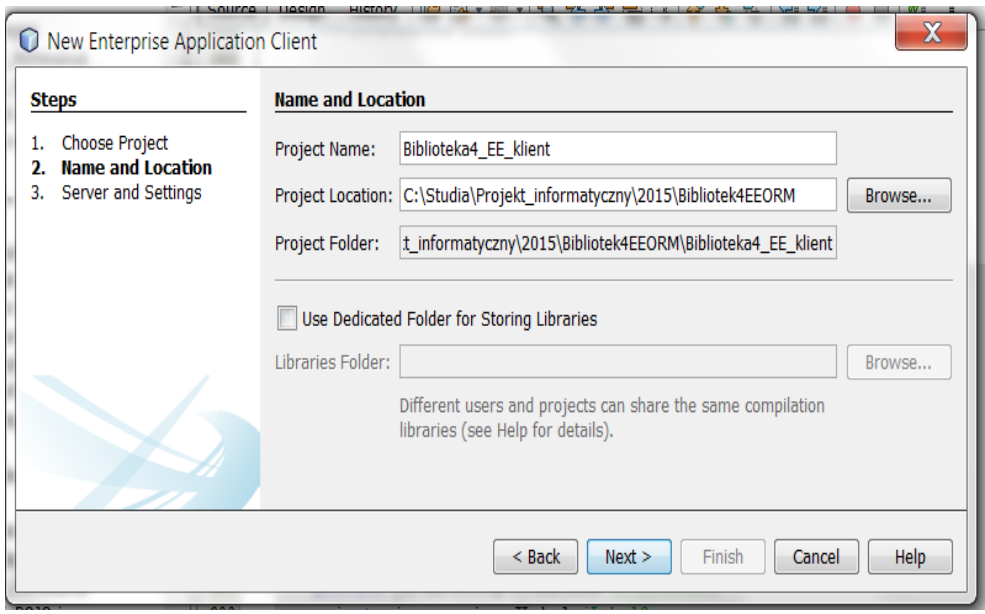
```
    public void uaktualnij_tytuly() throws Exception {
        tytuly = tytul_ksiazkiFacade.findAll().toArray(new Tytul_ksiazki[0]);
    }

    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_();
    }

}
```

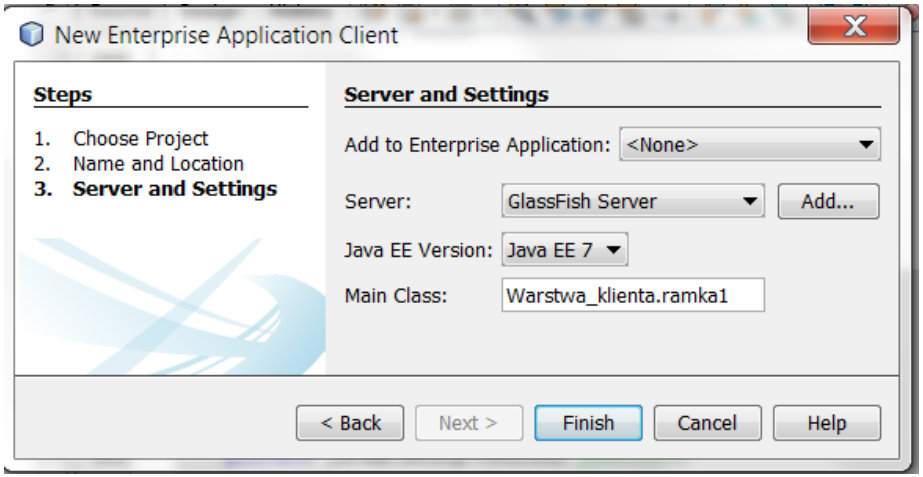
4.17. Dodanie adnotacji **Override** do wykonanych metod w klasie **Fasadaejb**

```
    }
    @Override
     public void dodaj_tytul(String dane_tytul[]) {...2 lines }

    @Override
    public void dodaj_ksiazke(String dane[]) {...2 lines }

    @Override
    public ArrayList<String> tytuly() {...2 lines }

    @Override
    public ArrayList<String> ksiazki() {...2 lines }


    @Override
    public void uaktualnij_tytuly() throws Exception {...3 lines }

    @Override
    public void uaktualnij_ksiazki() 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_ksiazki() throws Exception {...3 lines }

    @Override
    public ArrayList<String> ksiazki_() throws Exception {...3 lines }

    @Override
    public ArrayList<String> tytuly_() throws Exception {...3 lines }

}
```

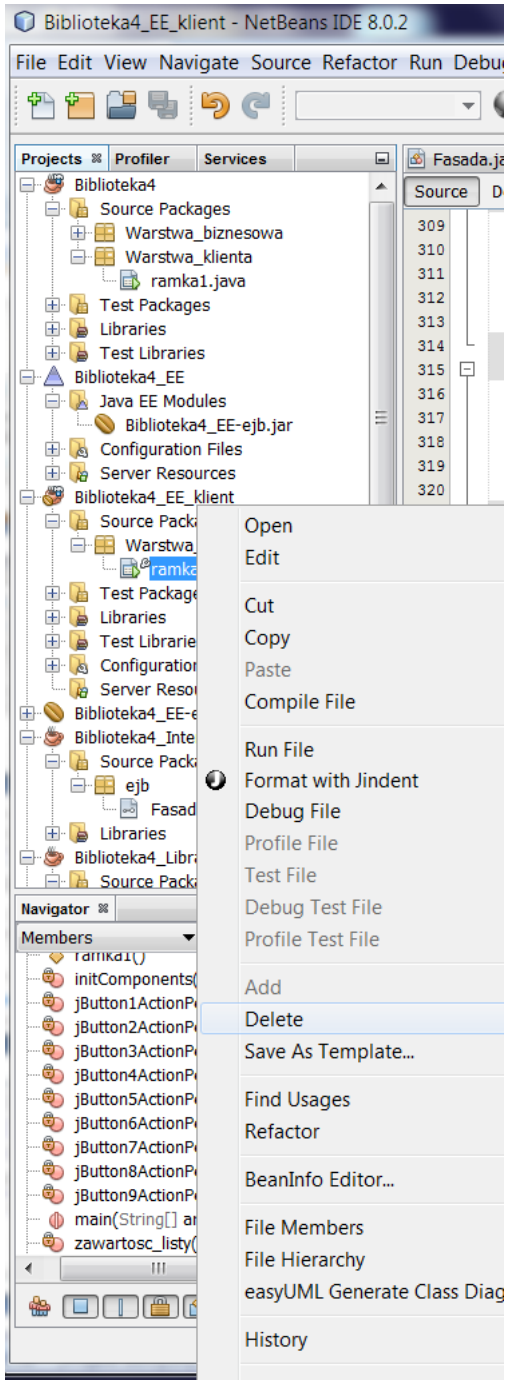
4.18. Dodanie projektu EE **Biblioteka4_EE_klient** typu **Enterprise Application Client**



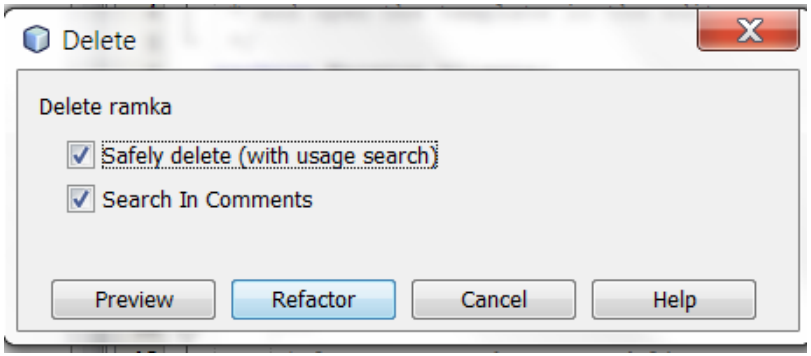




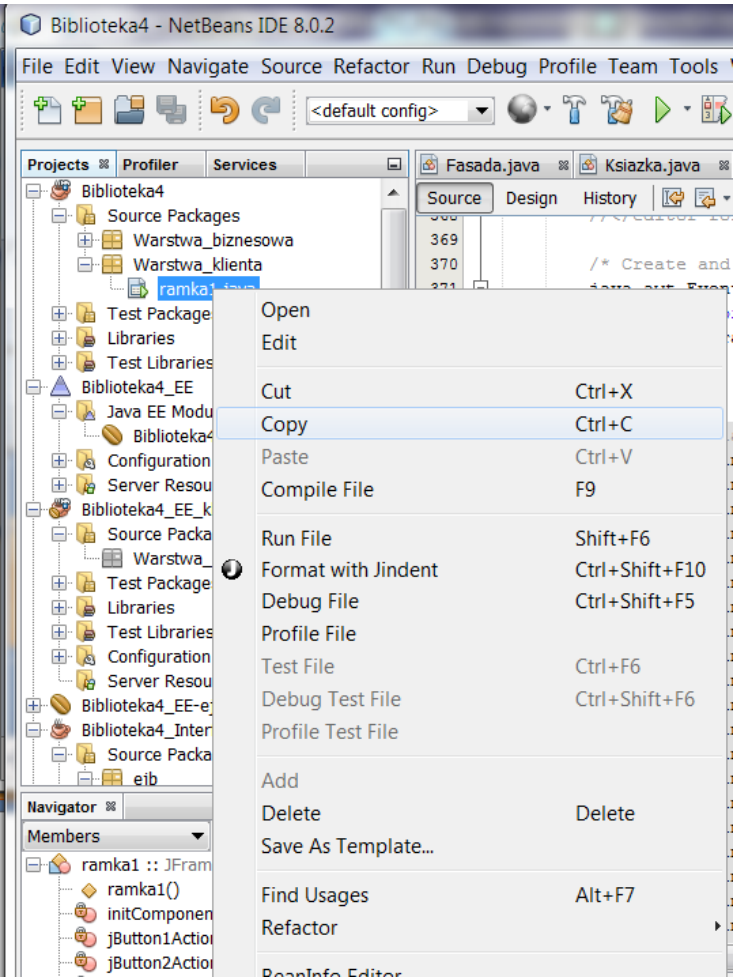
Definicja klasy o nazwie pakietowej takiej samej, jaką w projekcie SE posiada klasa reprezentująca GUI

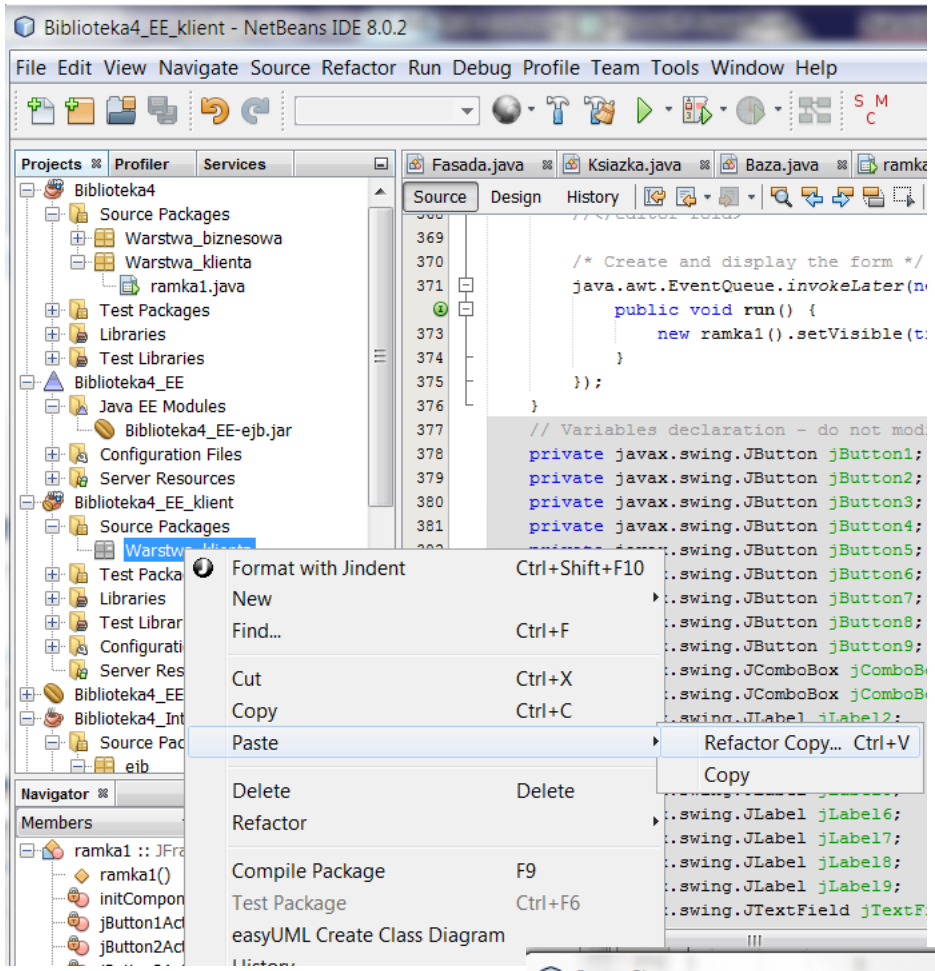


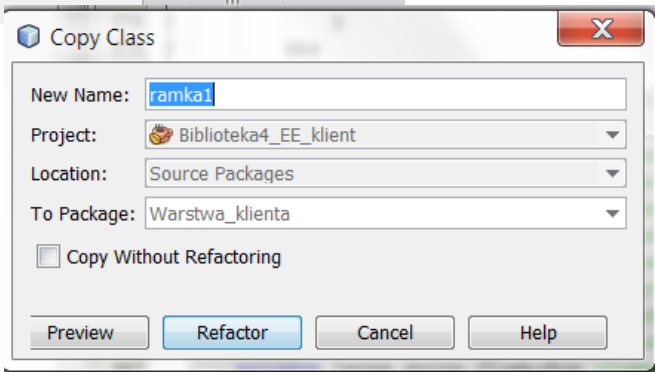
4.18. Bezpieczne usunięcie wykonanej klasy o nazwie pakietowej takiej samej, jak w projekcie SE posiada klasa reprezentująca GUI -cd



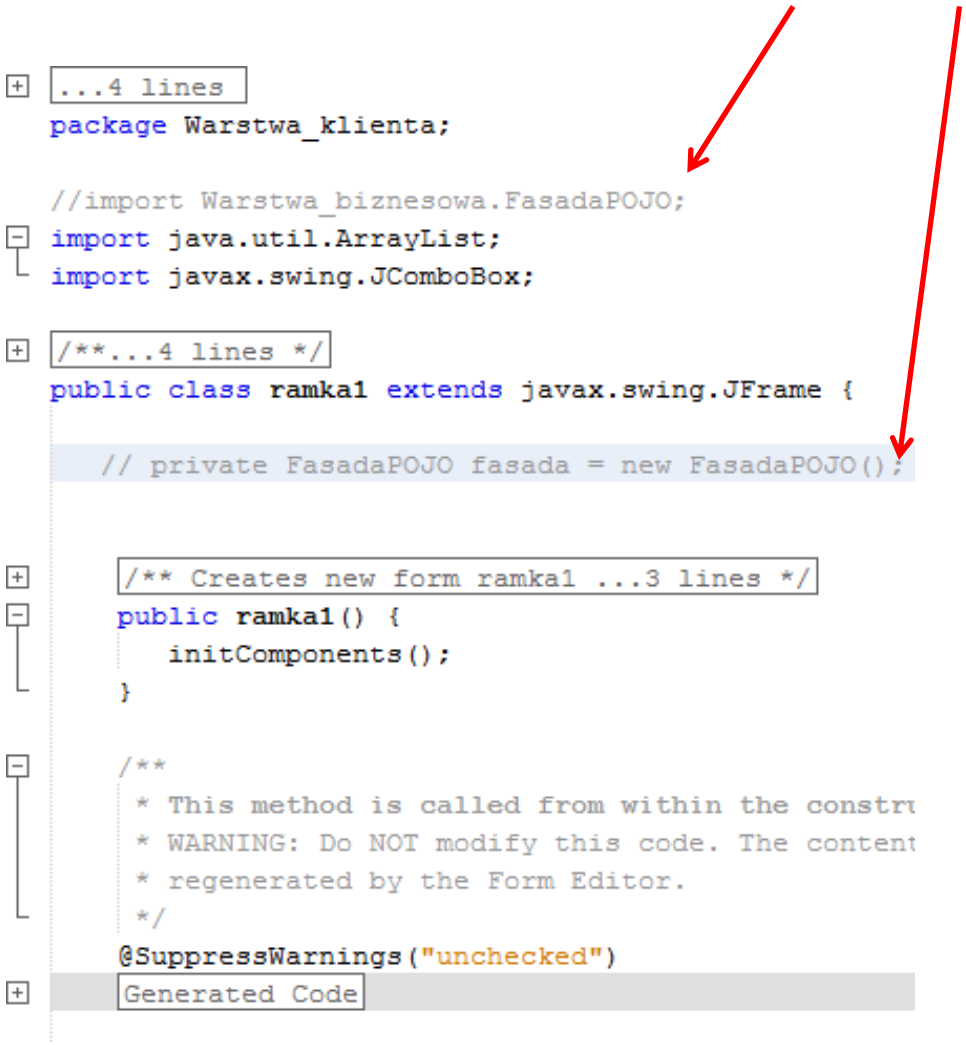
4.19. Skopiowanie klasy **ramka1** do obsługi GUI z projektu SE (Biblioteka4) do projektu **Biblioteka4_EE_klient**





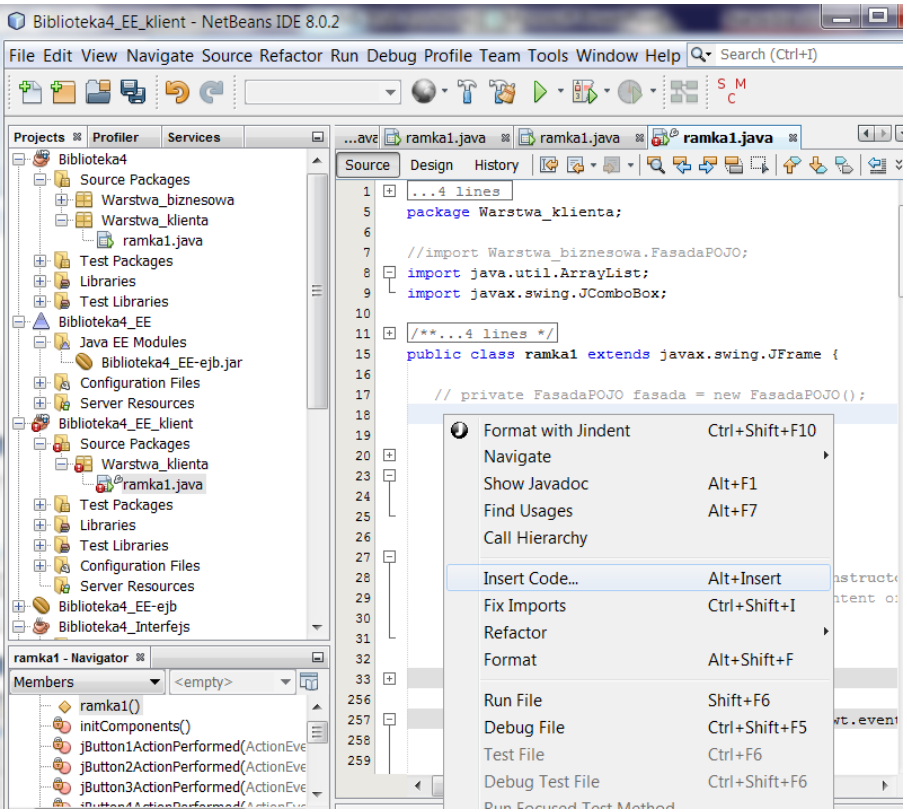


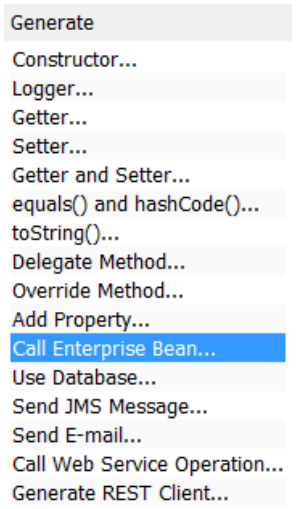
4.20. Przygotowanie klasy **ramka1** do wstawienia adnotacji typu EJB do obiektu klasy **Fasadaejb**, umożliwiającej zdalny dostęp do logiki biznesowej i bazy danych

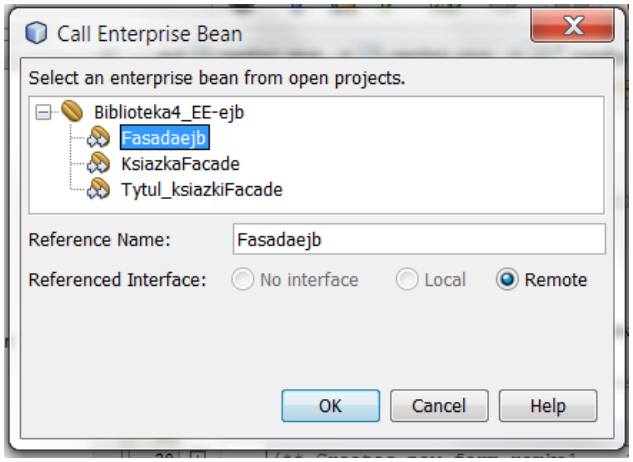


```
...4 lines
package Warstwa_klienta;

//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 constru
     * 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 **Fasadejb** w klasie **ramka1** cd








```
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); }
        });
    }
```

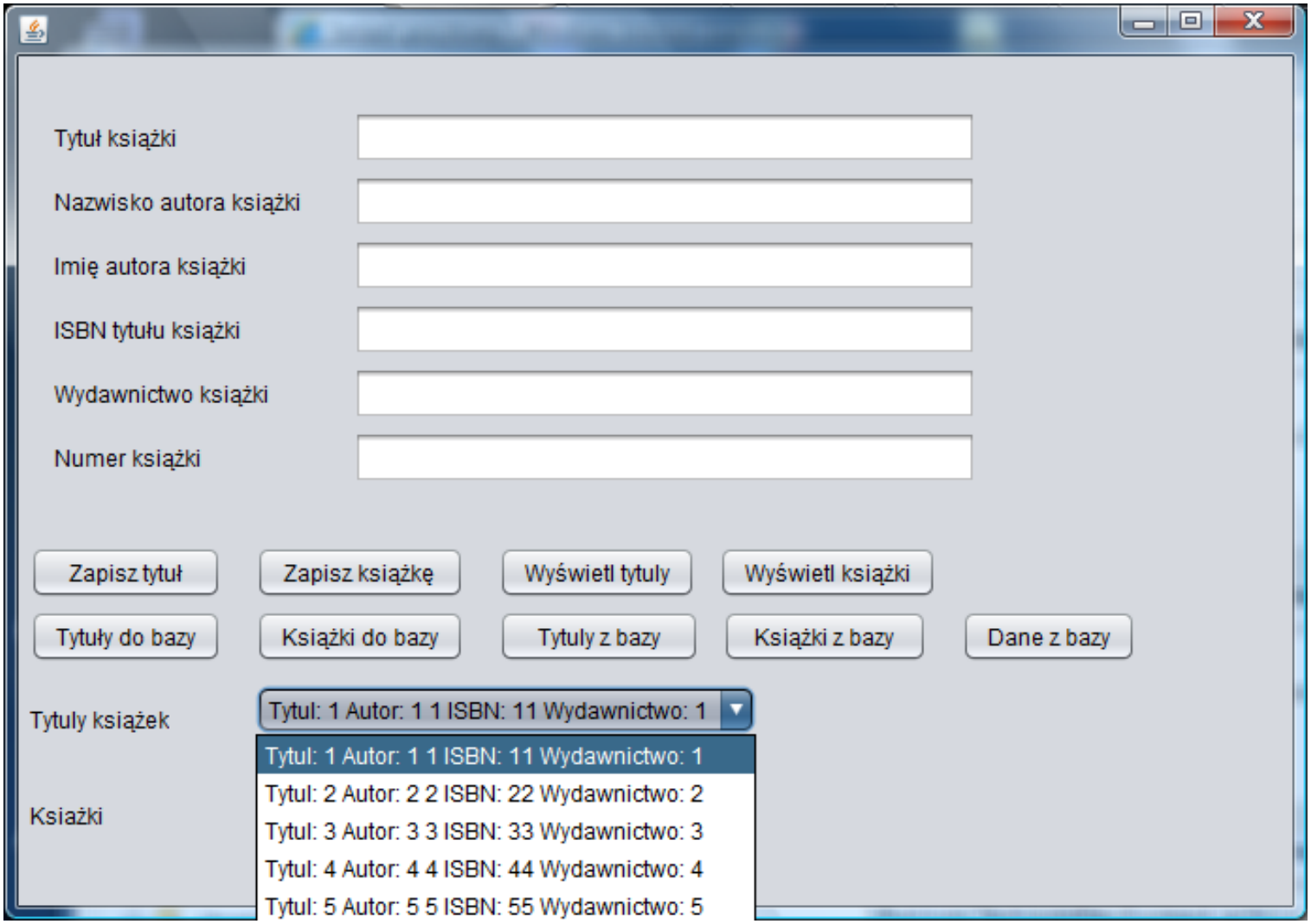
```
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    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

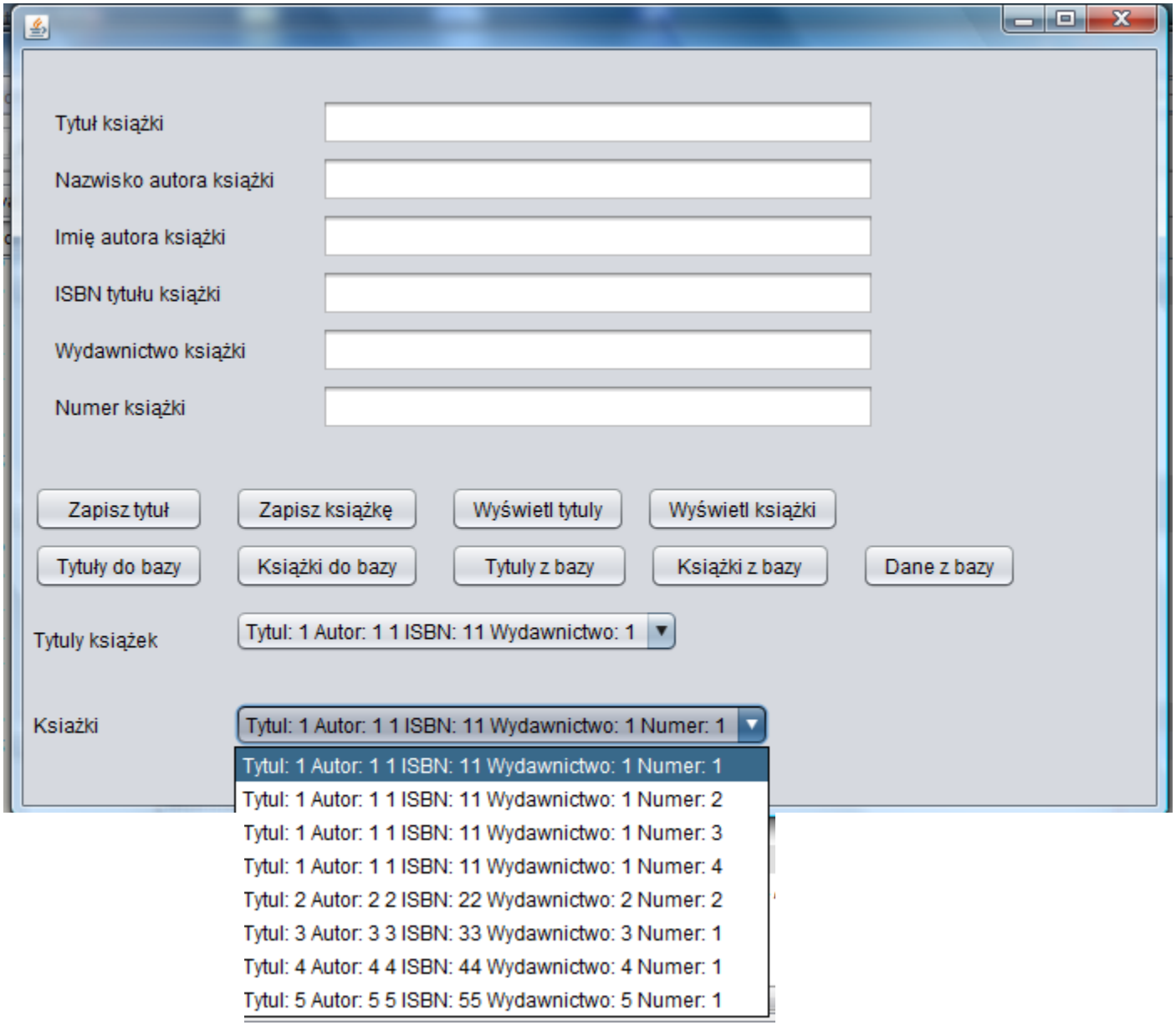
1. Należy kolejno wykonać Clean and Build projektu **Biblioteka4_Library** (biblioteka klas biznesowych)
2. Należy kolejno wykonać Clean and Build projektu **Biblioteka4_Interfejs** (interfejs zawierający deklaracje metod o zdalnym dostępie)
3. Należy kolejno wykonać Clean and Build projektu **Biblioteka4_EE-ejb** (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: **Tytuly 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**)

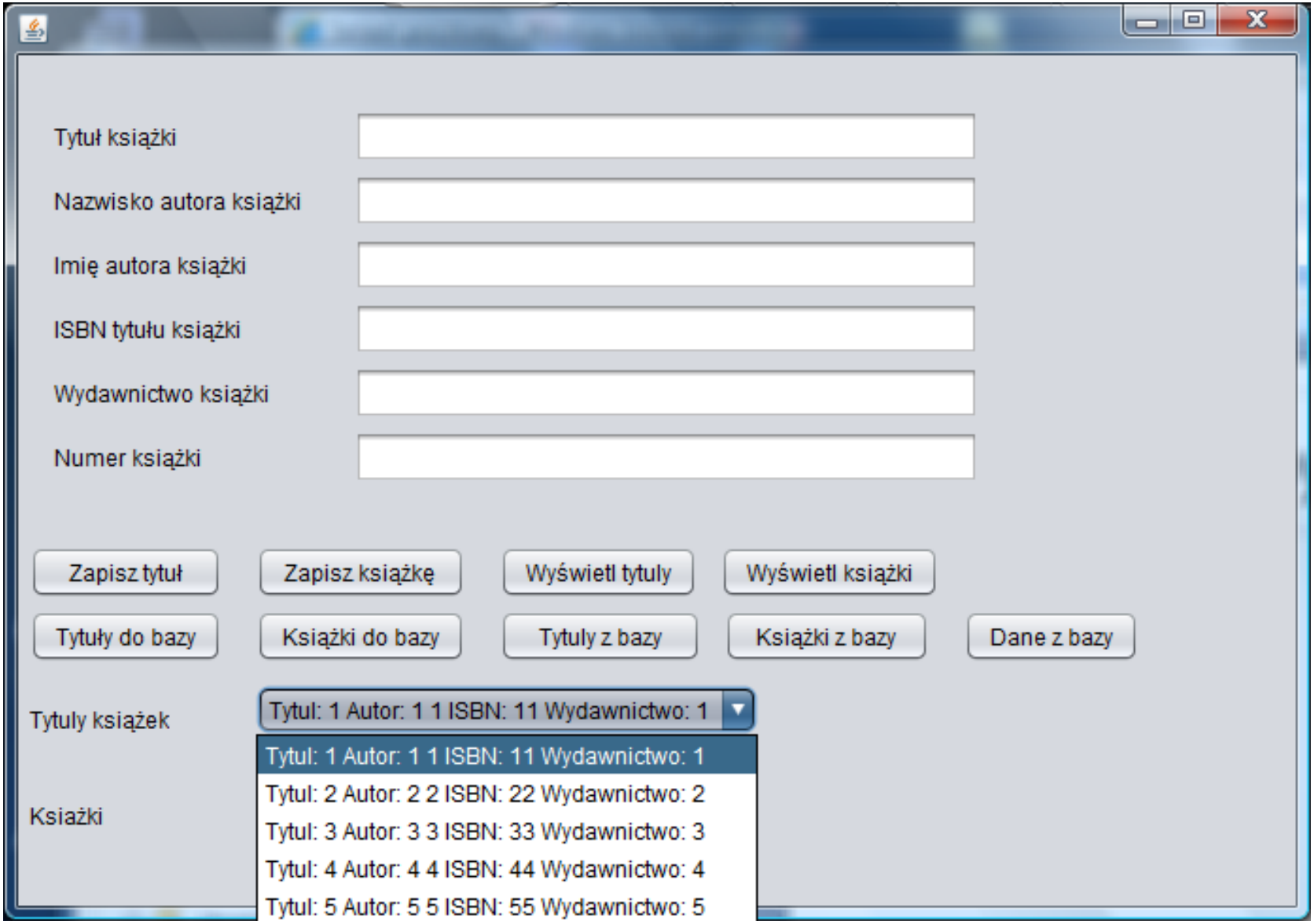
6.1. Wyświetlenie tytułów z aplikacji



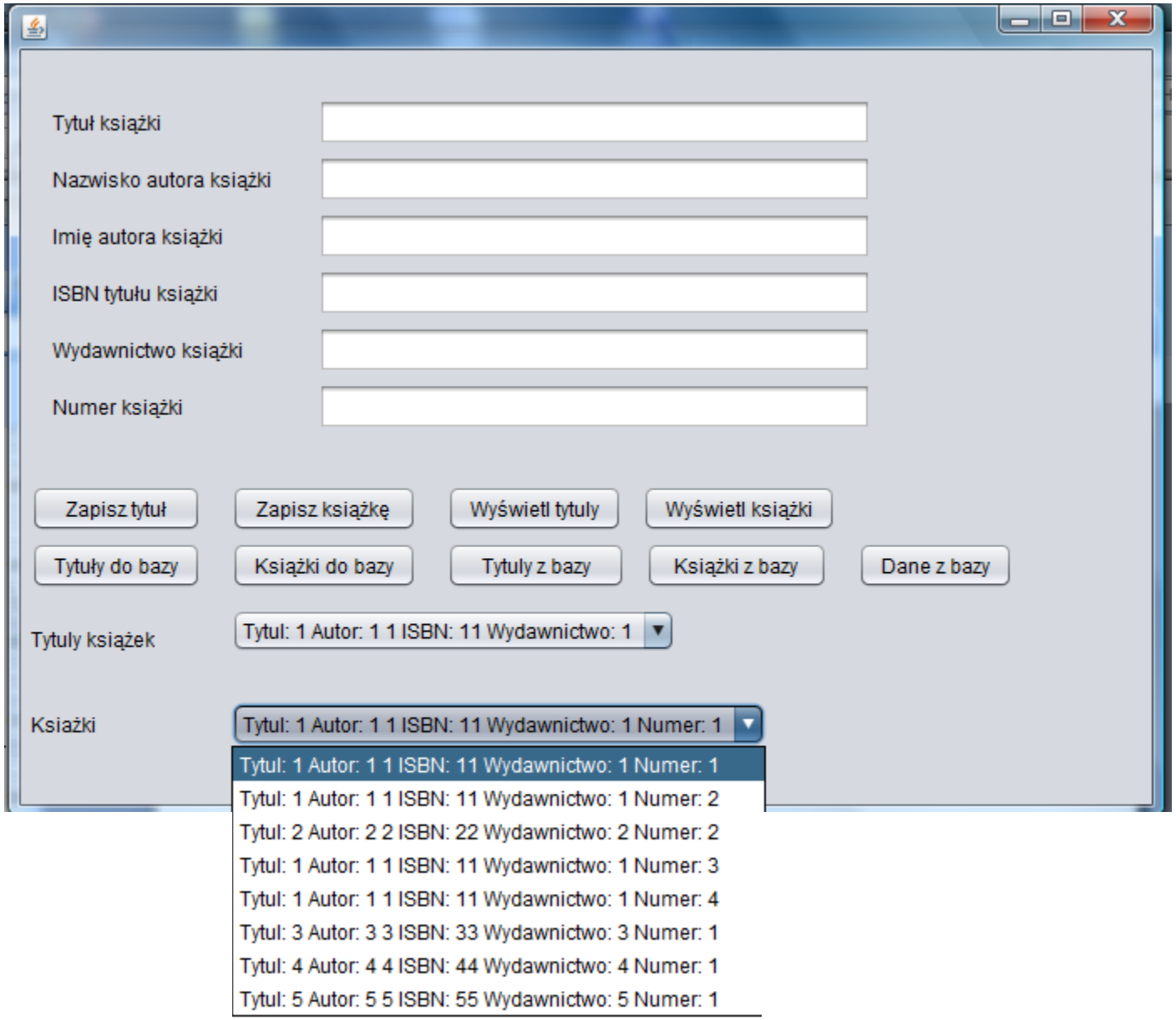
6.2. Wyświetlenie książek z aplikacji



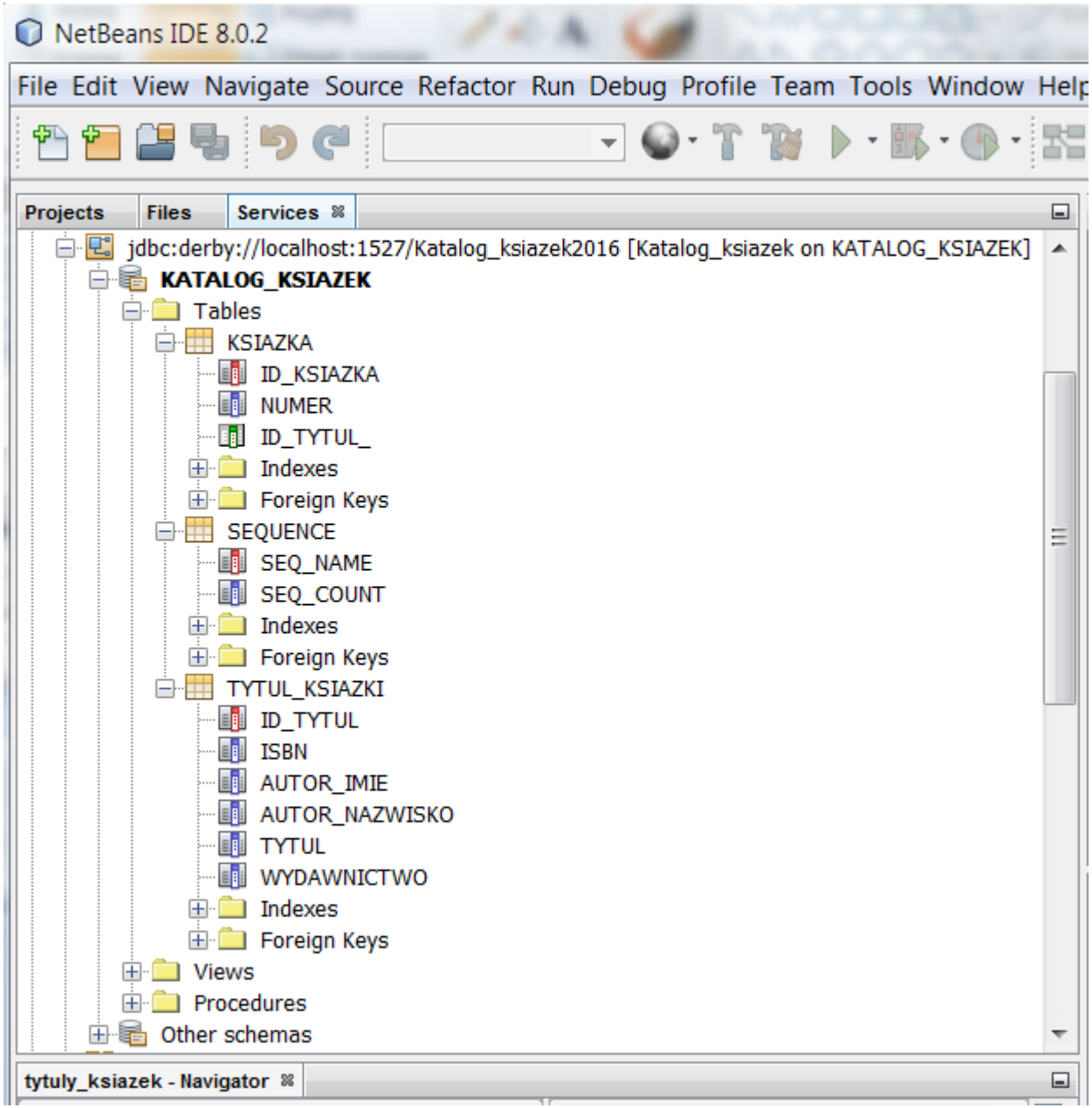
6.3. Wyświetlenie tytułów z bazy danych

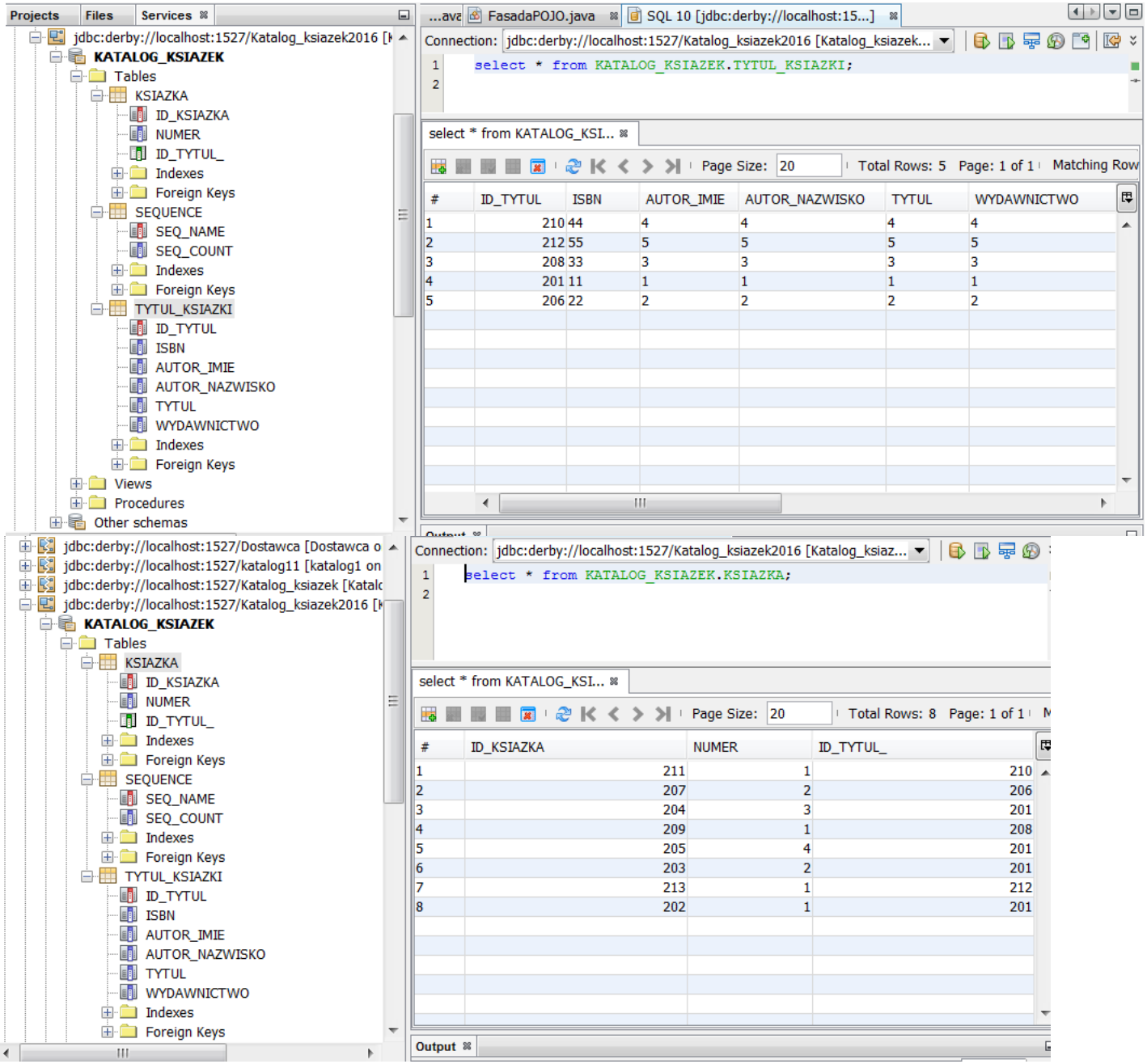


6.4. Wyświetlenie książek z bazy danych



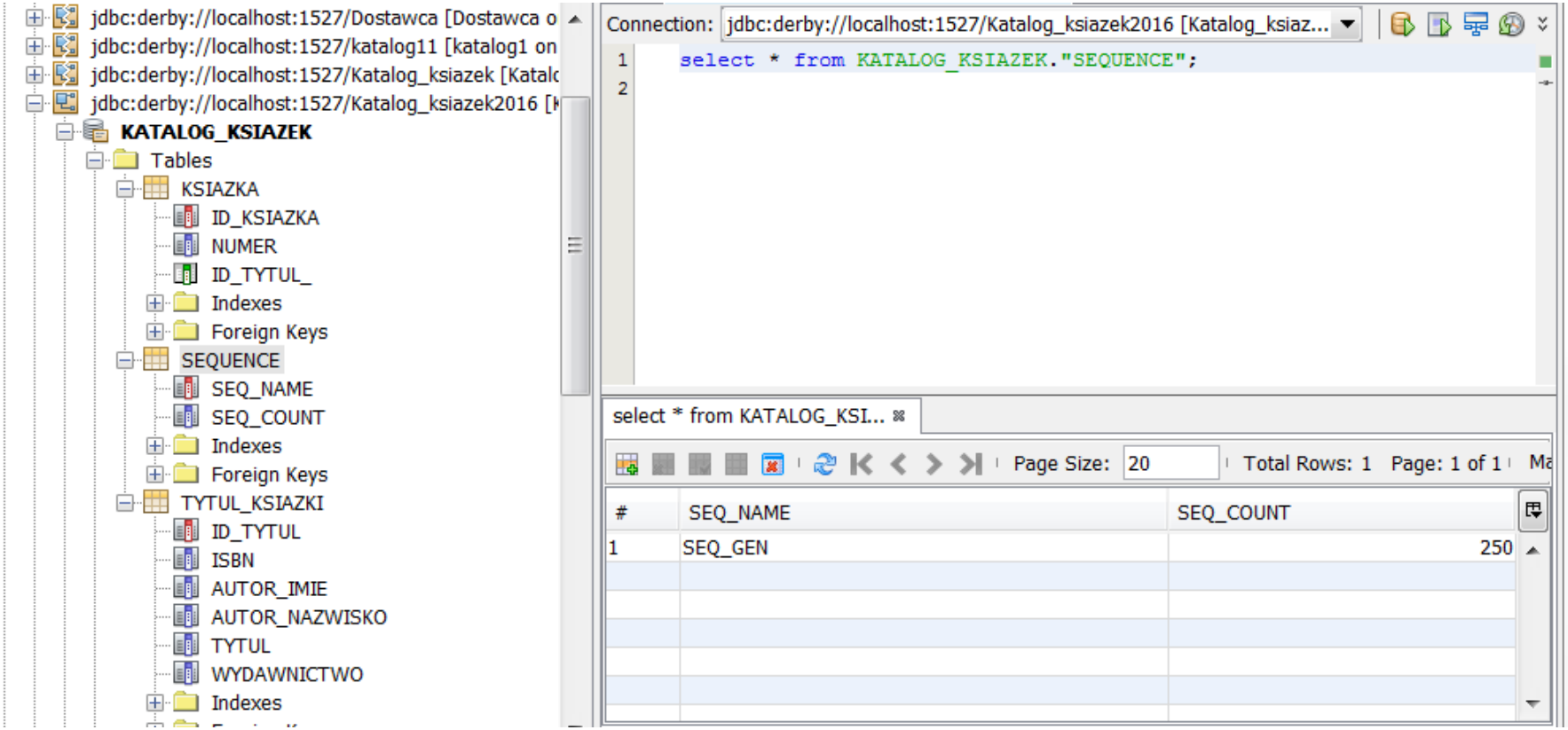
7. Widok struktury bazy danych, w której automatycznie zostały utworzone tabele





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



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