Utrwalanie danych – zastosowanie obiektowego modelu danych warstwy biznesowej do generowania schematu relacyjnej bazy danych (podrozdziały 1, 2, 3)
Tworzenie warstwy prezentacji – pierwszy etap (podrozdziały 4, 5)
Przykład

Autor
Zofia Kruczkiewicz

Wzorce oprogramowania - laboratorium5_6

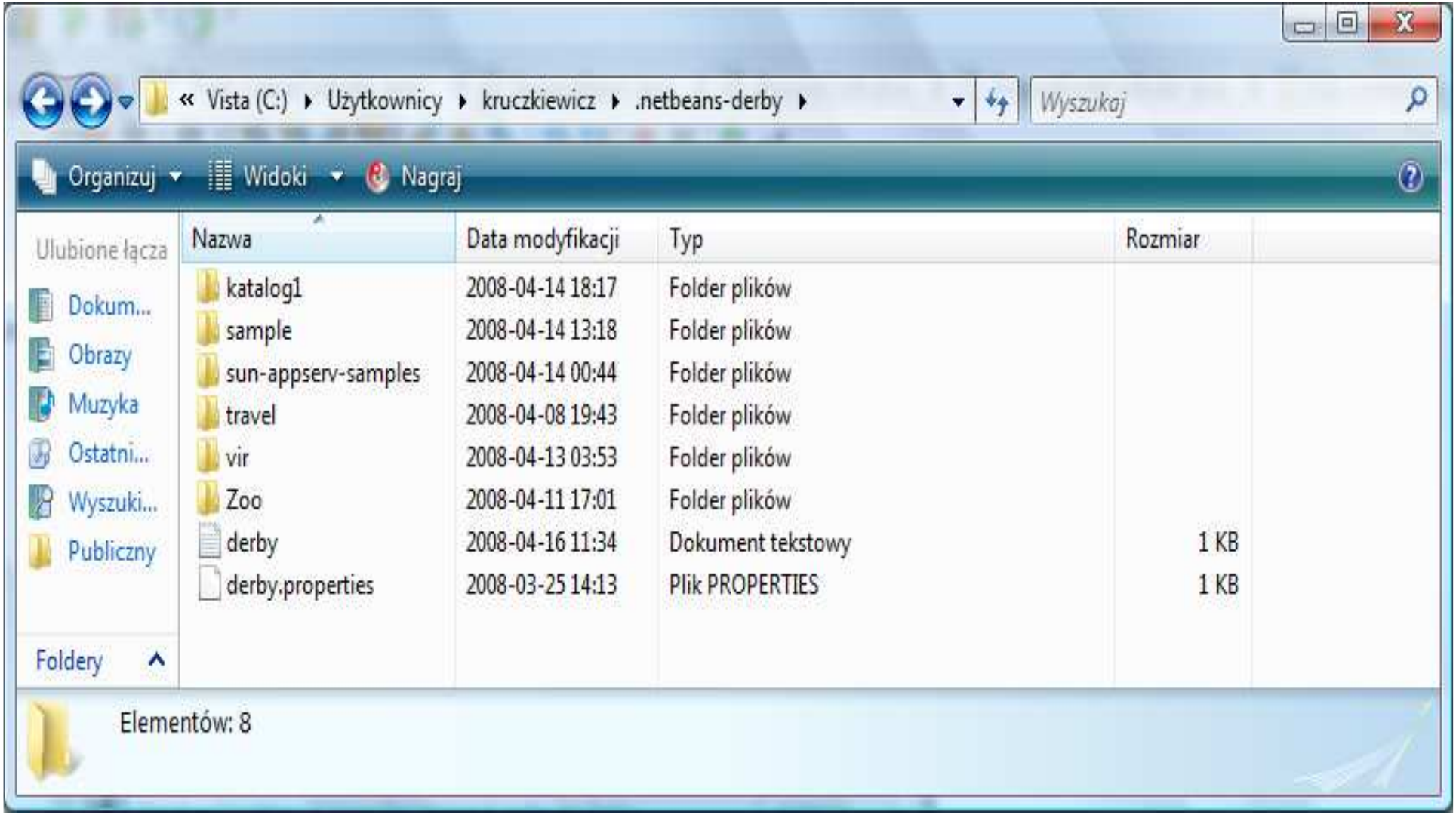
1. Tworzenie bazy danych w systemie baz danych Derby

Tworzenie bazy danych w systemie baz danych Derby

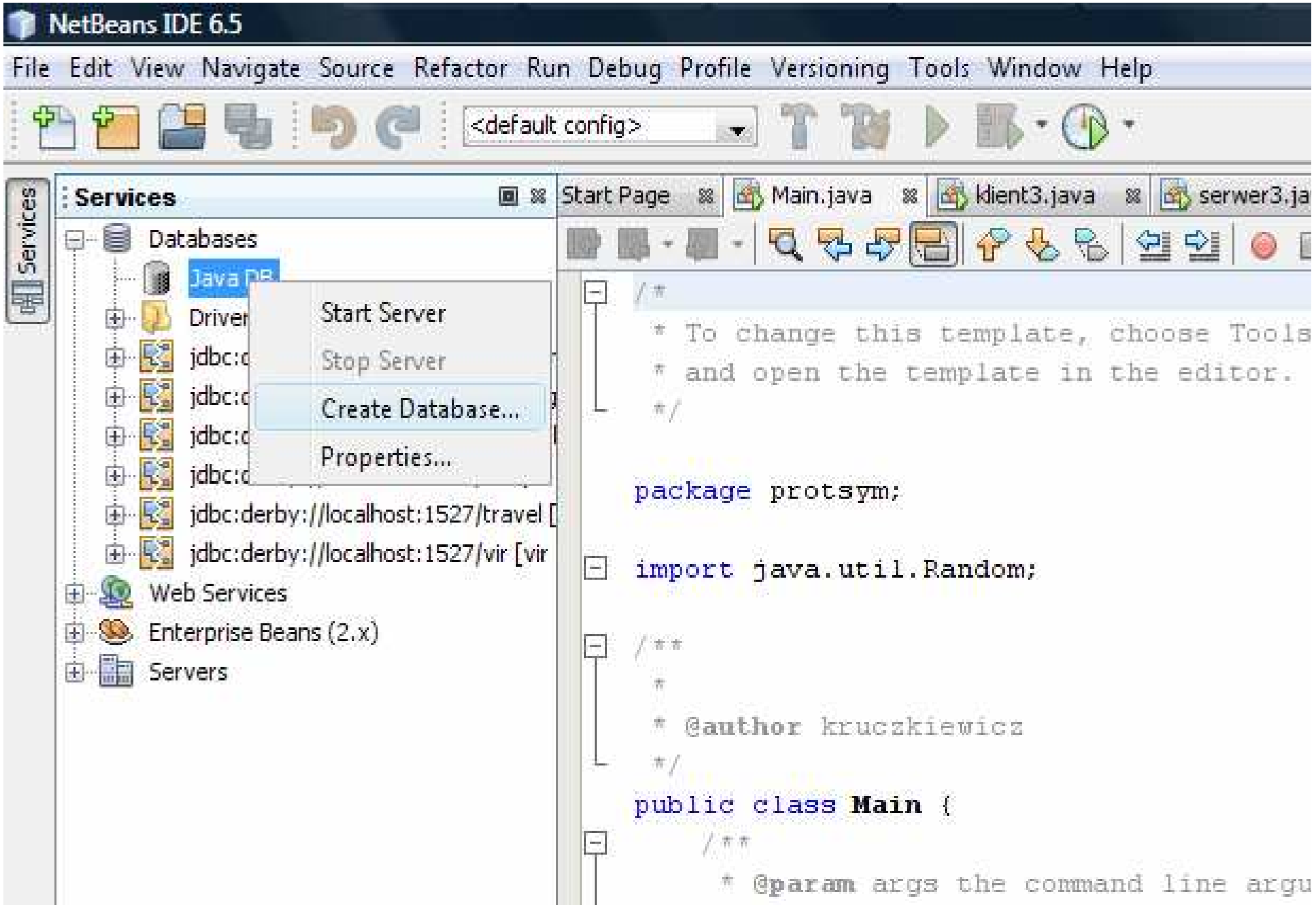
- Należy wybrać katalog na pustą bazę danych w systemie bazy danych Derby – domyślnym katalogiem jest katalog użytkowników systemu Windows \.netbeans-derby - slajd(1)
- Należy wybrać z zakładki **Services** środowiska NetBeans 6.5 opcję **Databases.** Następnie kliknąć prawym klawiszem myszy na **Java DB** i wybrać opcję **Create Database…** slajd (2)
- W formularzu tworzenia bazy danych wg wzoru podanego na slajdzie (3) wprowadzić dane dotyczące bazy danych: **Database Name, User, Password** oraz **Database Location** (domyślne lub wybrane przez użytkownika) – w wybranym katalogu pojawi się katalog o nazwie bazy danych z pustą bazą danych (slajd (4))
- W zakładce **Service** należy otworzyć opcję **Databases** (slajd (5)) i wybierając prawym klawiszem myszy pozycję z listy połączeń do baz danych typu

  jdbc:derby://localhost:1527/nazwa_bazy_danych:[użytkownik on schematbazydanych], należy nacisnąć **Connect**. Po chwili nastąpi połączenie z bazą danych – pustą – symboliczny kwadrat z lewej strony łańcucha połączenia ulegnie scaleniu. Przy połączeniu domyślnie zapamiętane jest hasło do bazy danych. Można to zmienić w okienku **Properties** (wybór po kliknięciu prawym klawiszem myszy na pozycję połączenia)
- Należy kliknąć na symbol [+] z lewej strony łańcucha połączenia – pojawi się zawartość katalogu z trzema pustymi podkatalogami: **Tables, Views, Procedures** (slajd (6))

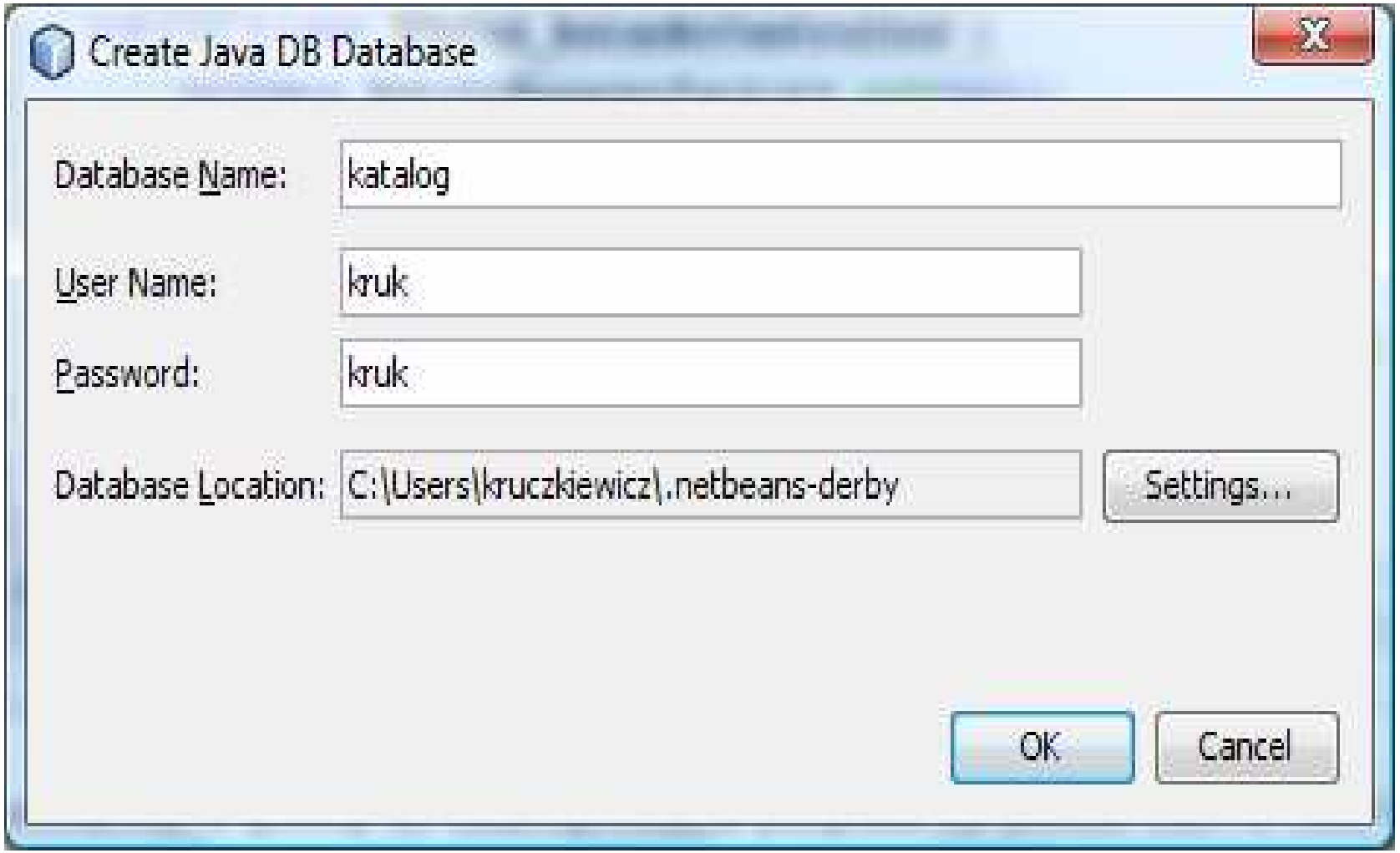
1.1.Zakładanie nowej bazy danych w domyślnym katalogu dla baz danych typu Derby (1)



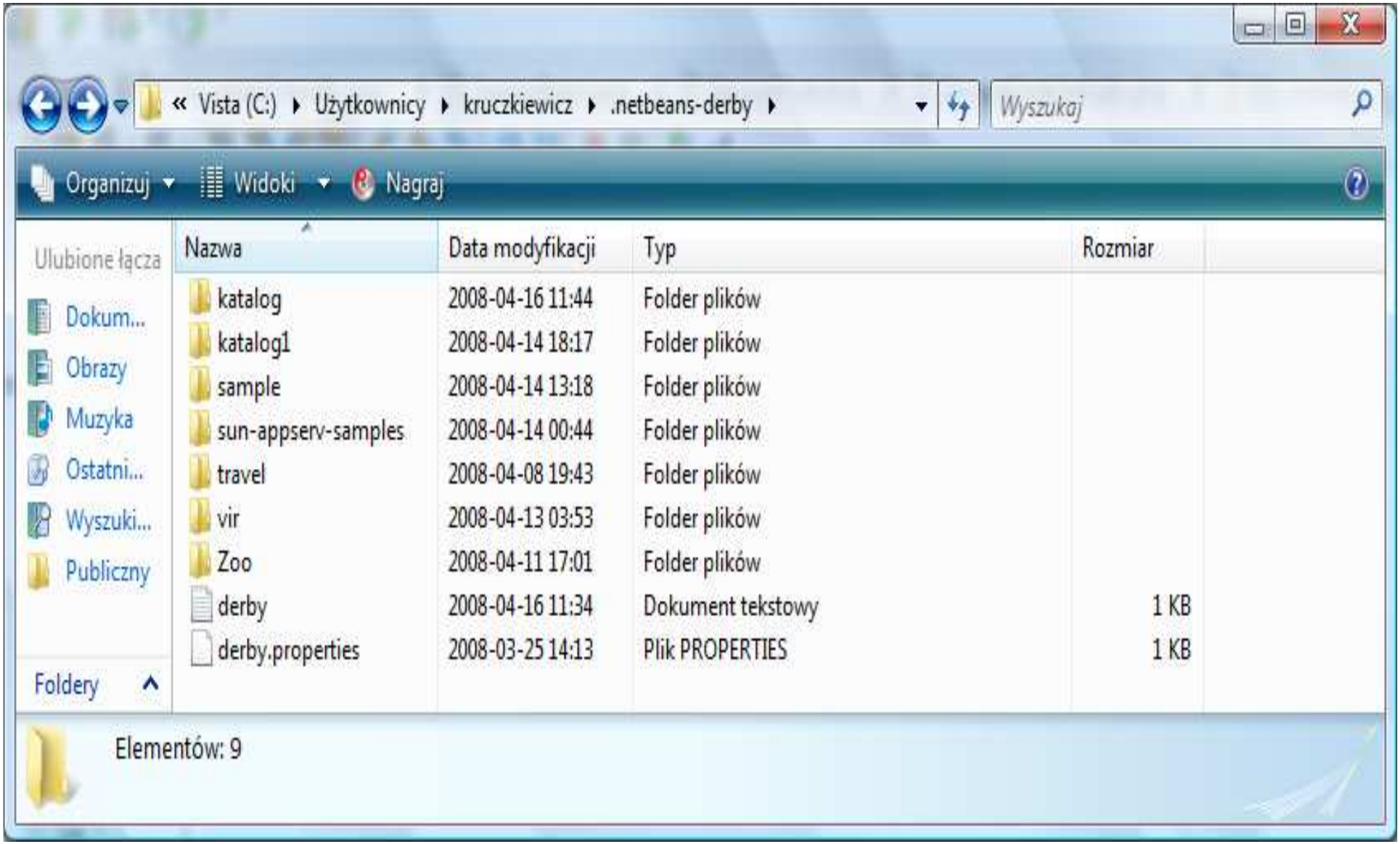
Zakładanie pustej bazy danych dla systemu baz danych Derby (2)



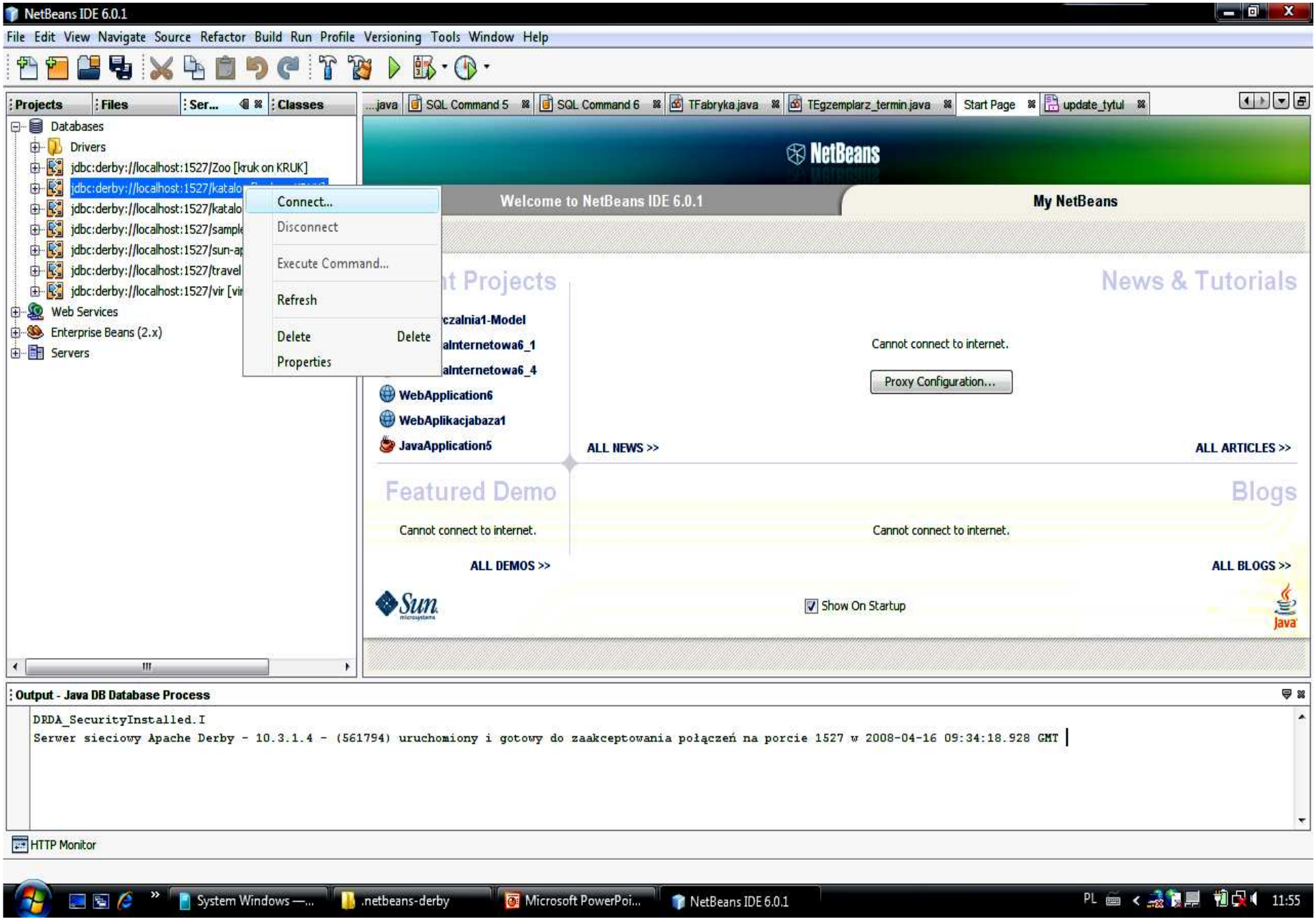
Zakładanie bazy danych **katalog** w systemie baz danych Derby (3)



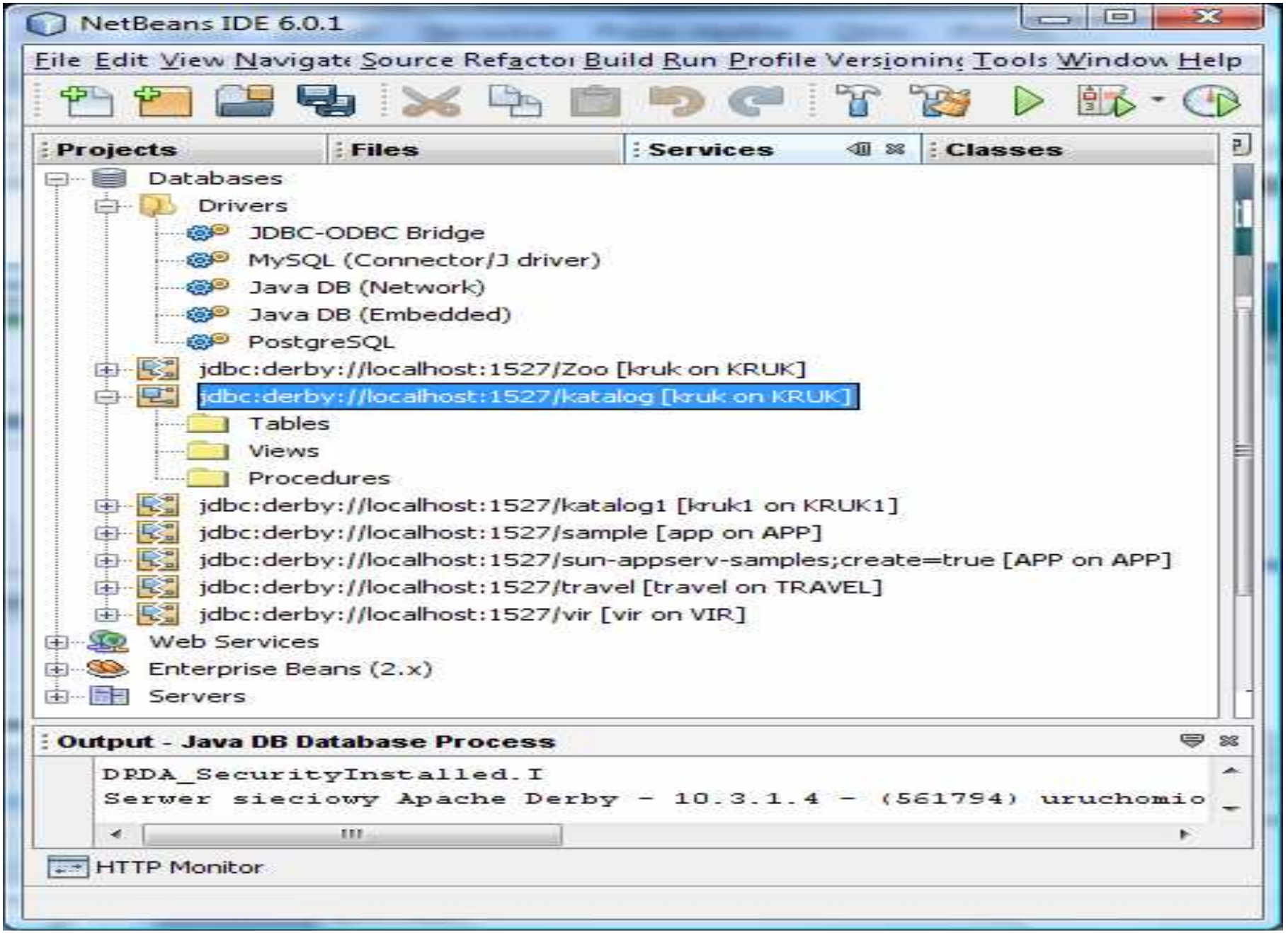
Założono nową bazę danych **katalog** w domyślnym katalogu dla baz danych typu Derby (4)



1.2. Łączenie z pustą bazą danych (5)



Połączenie z pustą bazą danych (6)



2. Tworzenie warstwy integracji w projekcje Java Application. Zastosowanie wzorców projektowych typu Domain Store i Transfer Object.

Przykład mapowania modelu obiektowego w środowisku NetBeans 6.01
za pomocą technologii TopLink (przykład z Laboratorium3_4).

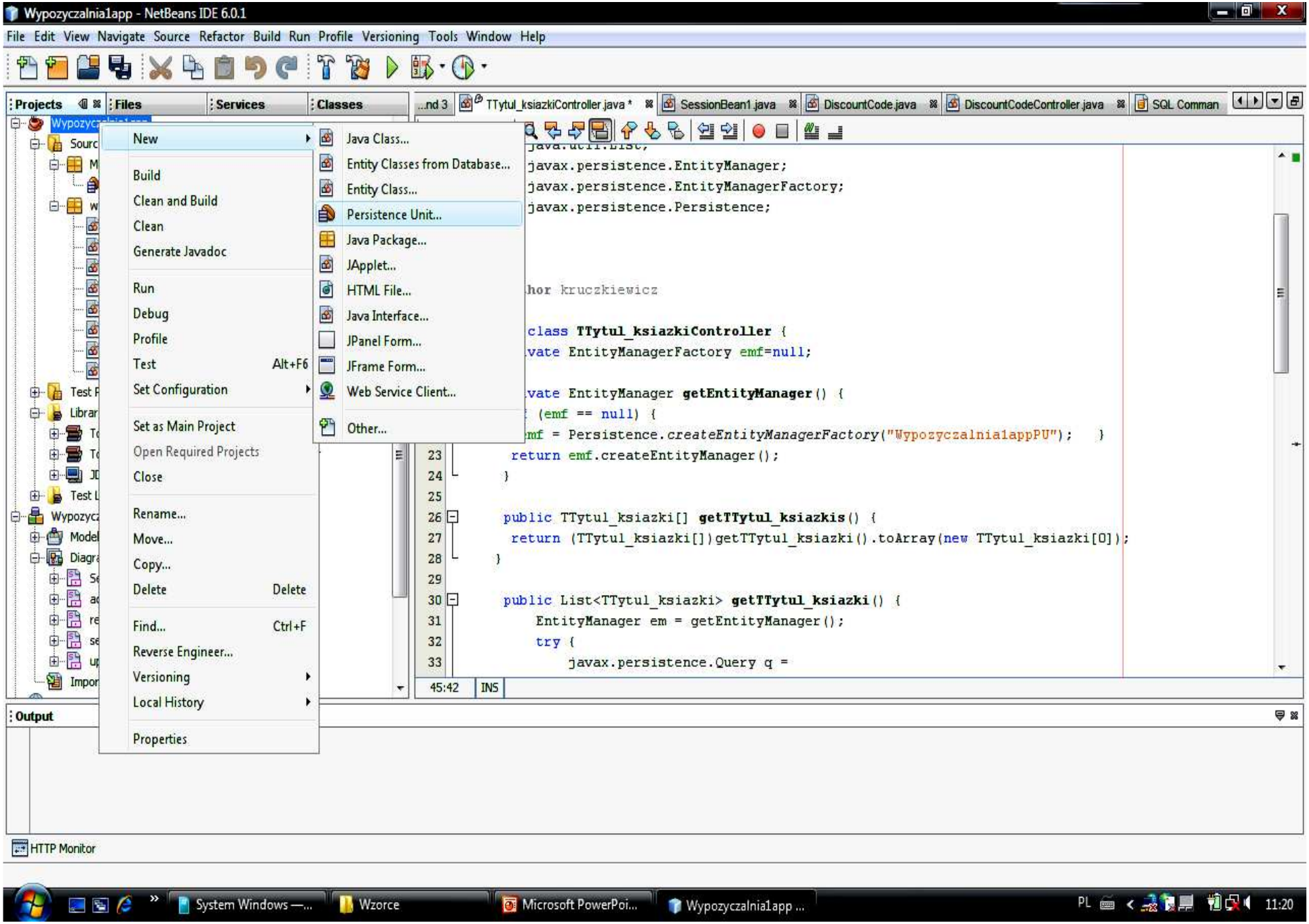
Wprowadzanie warstwy integracji i przekształcanie klas modelu obiektowego na encje (1)

- Należy wstawić moduł typu **Persistence Unit** - prawym klawiszem kliknąć na nazwę projektu w zakładce **Projects**, wybrać **New\ Persistence Unit** (slajd 1). Jeśli w liście nie ma pozycji **Persistence Unit**, należy wybrać pozycję **Other**, wybrać w formularzu **New File** w lewym oknie **Categories** pozycję **Persistence** i w prawym oknie **FileTypes** typ pliku **Persistence Unit.**
- Po naciśnięciu **Next** należy ustalić nazwę modułu. Nazwa modułu wynika z nazwy projektu z przyrostkiem PU. Można zmienić tę nazwę. Należy wybrać **Persistence Library** typu **TopLink** oraz połączyć się z pustą bazą danych założoną w pierwszym etapie prac wybierając właściwy łańcuch połączenia z listy **Data Source.**
- W formularzu tworzenia modułu należy wybrać **Strategy Generation Table** typu **Create** (slajd (3)) – jeśli tabel nie ma, zostaną utworzone, w przeciwnym wypadku nie będą tworzone nowe tabele (sytuacja sygnalizowana wyjątkami). W katalogu projektu **Source Packages** – **META-INF** pojawia się plik **persitence.xml** i w katalogu **Libraries** pojawiają się biblioteki technologii TopLink w plikach jar – zaznaczono je na slajdzie (4)
- Na slajdzie (4) pokazano docelową zawartość projektu typu **Java Application** – należy teraz utworzyć klasy typu **Controller** dla każdej z utrwalanych klas. W przypadku dziedziczenia należy utworzyć te klasy dla klas bazowych. Klasy typu **Controller** są fasadami warstwy integracji dla każdej z utrwalanych klas.
- W celu utworzenia każdego z plików **TTytul_ksiazkiController.java** oraz **TEgzemplarzController.java,** należy prawym klawiszem kliknąć na nazwę projektu w zakładce **Projects**, wybrać **New\Java Class**. Jeśli w liście nie ma pozycji **Java Class**, należy wybrać pozycję **Other**, wybrać w formularzu **New File** w lewym oknie **Categories** pozycję **Java** i w prawym oknie **FileTypes** typ pliku **Java Class.** Po naciśnięciu **Next** należy podać nazwę pliku w pozycji **Class Name** (**TTytul_ksiazkiController**) i wybrać z listy **Package** pakiet, w którym znajdują się klasy warstwy biznesowej (TTytul_ksiazki.java itd..) np. wypozyczalnia1app. Nacisnąć **Finish**. Całą procedurę powtórzyć do utworzenia pliku TEgzemplarzController.java. Po zakończeniu tych czynności powstają pliki zawierające klasy typu public z pustym ciałem.

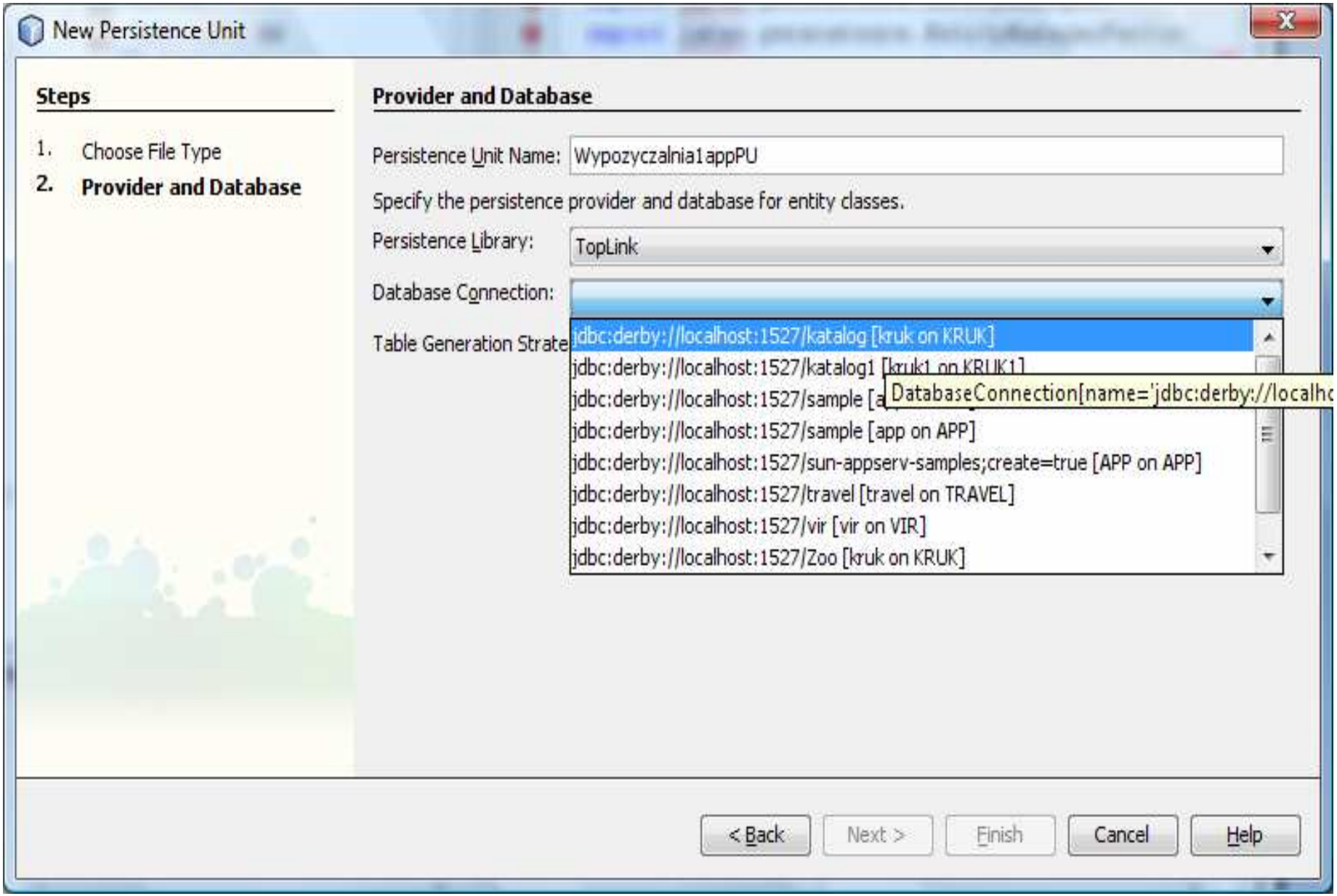
Wprowadzanie warstwy integracji i przekształcanie klas modelu obiektowego na encje (2)

- Na slajdach 5-13 pokazano przebieg tworzenia kodu dla klasy typu **Controller** dla **TTytul_ksiazki**. Należy napisać taki kod zgodnie ze wskazówkami również dla klasy **TEgzemplarz**. **Podczas pisania metody getEntityManager() łańcuch w liście parametrów metody createEntityManagerFactory powinien być nazwą modułu Persistence Unit**. Oba pliki mogą znajdować się w katalogu klas modelu biznesowego lub nowym pakiecie tego projektu. W celu ułatwienia czynności po zakończeniu tworzenia ciała klasy **TTytul_ksiazkiController** można skopiować je i wkleić do ciała klasy **TEgzemplarzController**, zamienić ciąg znaków **TTytul_ksiazki** na **TEgzemplarz** za pomocą **CTRL+H**. Podczas tworzenia kodu należy uzupełniać import pakietów za pomocą opcji **Fix Imports** (prawym klawiszem myszy należy klikać na ciało klasy i wybrać z wyskakującego menu **Fix Imports** i wybierać klasy z pakietu **javax.persistence**. Dla typu **List** wybrać pakiet **java.util**.). Po tych czynnościach jedynie linie z wywoływaną metodą **find** są niekompilowane. Sytuacja ta zmieni się po przekształceniu klas modelu obiektowego na rodzaj **Entity**.
- Na slajdach 14 do 15 (wyłączając 15) podano wskazówki dotyczące zamiany klas **TTytul_ksiazki, TTytul_ksiazki_na_kasecie, TEgzemplarz, TEgzemplarz_termin** na klasy z adnotacjami @Entity. W klasie **TEgzemplarz** wprowadzono wirtualne metody: **public Date getTermin()** i **public void setTermin(Date termin)**. Podczas tworzenia kodu należy uzupełniać import pakietów za pomocą opcji **Fix Imports** (prawym klawiszem myszy należy klikać na ciało klasy i wybrać z wyskakującego menu Fix Imports i wybierać klasy z pakietu **javax.persistence**).
- Zgodnie ze slajdem 15, należy w zakładce **Projects** wybrać w projekcie zakładkę META-INF i otworzyć plik **persistence.xml** w trybie **Design**. W formularzu pliku XML należy za pomocą klawisza **Add class** wstawić z listy utworzone cztery klasy typu **Entity**.
- Należy skompilować uzupełniany projekt – tutaj Wypozyczalnia1app

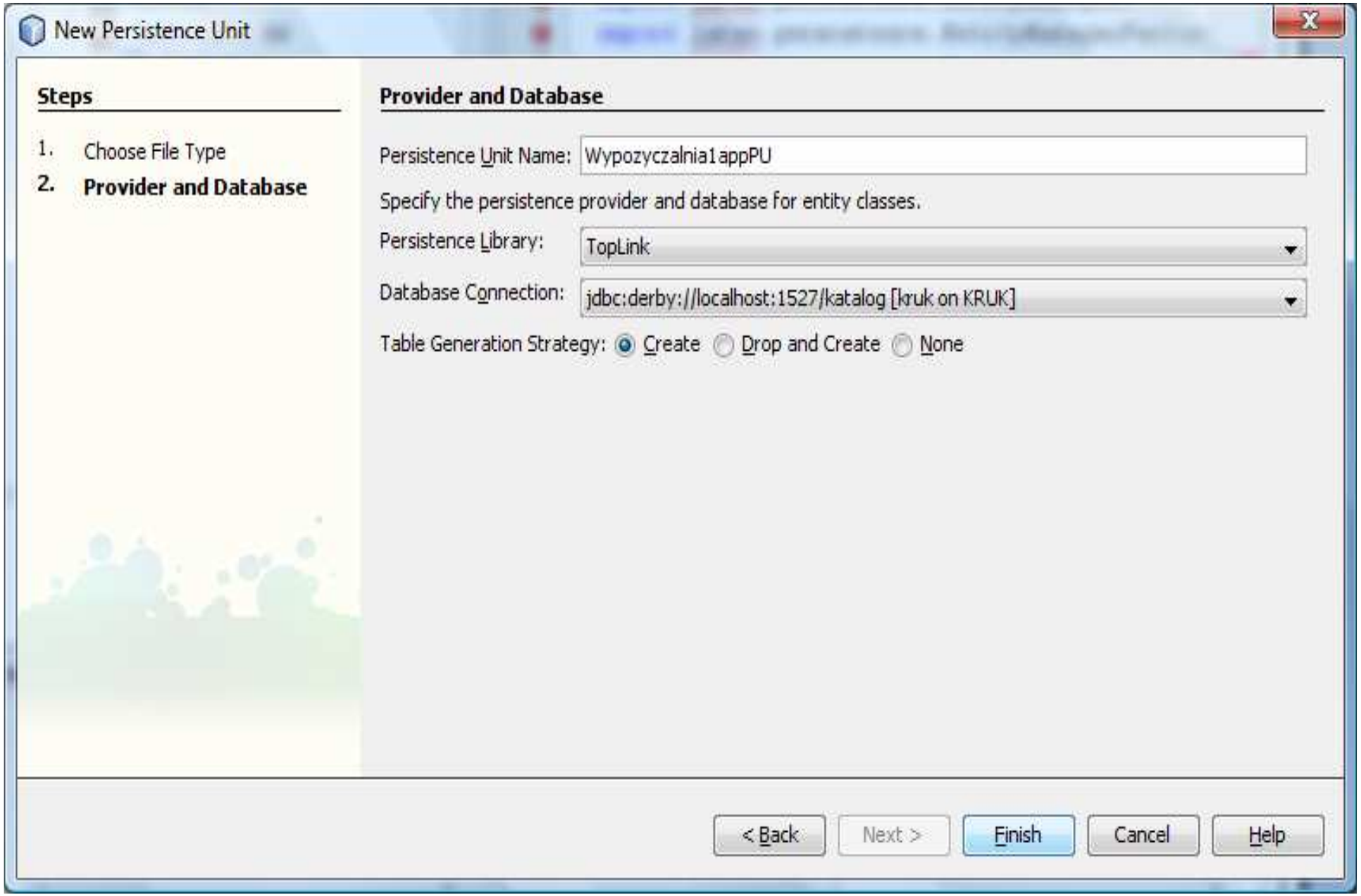
2.1. Wstawianie do projektu typu **Java Application** modułu typu **Persistence Unit (1)**



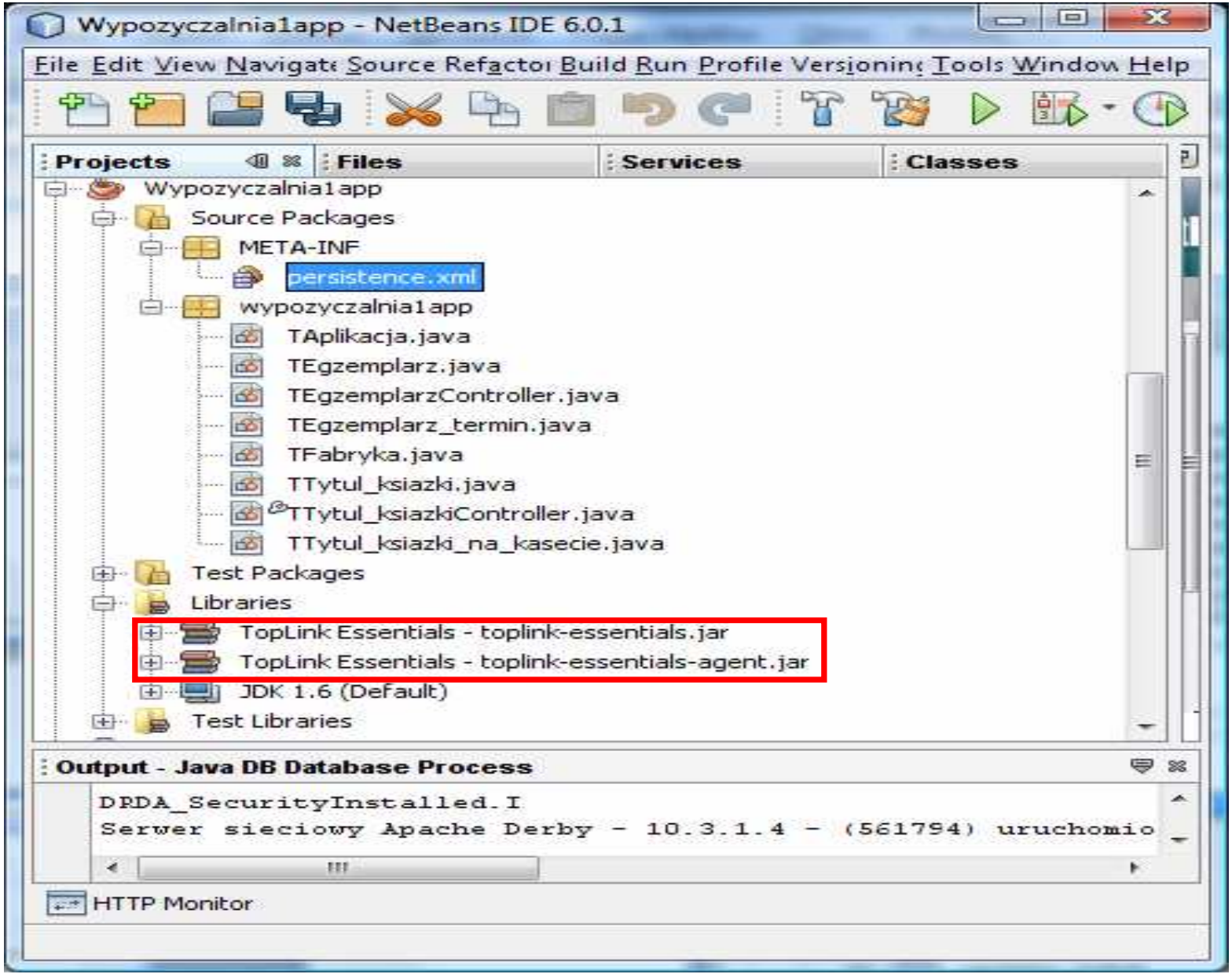
Wybór bazy danych, w której będą utrwalane obiekty – pustej (2)



Tworzenie modułu utrwalania danych dla technologii TopLink (3)



Plik **persistence.xml** reprezentujący moduł utrwalania danych typu **TopLink**
Uzupełnianie zawartości projektu typu **Java Application** o klasy typu **Controller** dla każdej z utrwalanych klas modelu obiektowego (4)



2.2. Inżynieria odwrotna dla utworzonych przez programistę klas typu **Controller** dla każdej z utrwalanych klas modelu obiektowego (**TTytul_ksiazki**) (5)

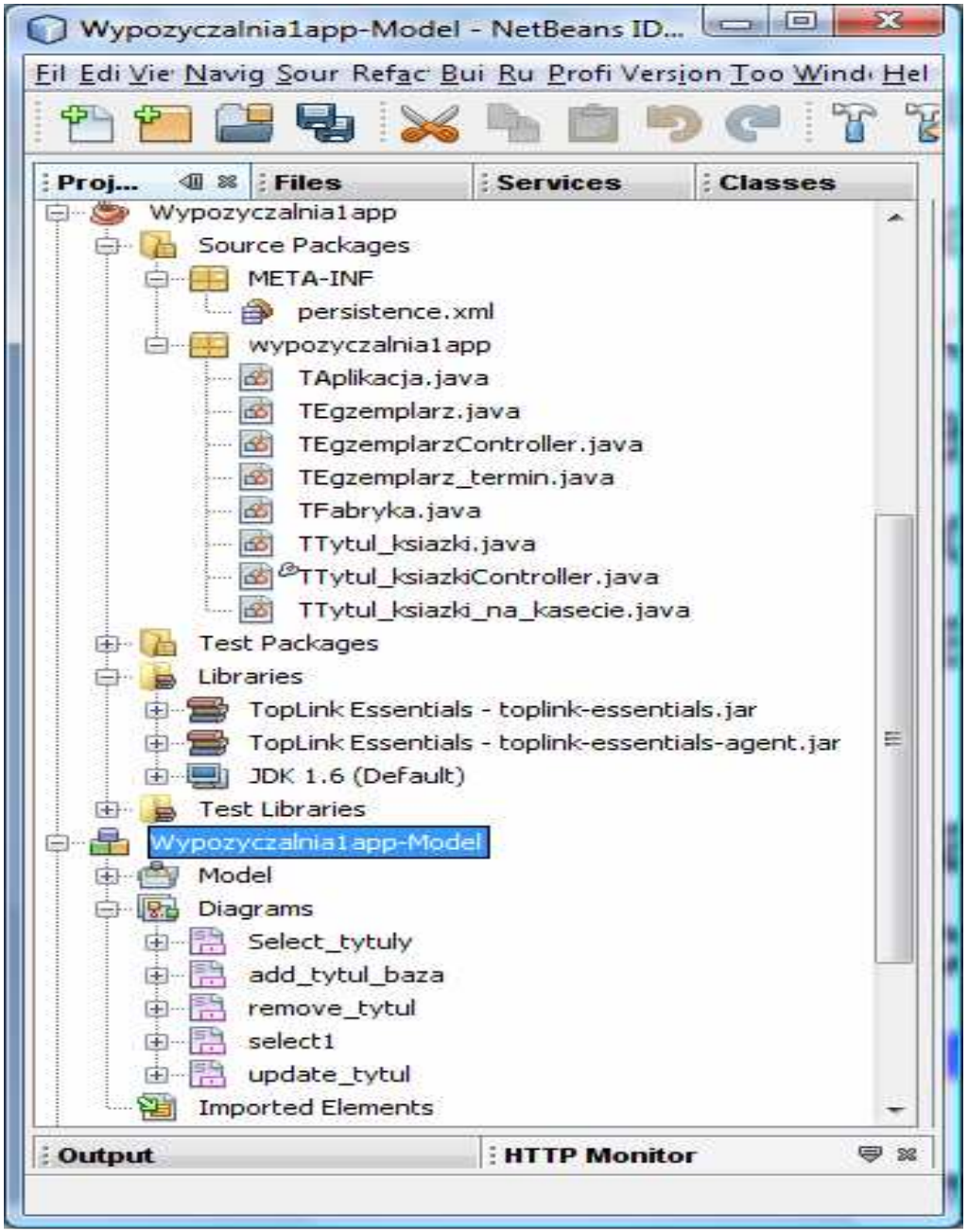
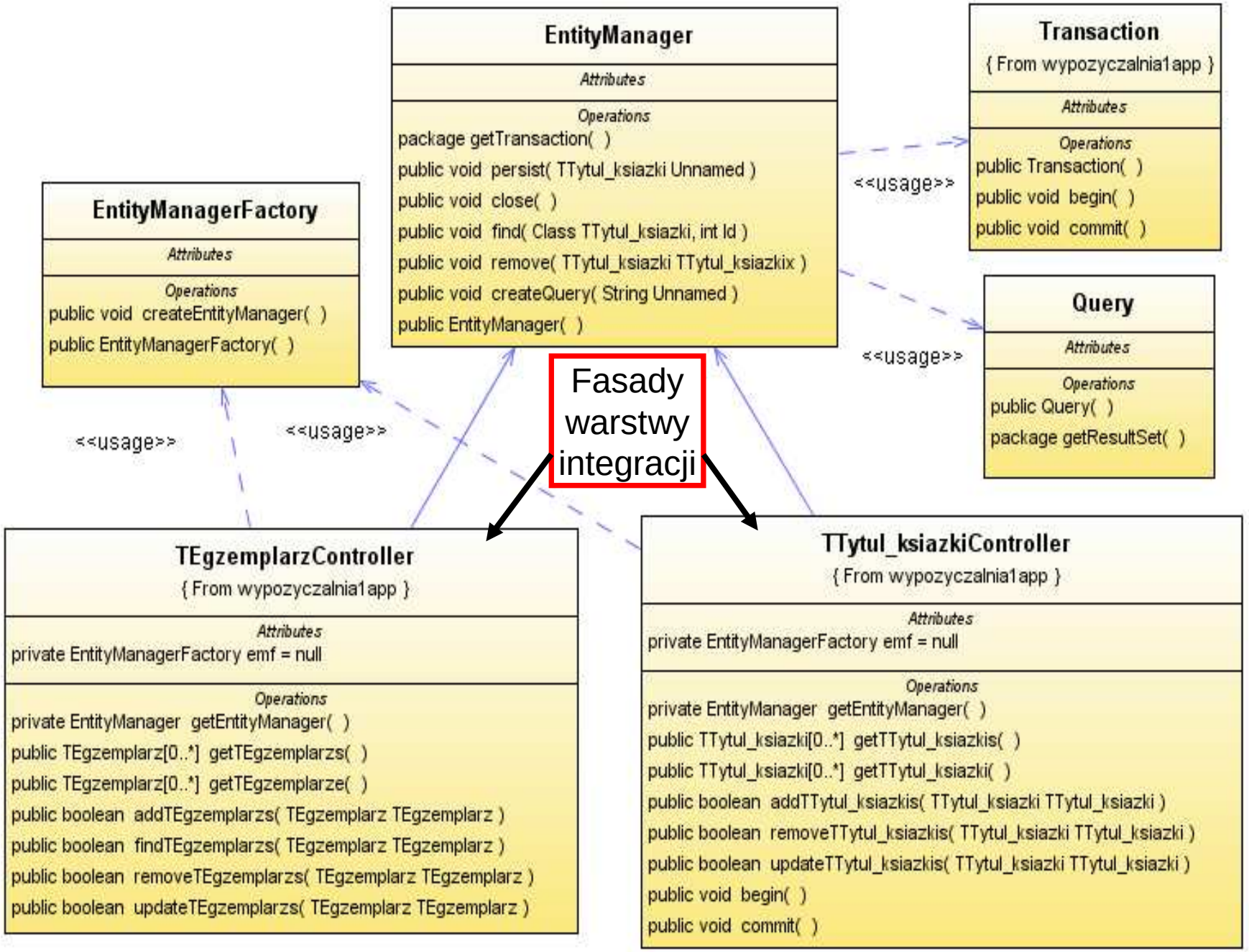
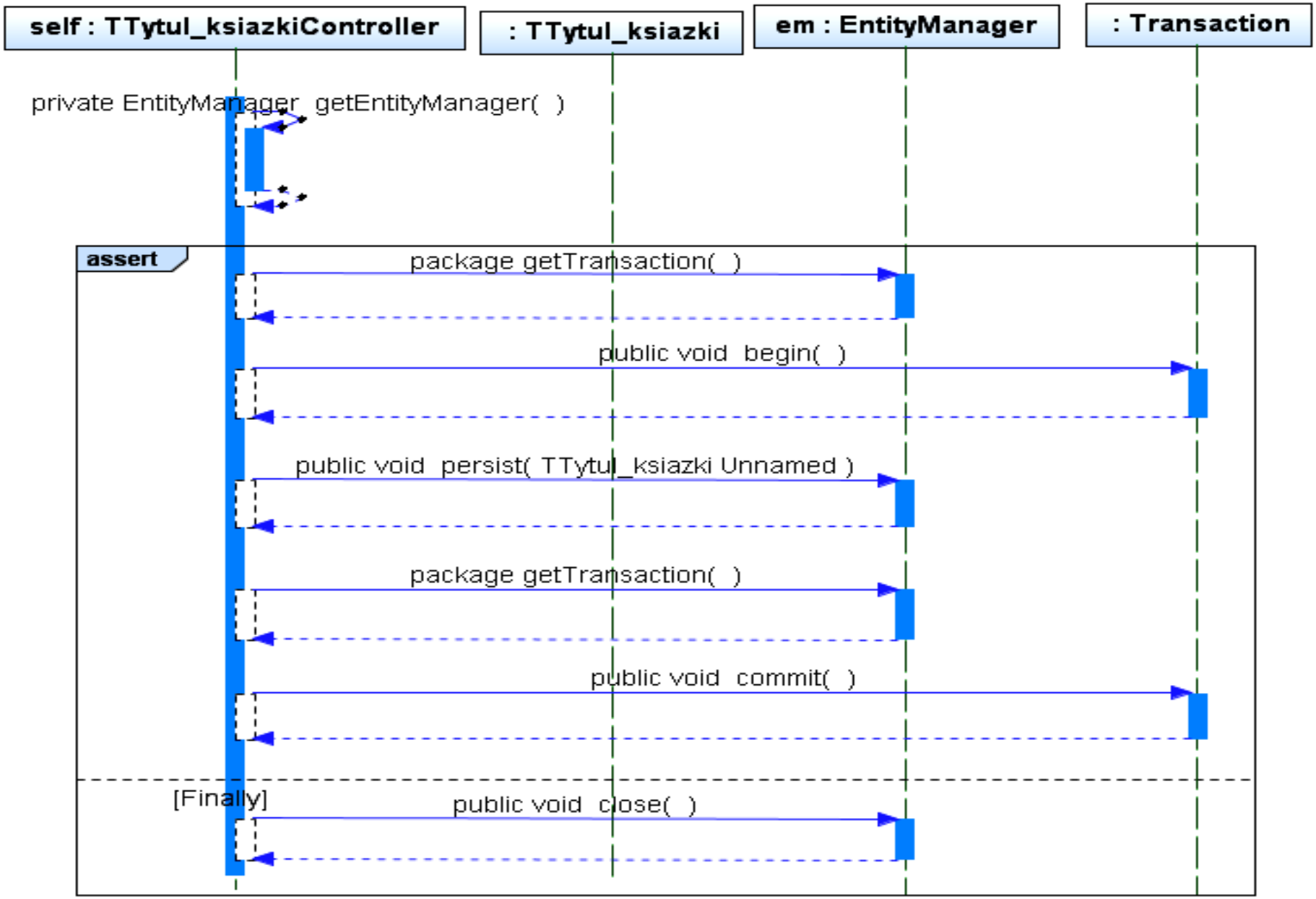


Diagram klas – uproszczony schemat warstwy integracji (6)



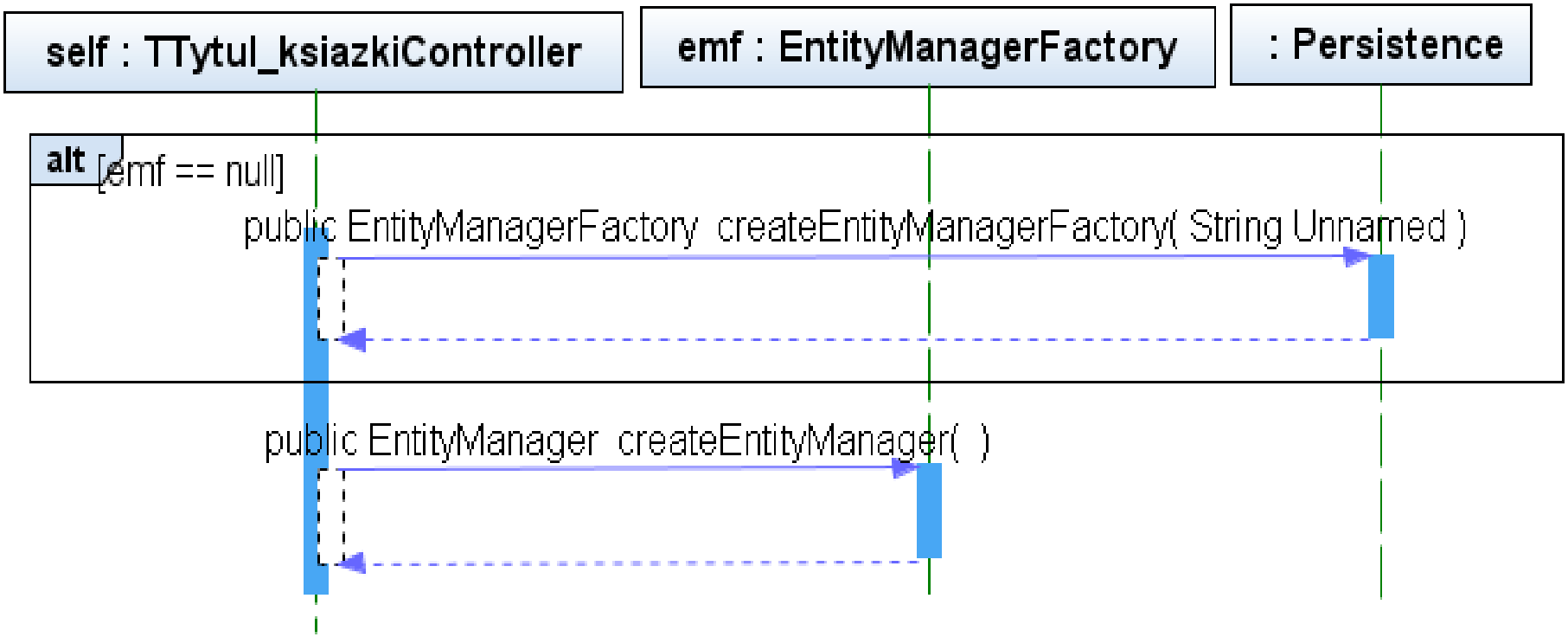
Dodawanie tytułów do bazy danych -

public boolean addTTytul_ksiazkis(TTytul_ksiazki TTytul_ksiazki) (7)



Dodawanie tytułów do bazy danych -

private EntityManager getEntityManager() (8)



Klasa typu Controller dla każdej z klas utrwalanych obiektów – (9)
Realizacja wzorców typu Domain Store i Transfer Object

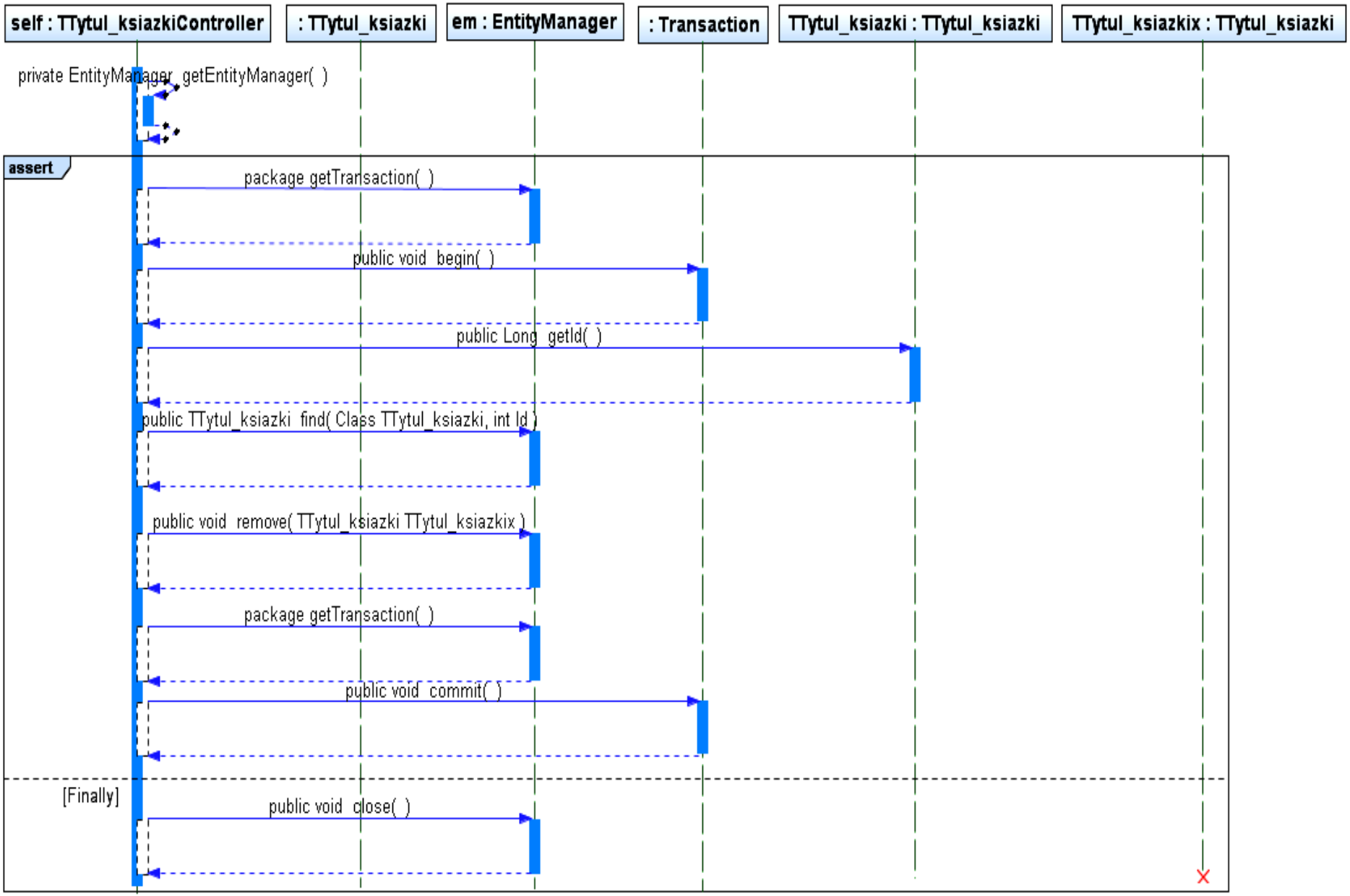
```
public class TTytul_ksiazkiController {
   private EntityManagerFactory emf=null;

    private EntityManager getEntityManager() {
     if (emf == null) {
      emf = Persistence.createEntityManagerFactory(
                                          "Wypozyczalnia1appPU");
         }
     return emf.createEntityManager();
    }
```

```
public boolean addTTytul_ksiazkis(
                        TTytul_ksiazki TTytul_ksiazki)  {
    EntityManager em = getEntityManager();
    try {
        em.getTransaction().begin();
        em.persist(TTytul_ksiazki);
        em.getTransaction().commit();
    } finally {
        em.close();
        return false; }
  }
```

Usuwanie tytułów z bazy danych (10)

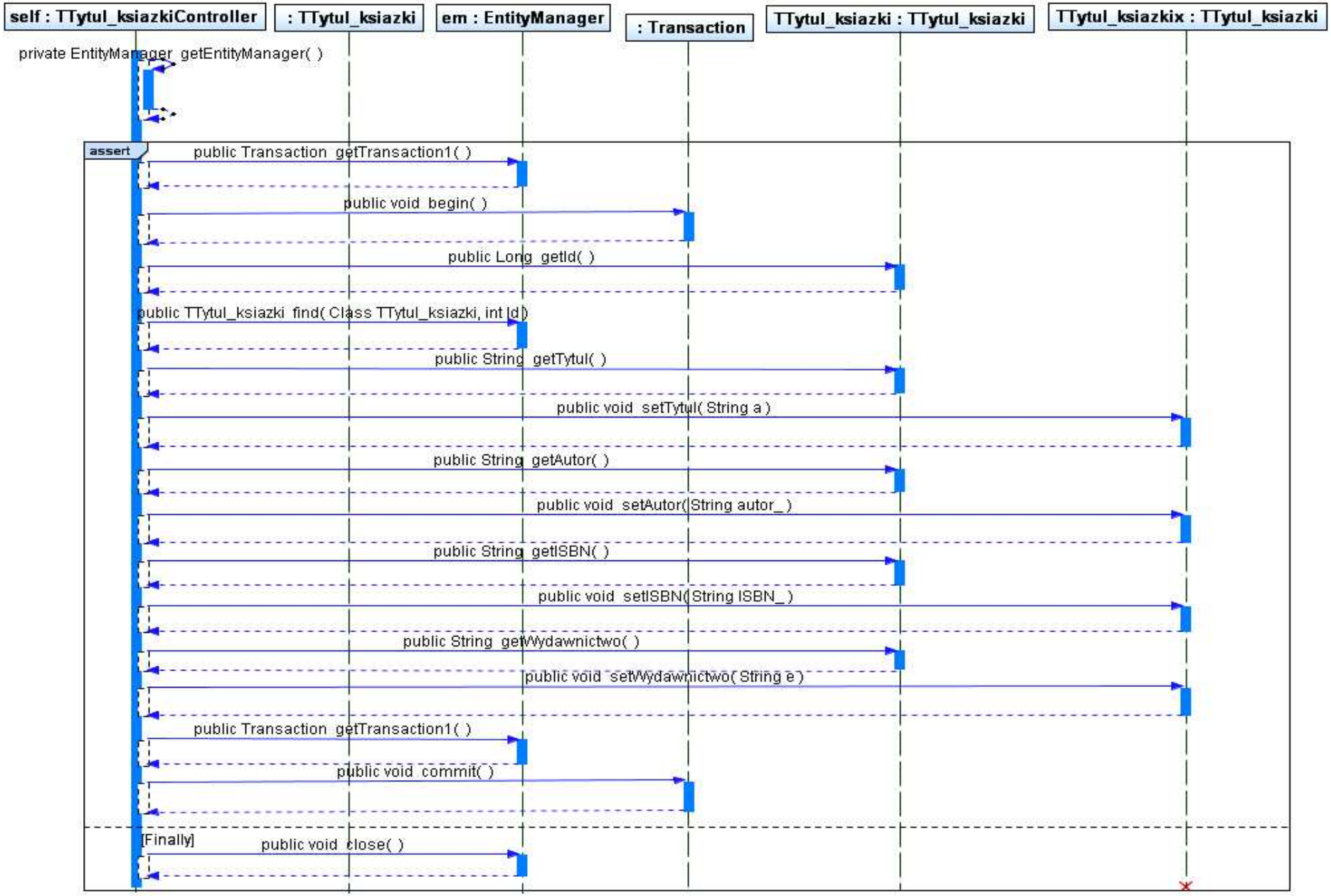
public boolean removeTTytul_ksiazkis(TTytul_ksiazki TTytul_ksiazki)



```
public boolean removeTTytul_ksiazkis(
                    TTytul_ksiazki TTytul_ksiazki) {
    EntityManager em = getEntityManager();
    try {
        em.getTransaction().begin();
        TTytul_ksiazki TTytul_ksiazkix =
                        em.find(TTytul_ksiazki.class,
                                TTytul_ksiazki.getId());
        em.remove(TTytul_ksiazkix);
        em.getTransaction().commit();
      } finally {
         em.close();
         return false; }
    }
```

Modyfikacja danych w bazie danych (11)

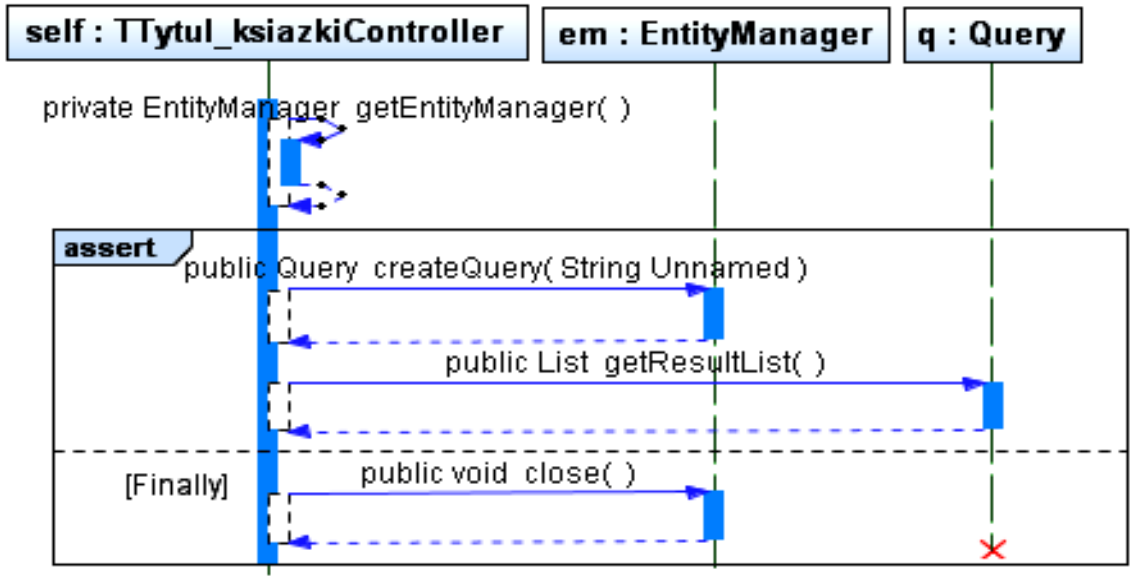
public boolean updateTTytul_ksiazkis(TTytul_ksiazki TTytul_ksiazki)



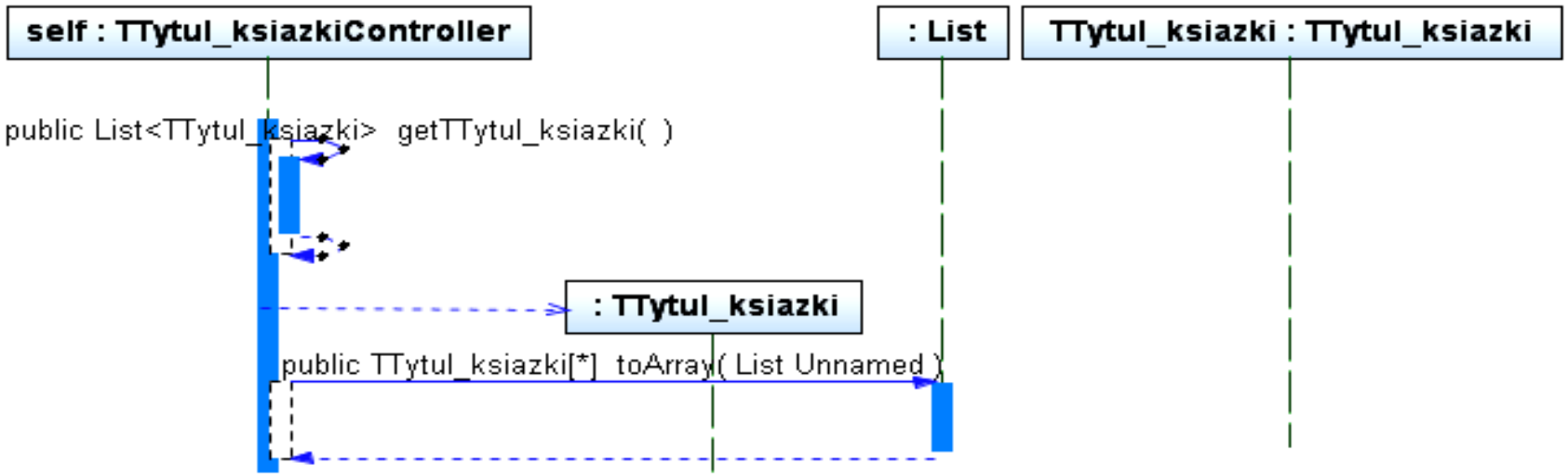
```
public boolean updateTTytul_ksiazkis(TTytul_ksiazki TTytul_ksiazki)
{  EntityManager em = getEntityManager();
   try
   {
    em.getTransaction().begin();
    TTytul_ksiazki TTytul_ksiazkix =
          em.find(TTytul_ksiazki.class, TTytul_ksiazki.getId());
    TTytul_ksiazkix.setTytul(TTytul_ksiazki.getTytul());
    TTytul_ksiazkix.setAutor(TTytul_ksiazki.getAutor());
    TTytul_ksiazkix.setISBN(TTytul_ksiazki.getISBN());
    TTytul_ksiazkix.setWydawnictwo(TTytul_ksiazki.getWydawnictwo());
    em.getTransaction().commit();
    }
     finally
      {
        em.close();
        return false; }
    }
```

Odczytanie tytułów z bazy danych w postaci listy i tablicy TTytul_ksiazki (12)
(wzorzec typu Transfer Object – TTytul_ksiazki[0..*])

public List<TTytul_ksiazki> getTTytul_ksiazki()



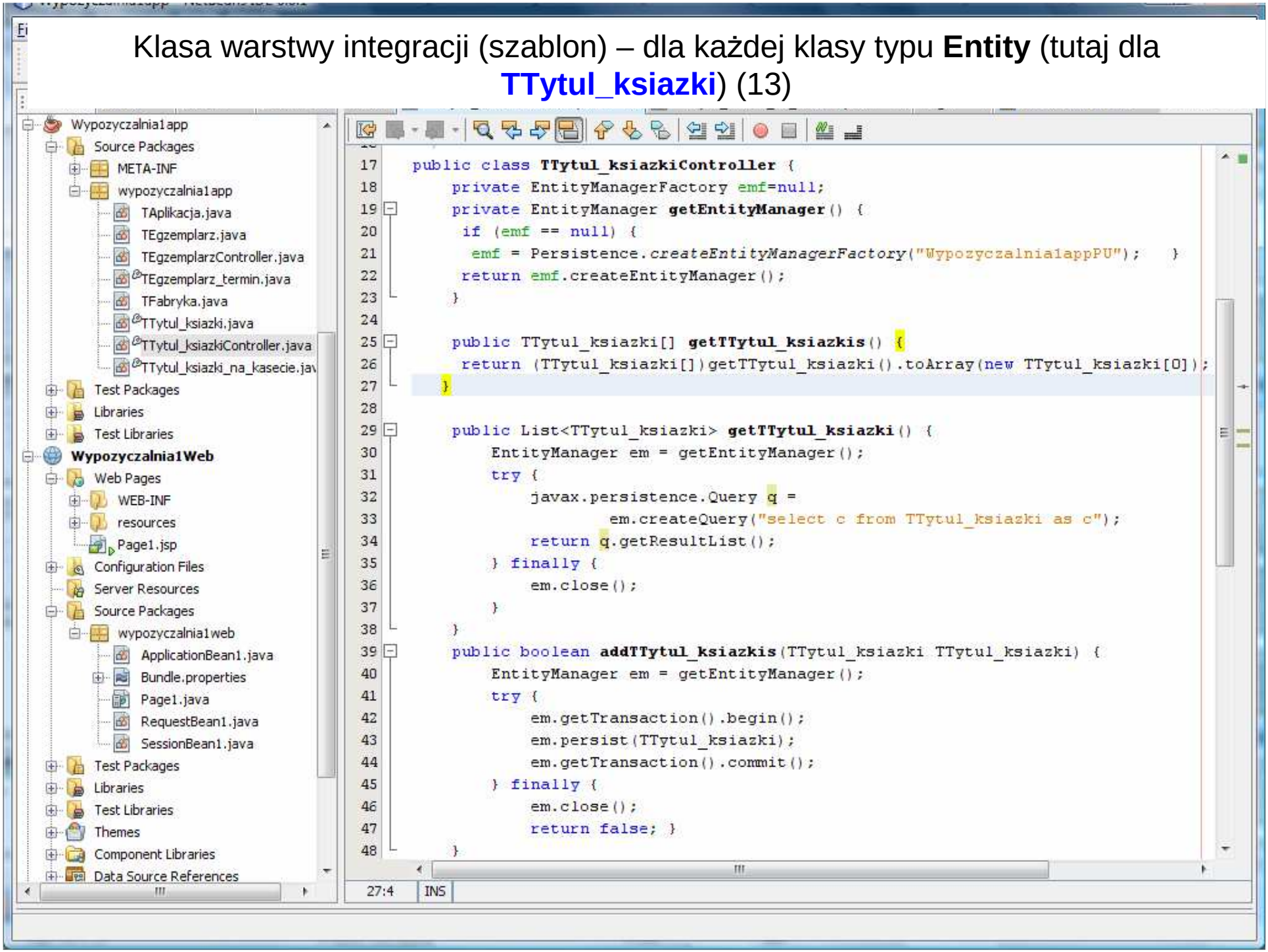
public TTytul_ksiazki[] getTTytul_ksiazkis()

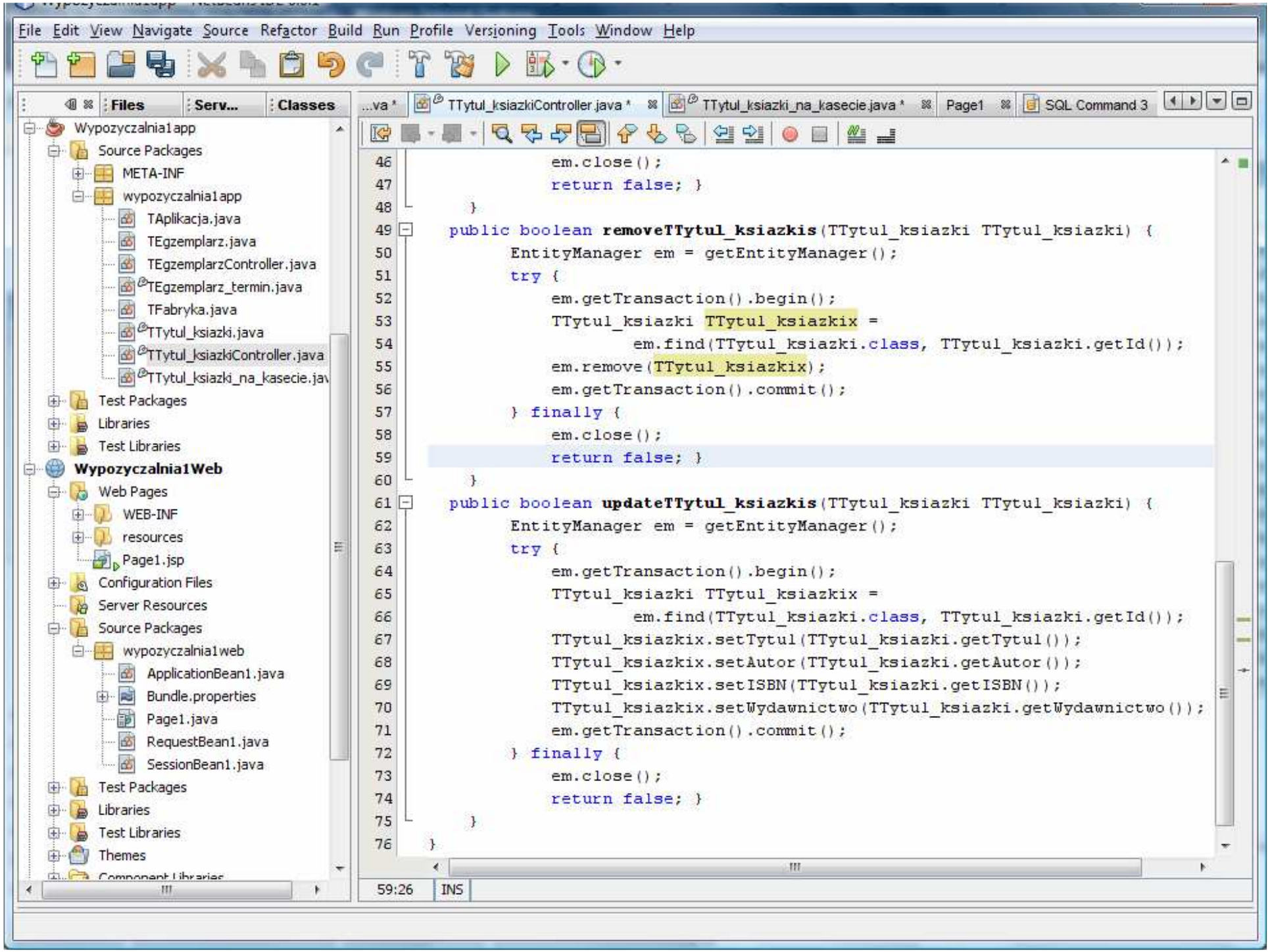


```
public TTytul_ksiazki[] getTTytul_ksiazkis()
 {
     return (TTytul_ksiazki[]) getTTytul_ksiazki().toArray(
                                                    new TTytul_ksiazki[0]);
 }
```

```
public List<TTytul_ksiazki> getTTytul_ksiazki()
 {
    EntityManager em = getEntityManager();
    try {
          javax.persistence.Query q =
             em.createQuery("select c from TTytul_ksiazki as c");
          return q.getResultList();
       } finally {
          em.close();
       }
    }
```

Klasa warstwy integracji (szablon) – dla każdej klasy typu **Entity** (tutaj dla **TTytul_ksiazki**) (13)




File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Files
Serv...
Classes
Wypozyczalnia1app
Source Packages
META-INF
wypozyczalnia1app
TAplikacja.java
TEgzemplarz.java
TEgzemplarzController.java
TEgzemplarz_termin.java
TFabryka.java
TTytul_ksiazki.java
TTytul_ksiazkiController.java
TTytul_ksiazki_na_kasecie.jav
Test Packages
Libraries
Test Libraries
Wypozyczalnia1Web
Web Pages
WEB-INF
resources
Page1.jsp
Configuration Files
Server Resources
Source Packages
wypozyczalnia1web
ApplicationBean1.java
Bundle.properties
Page1.java
RequestBean1.java
SessionBean1.java
Test Packages
Libraries
Test Libraries
Themes
...va *
TTytul_ksiazkiController.java *
TTytul_ksiazki_na_kasecie.java *
Page1
SQL Command 3
46 em.close();
47 return false; }
48 }
49 public boolean removeTTytul_ksiazkis(TTytul_ksiazki TTytul_ksiazki) {
50 EntityManager em = getEntityManager();
51 try {
52 em.getTransaction().begin();
53 TTytul_ksiazki TTytul_ksiazkix =
54 em.find(TTytul_ksiazki.class, TTytul_ksiazki.getId());
55 em.remove(TTytul_ksiazkix);
56 em.getTransaction().commit();
57 } finally {
58 em.close();
59 return false; }
60 }
61 public boolean updateTTytul_ksiazkis(TTytul_ksiazki TTytul_ksiazki) {
62 EntityManager em = getEntityManager();
63 try {
64 em.getTransaction().begin();
65 TTytul_ksiazki TTytul_ksiazkix =
66 em.find(TTytul_ksiazki.class, TTytul_ksiazki.getId());
67 TTytul_ksiazkix.setTytul(TTytul_ksiazki.getTytul());
68 TTytul_ksiazkix.setAutor(TTytul_ksiazki.getAutor());
69 TTytul_ksiazkix.setISBN(TTytul_ksiazki.getISBN());
70 TTytul_ksiazkix.setWydawnictwo(TTytul_ksiazki.getWydawnictwo());
71 em.getTransaction().commit();
72 } finally {
73 em.close();
74 return false; }
75 }
76 }
59:26
INS

(14a) – Standaryzacja nazw metod dostępu do składowych w klasach @Entity - przykłady

1. Należy przyjąć zasadę, że **metody dostępu** do składowych typu **set** i **get** przyjmują pełne nazwy przez dodanie przyrostka jako nazwy atrybutu z zamienioną pierwszą literą na dużą

   np. definicja atrybutu – TTytul_ksiazki mTytul_ksiazki;

   metoda typu set – **public void** setMTytul_ksiazki (TEgzemplarz val)

   { mTytul_ksiazki = val; }

   metoda typu get - public TTytul_ksiazki getMKsiazka ()

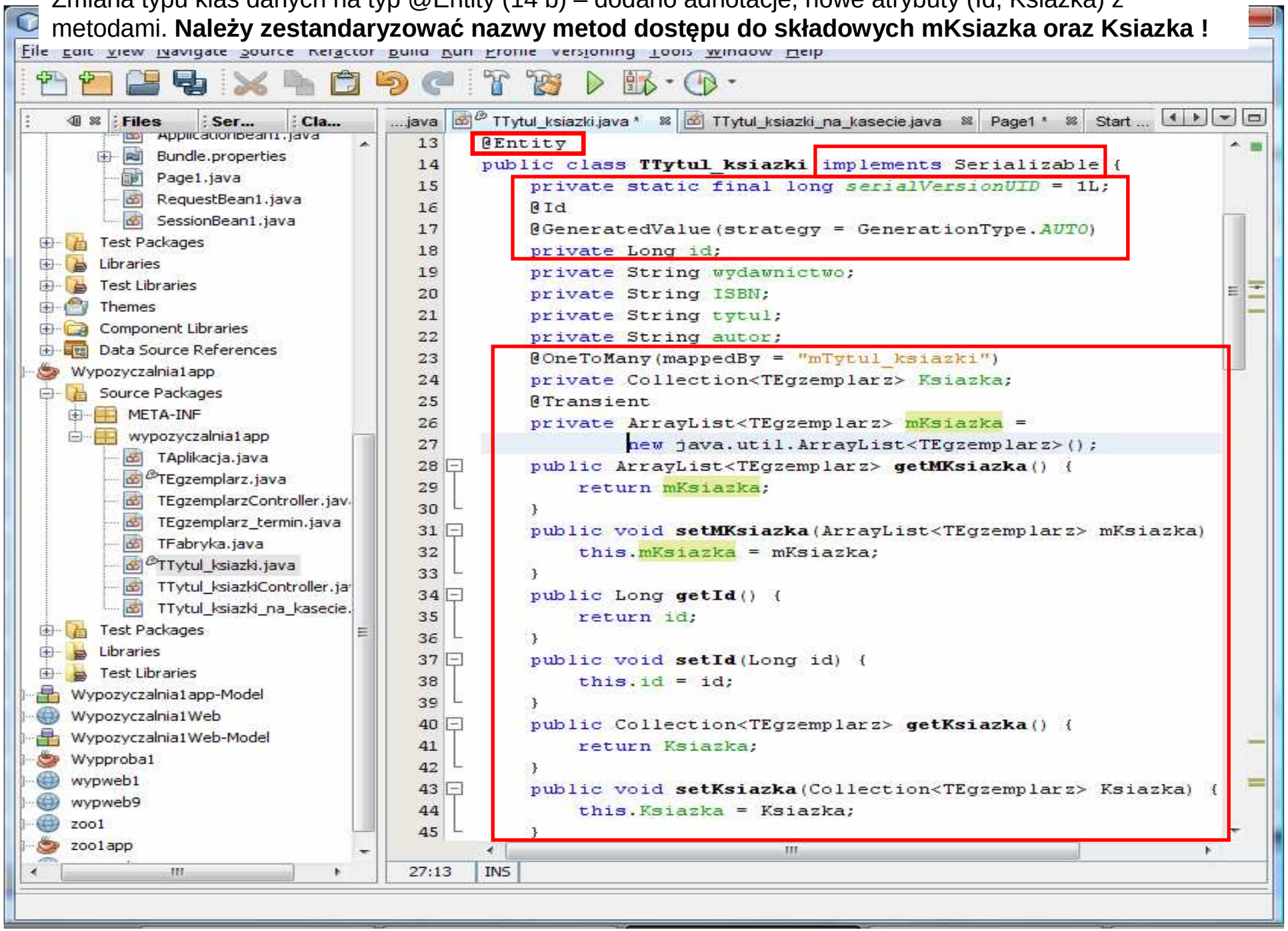
   { **return** mTytul_ksiazki; }

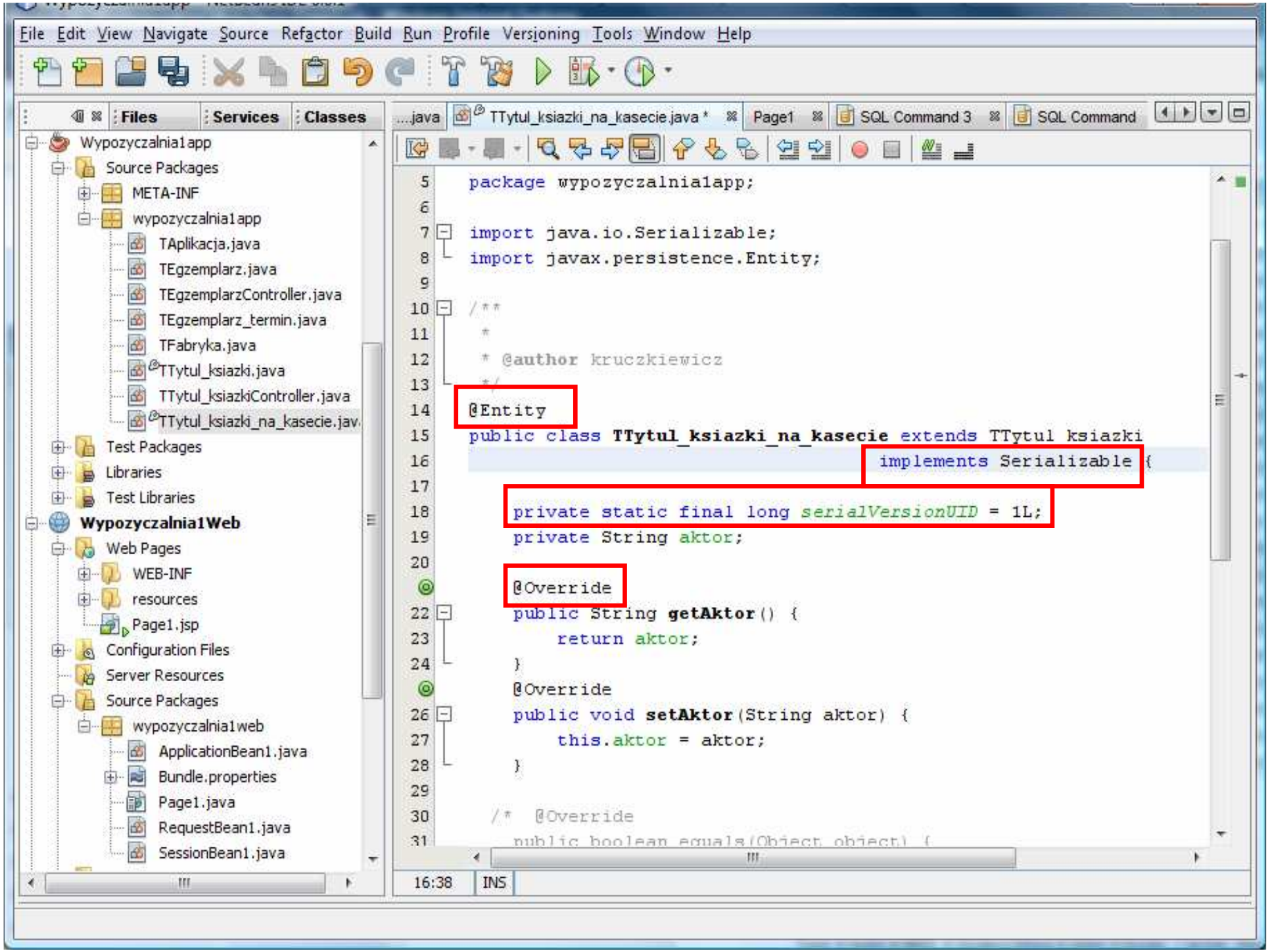
2. Usunąć wszystkie niestandardowe nazwy metod dostępu do każdej ze składowych
3. Wykonać metody typu **set** i **get** zgodnie z p.1 w sposób automatyczny:

   3.1. Kliknąć prawym klawiszem myszy na ciało klasy w trybie Java, wybrać Refactor, następnie Encapsulate Fields

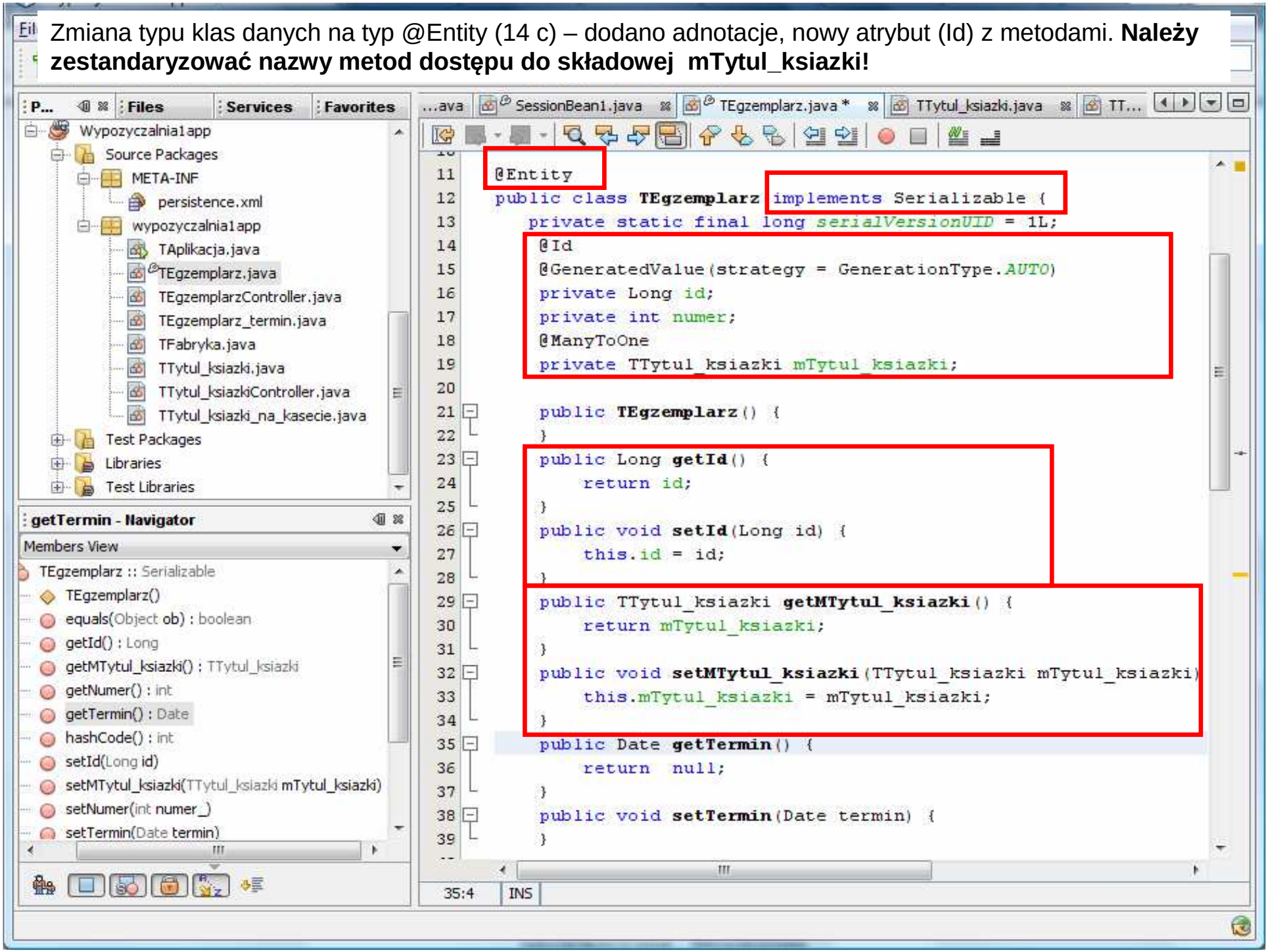
   3.2. Zaznaczyć Create Getter oraz Create Setter i nacisnąć Refactor

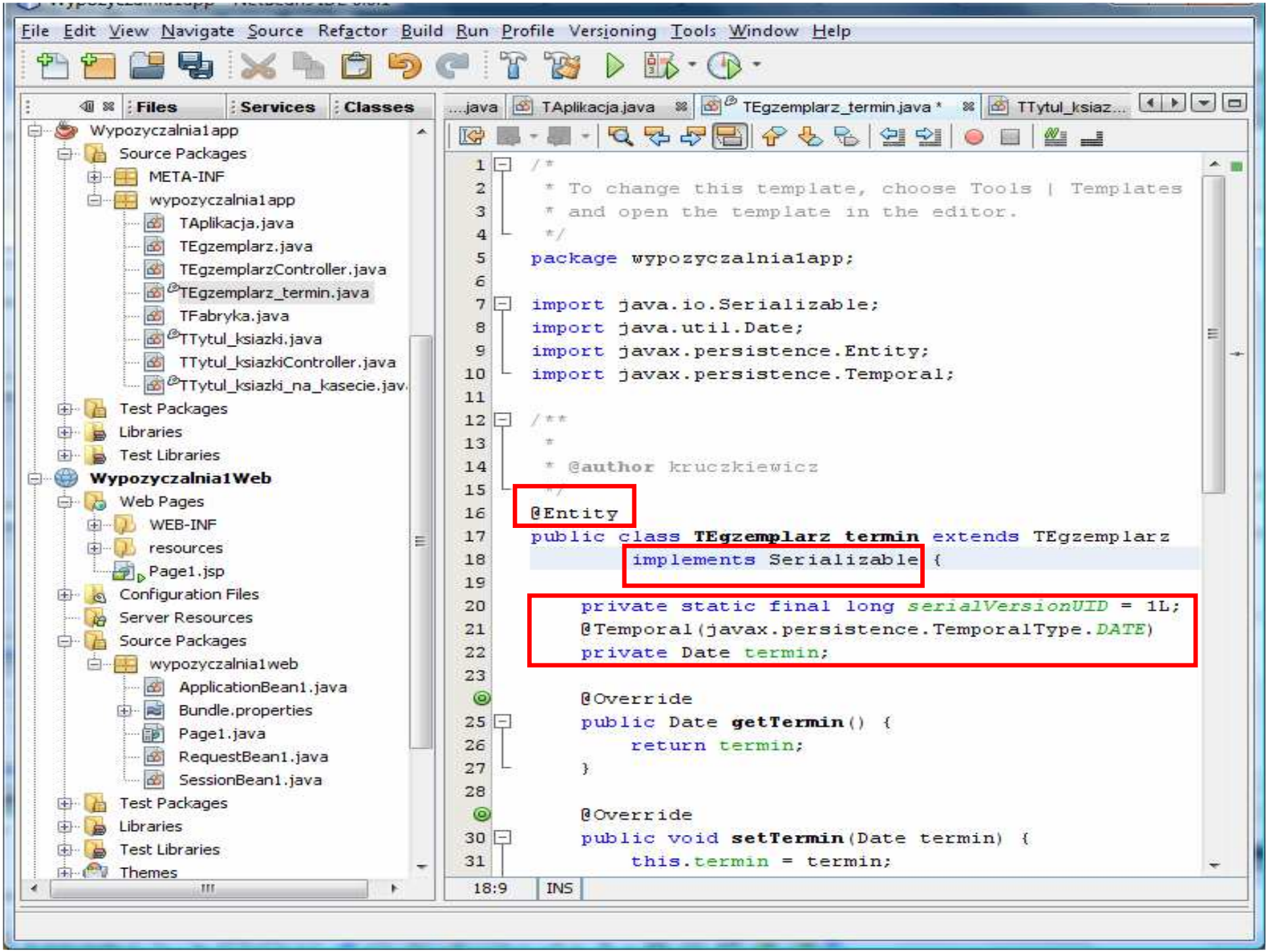
Zmiana typu klas danych na typ @Entity (14 b) – dodano adnotacje, nowe atrybuty (Id, Ksiazka) z metodami. **Należy zestandaryzować nazwy metod dostępu do składowych mKsiazka oraz Ksiazka !**



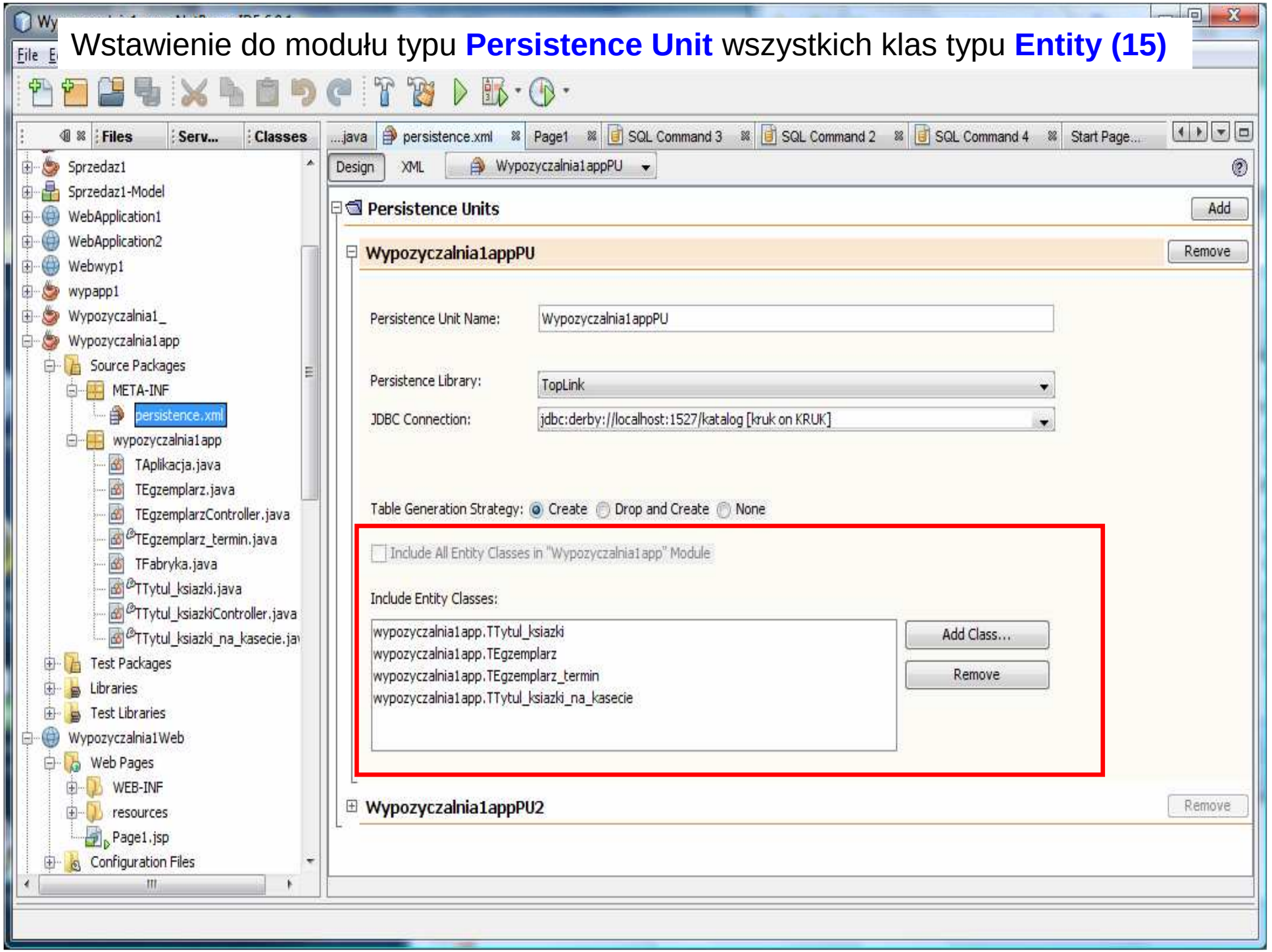

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Files
Services
Classes
Wypozyczalnia1app
Source Packages
META-INF
wypozyczalnia1app
TAplikacja.java
TEgzemplarz.java
TEgzemplarzController.java
TEgzemplarz_termin.java
TFabryka.java
TTytul_ksiazki.java
TTytul_ksiazkiController.java
TTytul_ksiazki_na_kasecie.jav
Test Packages
Libraries
Test Libraries
Wypozyczalnia1Web
Web Pages
WEB-INF
resources
Page1.jsp
Configuration Files
Server Resources
Source Packages
wypozyczalnia1web
ApplicationBean1.java
Bundle.properties
Page1.java
RequestBean1.java
SessionBean1.java
...java
TTytul_ksiazki_na_kasecie.java *
Page1
SQL Command 3
SQL Command
5 package wypozyczalnia1app;
6
7 import java.io.Serializable;
8 import javax.persistence.Entity;
9
10 /**
11 *
12 * @author kruczkiewicz
13 */
14 @Entity
15 public class TTytul_ksiazki_na_kasecie extends TTytul_ksiazki
16 implements Serializable {
17
18 private static final long serialVersionUID = 1L;
19 private String aktor;
20
22 @Override
23 public String getAktor() {
24 return aktor;
26 }
27 @Override
28 public void setAktor(String aktor) {
29 this.aktor = aktor;
30 }
31
/* @Override
public boolean equals(Object object) {
16:38
INS

Zmiana typu klas danych na typ @Entity (14 c) – dodano adnotacje, nowy atrybut (Id) z metodami. **Należy zestandaryzować nazwy metod dostępu do składowej mTytul_ksiazki!**




File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Files
Services
Classes
Wypozyczalnia1app
Source Packages
META-INF
wypozyczalnia1app
TAplikacja.java
TEgzemplarz.java
TEgzemplarzController.java
TEgzemplarz_termin.java
TFabryka.java
TTytul_ksiazki.java
TTytul_ksiazkiController.java
TTytul_ksiazki_na_kasecie.jav
Test Packages
Libraries
Test Libraries
Wypozyczalnia1Web
Web Pages
WEB-INF
resources
Page1.jsp
Configuration Files
Server Resources
Source Packages
wypozyczalnia1web
ApplicationBean1.java
Bundle.properties
Page1.java
RequestBean1.java
SessionBean1.java
Test Packages
Libraries
Test Libraries
Themes
....java
TAplikacja.java
TEgzemplarz_termin.java *
TTytul_ksiaz...
1 /*
2 * To change this template, choose Tools | Templates
3 * and open the template in the editor.
4 */
5 package wypozyczalnia1app;
6
7 import java.io.Serializable;
8 import java.util.Date;
9 import javax.persistence.Entity;
10 import javax.persistence.Temporal;
11
12 /**
13 *
14 * @author kruczkiewicz
15 */
16 @Entity
17 public class TEgzemplarz_termin extends TEgzemplarz
18 implements Serializable {
19
20 private static final long serialVersionUID = 1L;
21 @Temporal(javax.persistence.TemporalType.DATE)
22 private Date termin;
23
25 @Override
26 public Date getTermin() {
27 return termin;
28 }
30 @Override
31 public void setTermin(Date termin) {
this.termin = termin;
18:9
INS

Wstawienie do modułu typu **Persistence Unit** wszystkich klas typu **Entity (15)**



3. Tworzenie warstwy klienta, prezentacji

Integrowanie warstwy prezentacji i klienta wykonane w technologii **Java Server Faces**

- z projektem warstwy biznesowej zawierającej model obiektowy
- oraz warstwą integracji

Tworzenie warstwy prezentacji pozwalającej na utworzenie pustych tabel w bazie danych skojarzonej projektem - persistence.xml (1)

- Należy założyć projekt typu **Web Application** - slajd(1), slajd(2), slajd(3)
- Należy projekt typu **Java Application** zawierający warstwę biznesową oraz integracji (w przykładzie Wypozyczalnia1app) połączyć z projektem **Web Application** – należy w zakładce **Projects** prawym klawiszem myszy kliknąć na nazwę projektu i wybrać opcję **Properties** - slajd (4). Zgodnie ze slajdem (5) należy wybrać opcję **Libraries** w części **Categories** formularza **Properties**, i następnie kliknąć na przycisk **Add Project** – w kolejnym formularzu wybrać projekt typu **Java Application** i zatwierdzić. Po powrocie do zakładki **Projects** w opcji **Libraries** pojawił się plik typu jar zawierający pakiety projektu **Wypozyczalnia1app** z klasami typu **Entity** oraz z warstwą integracji (slajd (6)).

  Uwaga: Przy przenoszeniu projektu należy usunąć podobnie jak przy łączeniu projektów projekt typu JavaApplication z projektu typu WebApplication i ponownie połączyć projekty jak wyżej.
- Zgodnie ze slajdem (7) należy przeciągnąć w trybie **Design** komponent **Table** na stronę **Page1.jsp**, reprezentującą warstwę prezentacji. Należy kliknąć prawym klawiszem myszy na komponent i następnie kliknąć na pozycję **Add Binding Attribute (slajd 0)**
- Zgodnie ze slajdem (8) i slajdem(9), należy w klasie sesji typu **SessionBean1** wpisać kod pozwalający pobierać dane o tytułach z bazy danych i mapować je na obiekty typu **TTytul_ksiazki** i **TTytul_ksiazki_na_kasecie** zapisując je do tabeli **TTytul_ksiazki tytuly[ ].** Wykonuje to metoda **updateTytuls()** wywołana w metodzie **init** klasy **SessionBean1.** Metoda **updateTytuls()** wywołuje metodę **getTTytul_ksiazkis** z klasy **TTytul_ksiazkiController,** czyli pobiera z bazy danych zawartość tabeli TTytul_ksiazki**. Jeśli baza jest pusta, spowoduje wywołanie skryptów typu Create tworzących puste tabele w wyniku mapowania modelu obiektowego**
- Zgodnie ze slajdami (10) i (11), należy wykonać operację **Bind to Data**: kliknąć prawym klawiszem myszy komponent **Table**, wybrać opcję **Bind to Data**. Następnie w formularzu tej operacji w zakładce **Bind to object** wybrać z opcji **Get Data From** tablicę **tytuly** z klasy **SessionBean1**. Na slajdzie (13) pokazano rezultat działań.
- Należy uruchomić aplikację (Run) – slajd(14)

Tworzenie warstwy prezentacji pozwalającej na utworzenie pustych tabel w bazie danych skojarzonej projektem - persistence.xml (2)

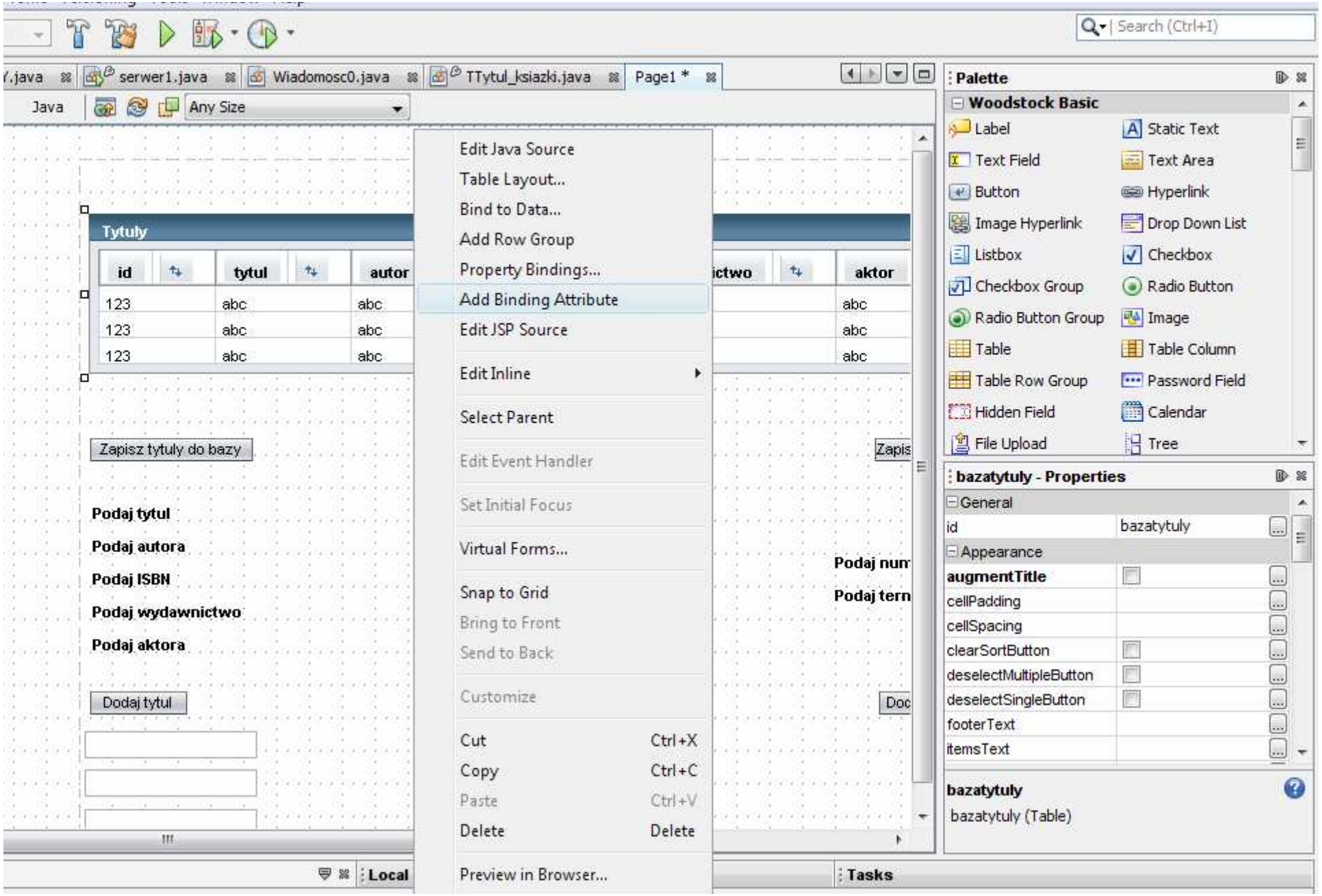
- Zgodnie ze slajdem (15) w zakładce **Services** po wyborze bazy danych współpracującej z aplikacją (+) pojawiły się trzy tabele: **TTytul_ksiazki** i **TEgzemplarz** – obie przechowują dane odpowiadające dwom typom danych dzięki dziedziczeniu w obu tych klasach, Kolumna **DTYPE** służy do rozróżniania typu tych danych w modelu relacyjnym. Trzecia tabela **SEQUENCE** przechowuje informacje do generowania nie powtarzających się wartości kluczy głównych dzięki adnotacji

  @Id

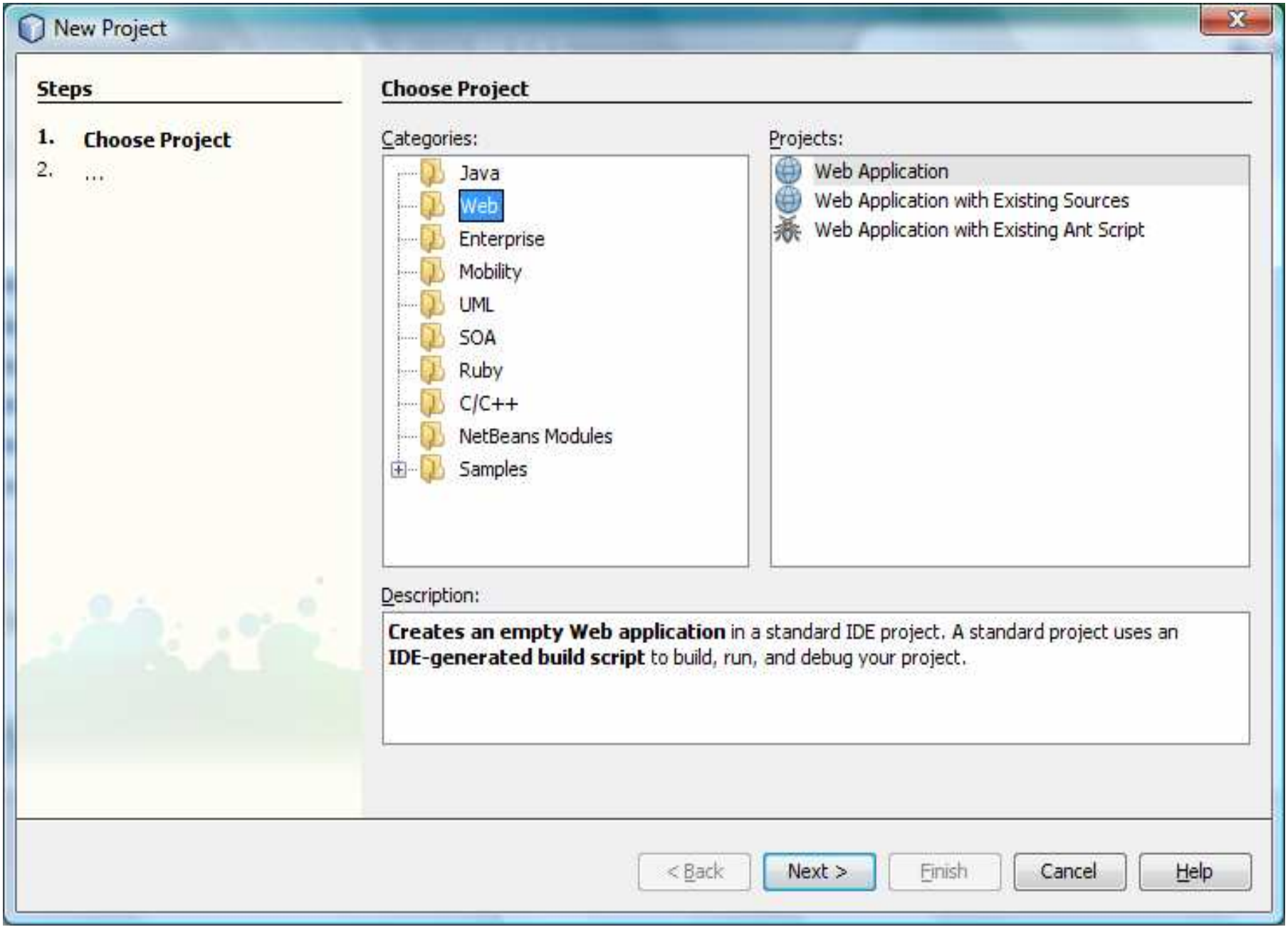
  @GeneratedValue(strategy = GenerationType.AUTO) przy kluczach głównych w klasach typu **Entity**, określającą strategię domyślną generowania kluczy głównych w technologii **TopLink**

  Pokazano wynik działania opcji **Design Query** wybranej z wyskakującej listy po naciśnięciu prawego klawisza na nazwie tabeli **TEgzemplarz** i przeciągnięciu lewym klawiszem myszy tabeli **TTytul_ksiazki.**
- Po uruchomieniu w oknie Output, w zakładce GlassFishV2 przedstawiono skrypty tworzenia pustych tabel w bazie danych (skrypty Create) przy drugim uruchomieniu Run (wygenerowane wyjątki są związane z próbą tworzenia ponownie tabel) – slajd (16). Na slajdzie (17) pokazano działanie aplikacji z wyłączona opcją Create w pliku persistence.xml (włączono opcję None).
- **W dalszej części projektu należy sprawdzać komunikaty wyświetlane w zakładce GlassFishV2 okienka Output. Mogą pomóc w wykryciu błędów w aplikacji. Część błędów, mimo poprawy w kodzie aplikacji, można usunąć po zamknięciu serwera aplikacji – można to zrobić w zakładce Services w pozycji Servers lub przez ponowne uruchomienie środowiska NetBeans 6.01. Każda zmiana w dołączonym projekcie typu Java Application wymaga ponownego skompilowania tego projektu w trybie Clean and Build**

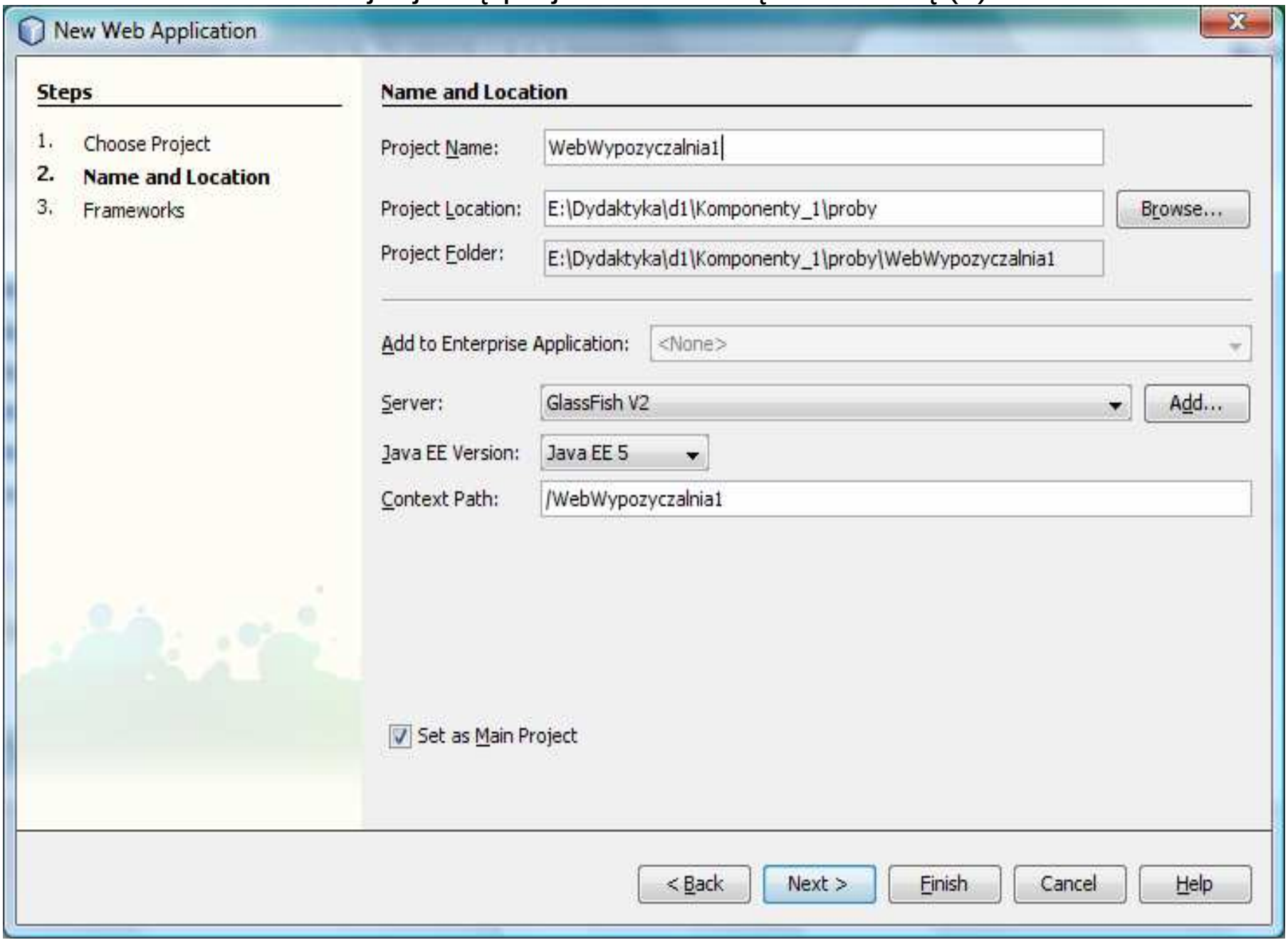
Wstawianie do pliku typu Java strony JSP obiektów reprezentujących komponenty wizualne interfejsu graficznego użytkownika (0)



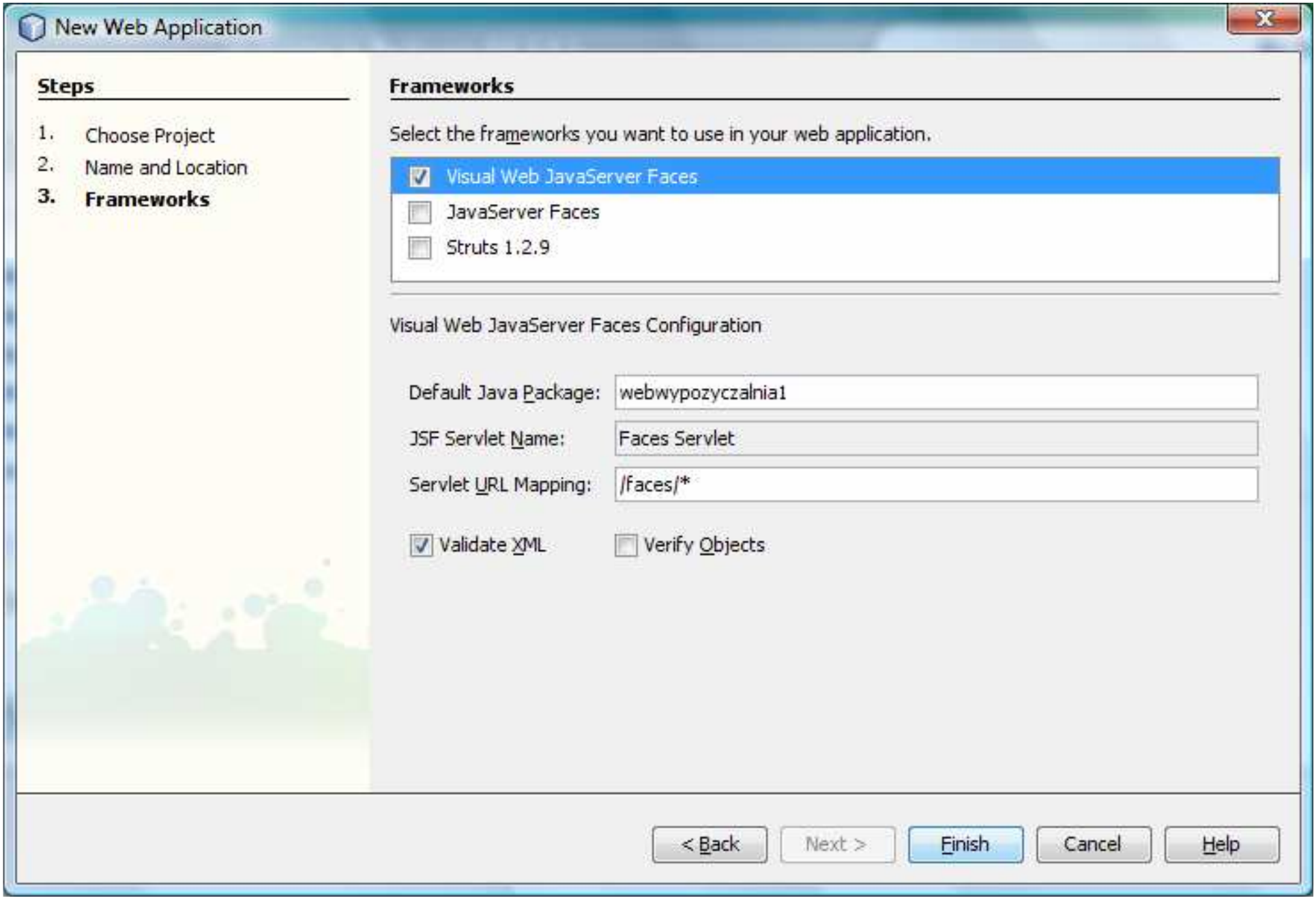
Tworzenie projektu kategorii Web typu Web Application (1)

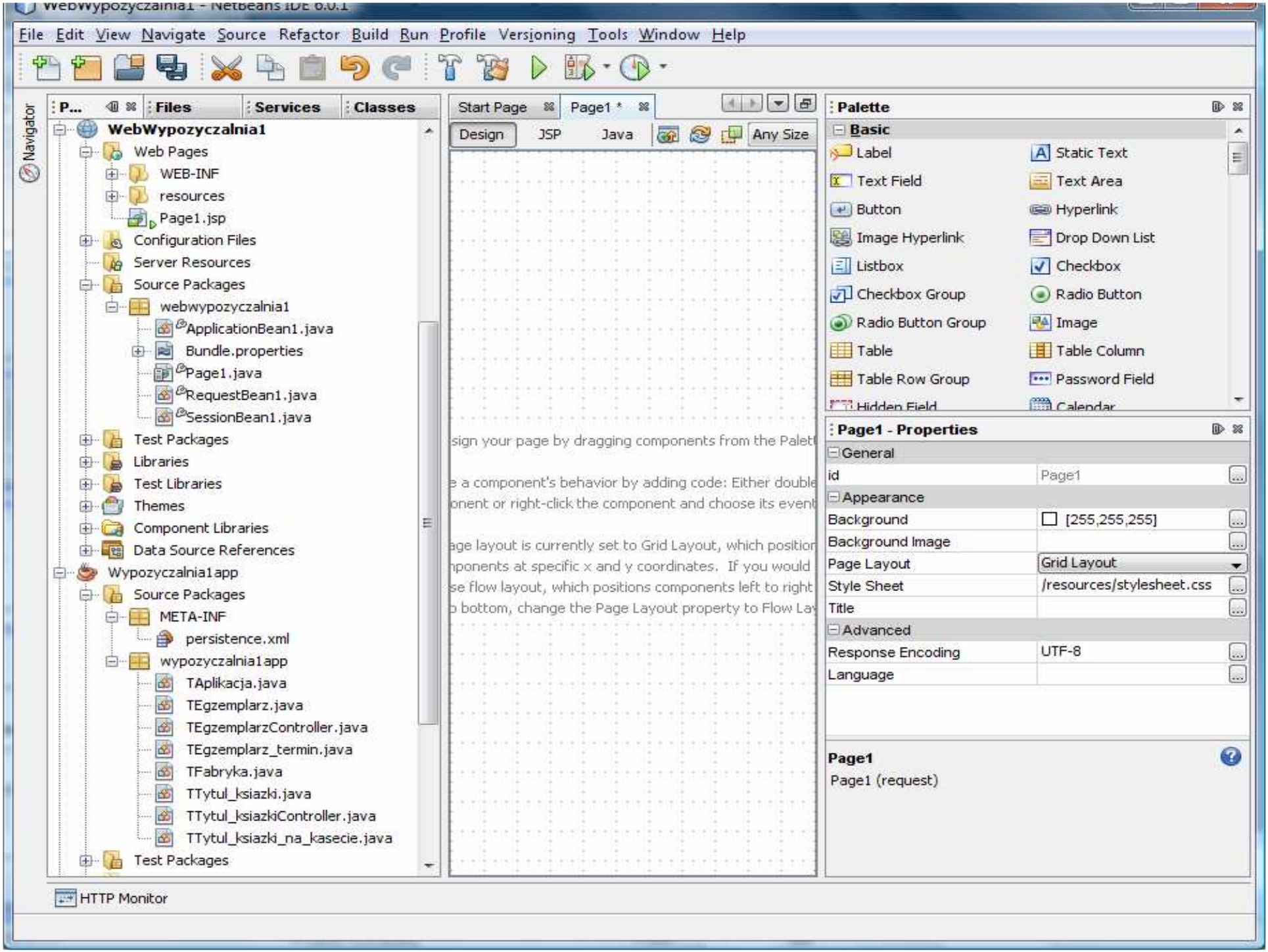


Projekt typu WebApplication tworzony w tym samym katalogu, w którym znajduje się projekt z warstwą biznesową (2)

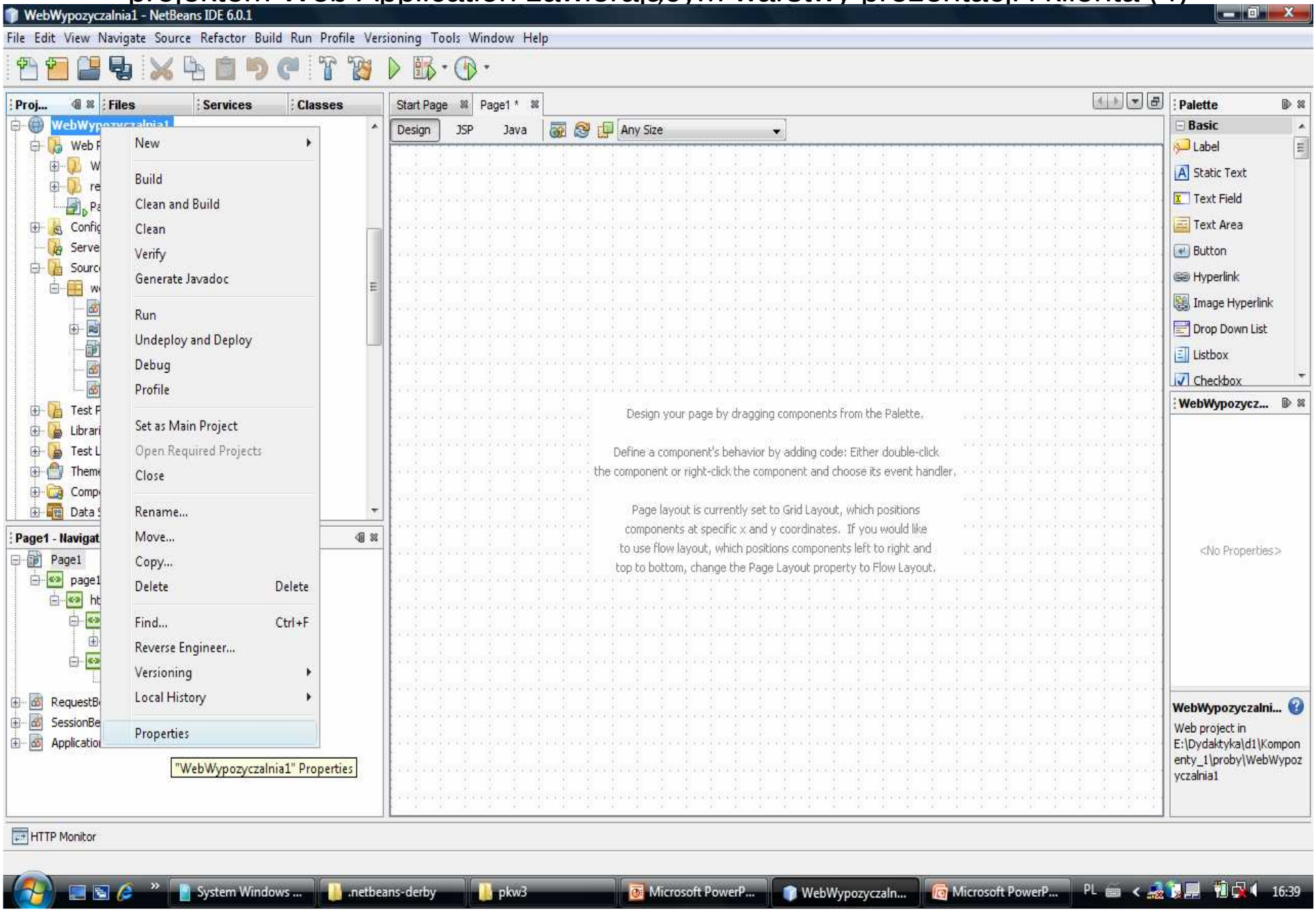


Projekt powinien być oparty na Visual Web JavaSever Faces (3)

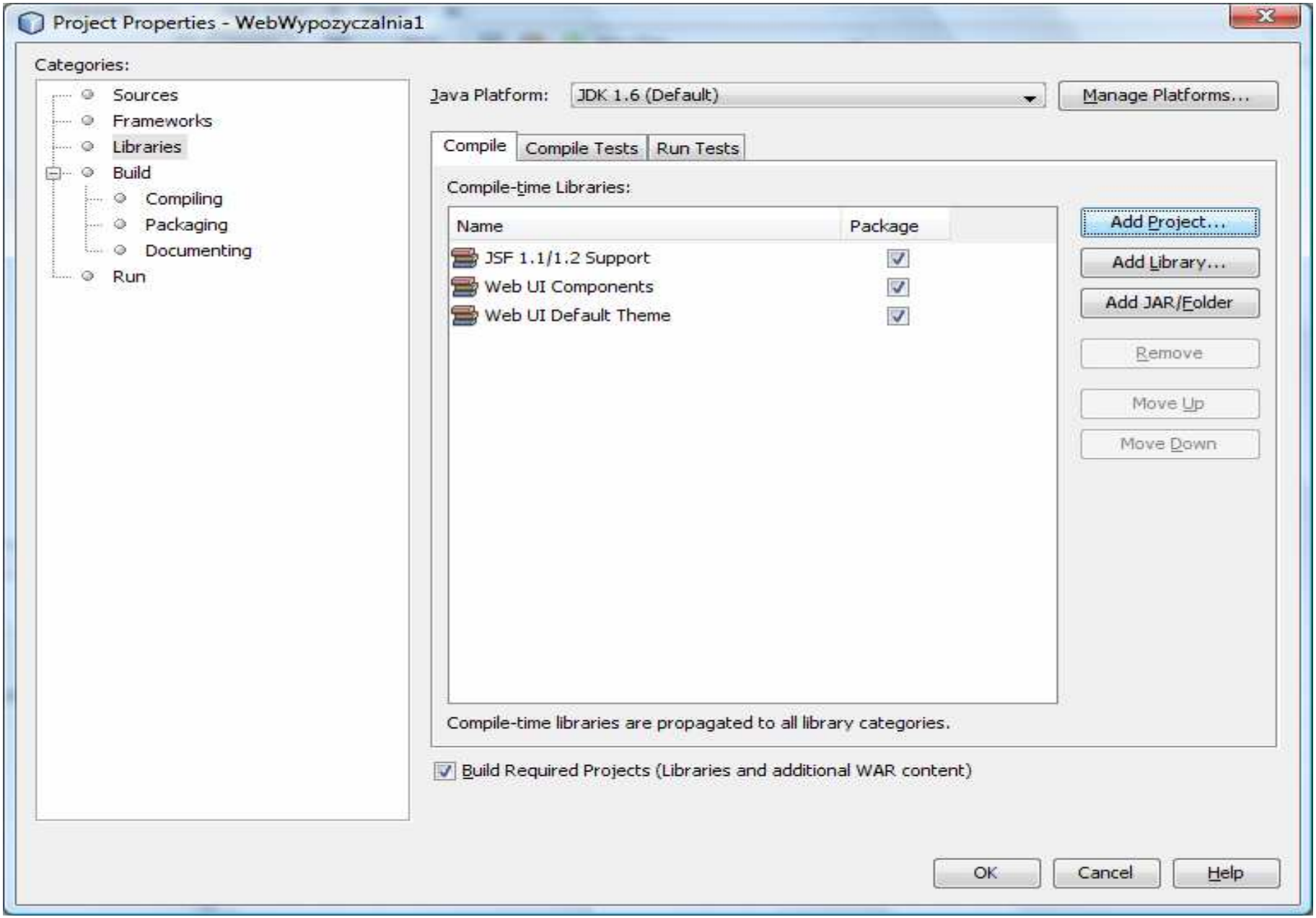


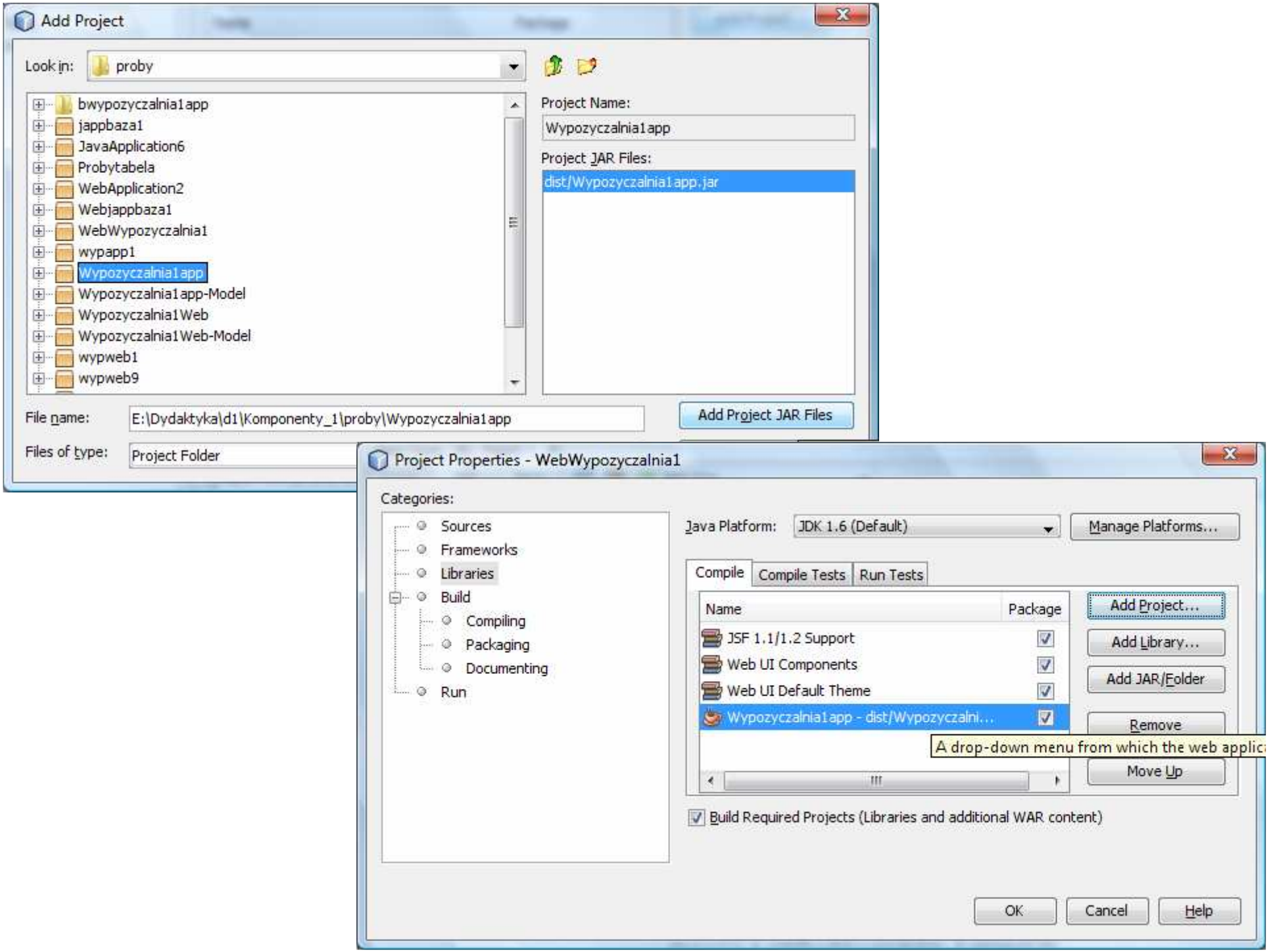

WebWypozyczalnia1 - Netbeans IDE 6.0.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Navigator
P... Files Services Classes
WebWypozyczalnia1
Web Pages
WEB-INF
resources
Page1.jsp
Configuration Files
Server Resources
Source Packages
webwypozyczalnia1
ApplicationBean1.java
Bundle.properties
Page1.java
RequestBean1.java
SessionBean1.java
Test Packages
Libraries
Test Libraries
Themes
Component Libraries
Data Source References
Wypozyczalnia1app
Source Packages
META-INF
persistence.xml
wypozyczalnia1app
TAplikacja.java
TEgzemplarz.java
TEgzemplarzController.java
TEgzemplarz_termin.java
TFabryka.java
TTytul_ksiazki.java
TTytul_ksiazkiController.java
TTytul_ksiazki_na_kasecie.java
Test Packages
Start Page
Page1 *
Design JSP Java Any Size
sign your page by dragging components from the Palett
e a component's behavior by adding code: Either double
onent or right-click the component and choose its event
age layout is currently set to Grid Layout, which position
nponents at specific x and y coordinates. If you would
se flow layout, which positions components left to right
o bottom, change the Page Layout property to Flow Lay
Palette
Basic
Label
Static Text
Text Field
Text Area
Button
Hyperlink
Image Hyperlink
Drop Down List
Listbox
Checkbox
Checkbox Group
Radio Button
Radio Button Group
Image
Table
Table Column
Table Row Group
Password Field
Hidden Field
Calendar
Page1 - Properties
General
id
Page1
Appearance
Background
[255,255,255]
Background Image
Page Layout
Grid Layout
Style Sheet
/resources/stylesheet.css
Title
Advanced
Response Encoding
UTF-8
Language
Page1
Page1 (request)
HTTP Monitor

Łączenie projektu typu JavaApplication zawierającym warstwę biznesową i integracji z projektem Web Application zawierającym warstwy prezentacji i klienta (4)

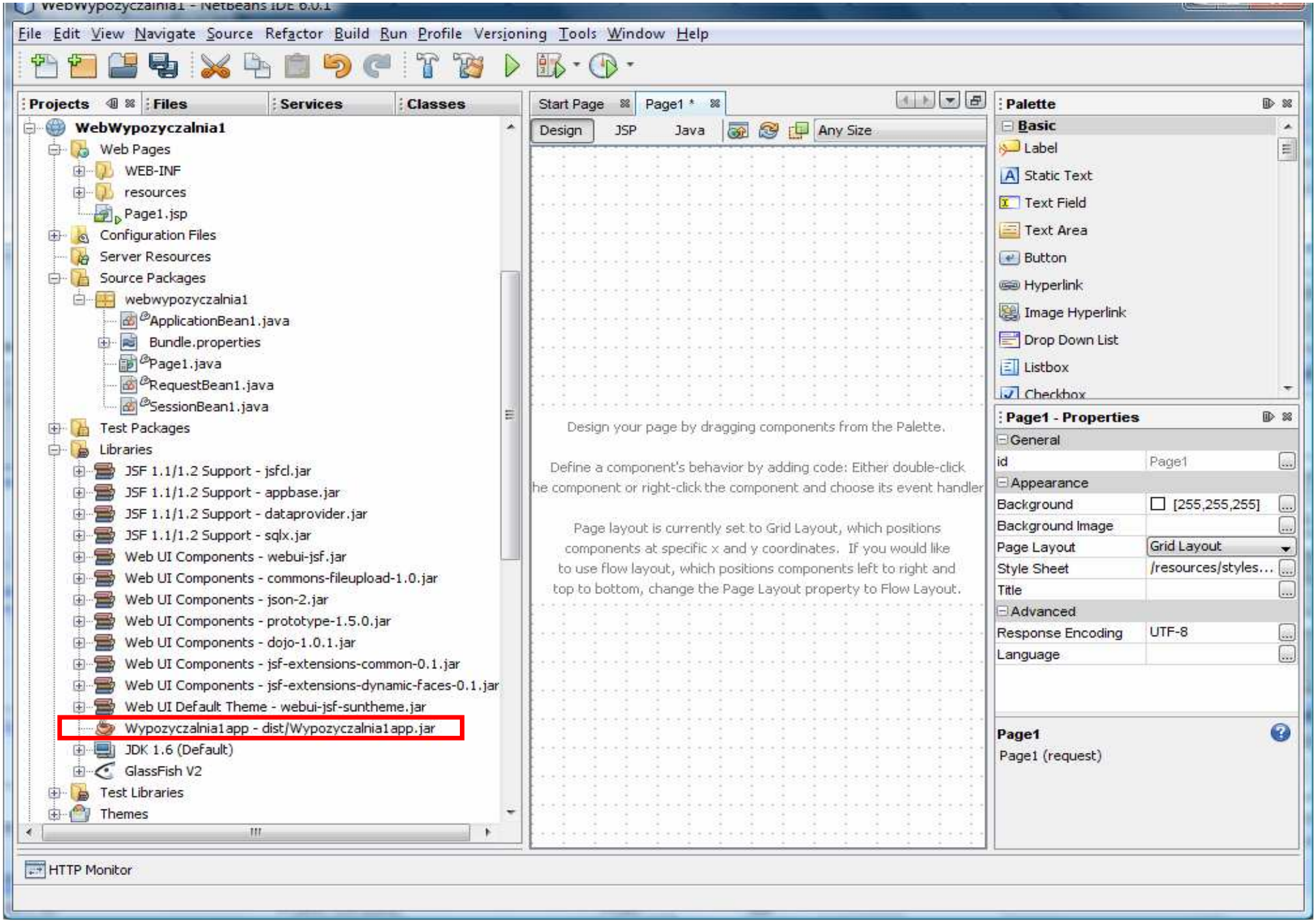


Wybór w okienku **Properties** projektu **Web Application** opcji **Libraries** oraz Add **Project (5)**

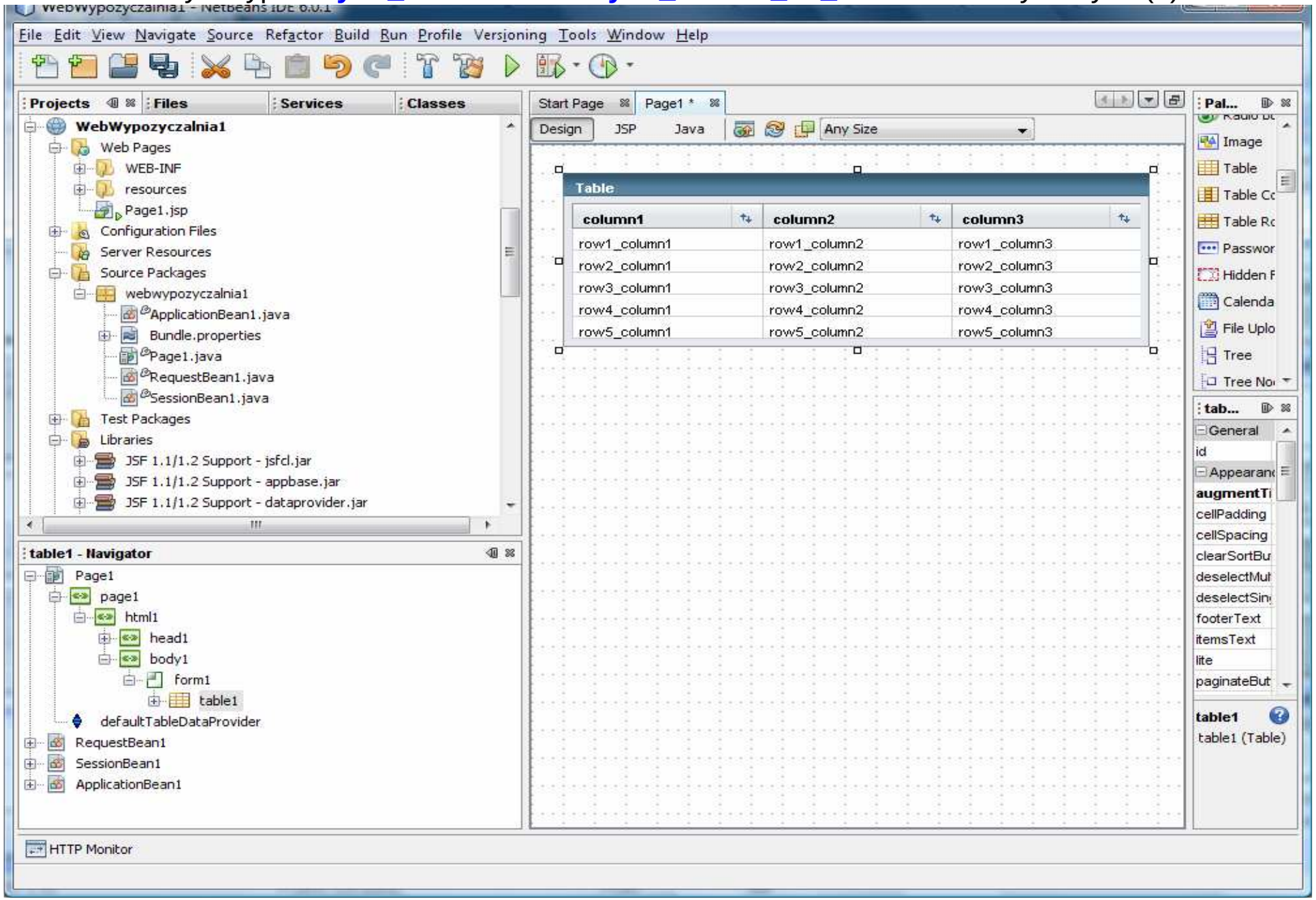



Add Project
Look in:
proby
bwypozyczalnia1app
jappbaza1
JavaApplication6
Probytabela
WebApplication2
Webjappbaza1
WebWypozyczalnia1
wypapp1
Wypozyczalnia1app
Wypozyczalnia1app-Model
Wypozyczalnia1Web
Wypozyczalnia1Web-Model
wypweb1
wypweb9
Project Name:
Wypozyczalnia1app
Project JAR Files:
dist/Wypozyczalnia1app.jar
File name:
E:\Dydaktyka\d1\Komponenty_1\proby\Wypozyczalnia1app
Add Project JAR Files
Files of type:
Project Folder
Project Properties - WebWypozyczalnia1
Categories:
Sources
Frameworks
Libraries
Build
Compiling
Packaging
Documenting
Run
Java Platform:
JDK 1.6 (Default)
Manage Platforms...
Compile
Compile Tests
Run Tests
Name
Package
JSF 1.1/1.2 Support
Web UI Components
Web UI Default Theme
Wypozyczalnia1app - dist/Wypozyczalni...
Add Project...
Add Library...
Add JAR/Folder
Remove
Move Up
A drop-down menu from which the web applic
Build Required Projects (Libraries and additional WAR content)
OK
Cancel
Help

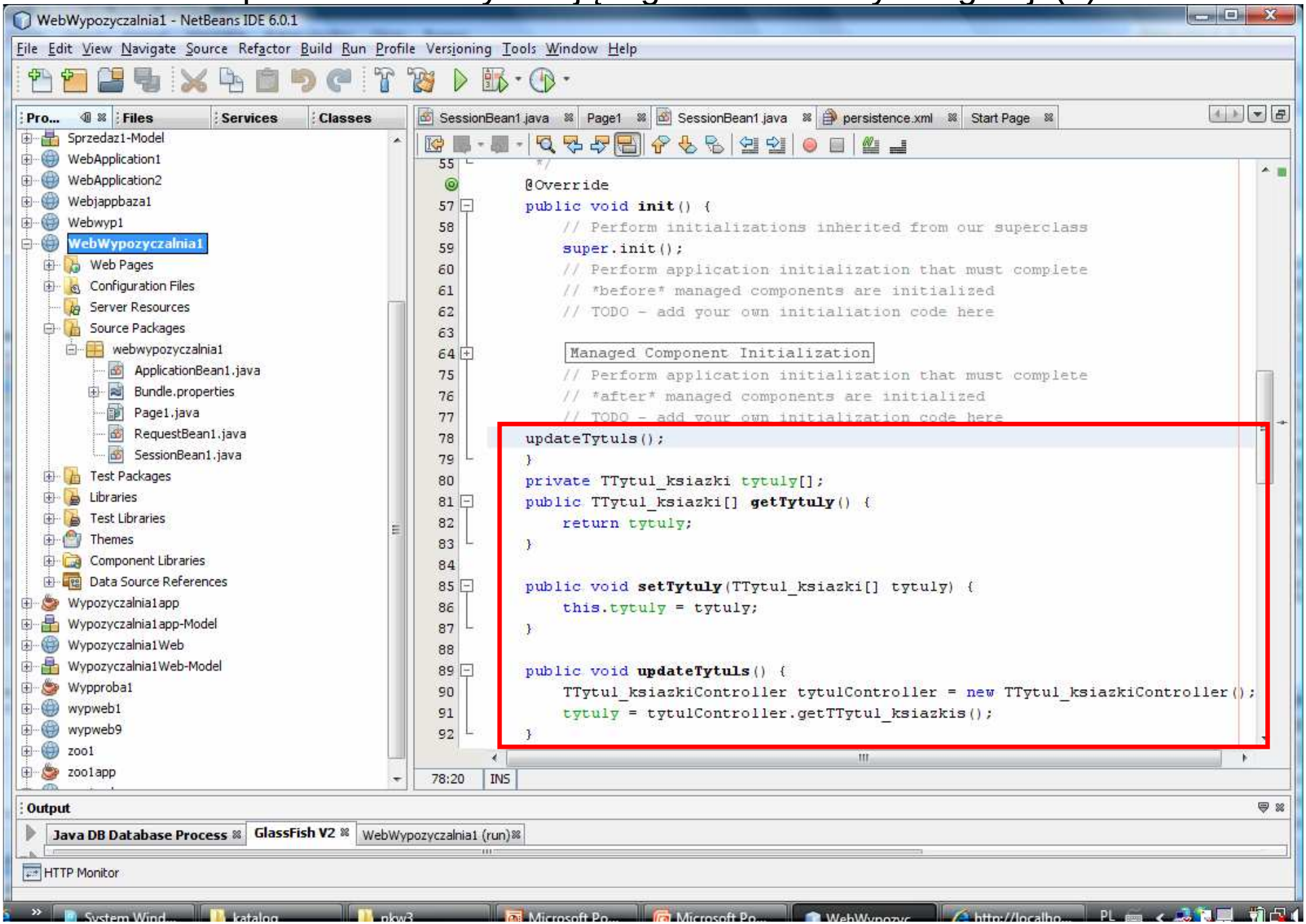
Połączone projekty–projekt Web Application korzysta z klas projektu Java Application (6)



Wstawienie do warstwy prezentacji (**Page1.jsp)** komponentu **Table** w celu wyświetlania w tabeli danych typu **TTytul_ksiazki** oraz **TTytul_ksiazki_na_kasecie** z bazy danych (7)

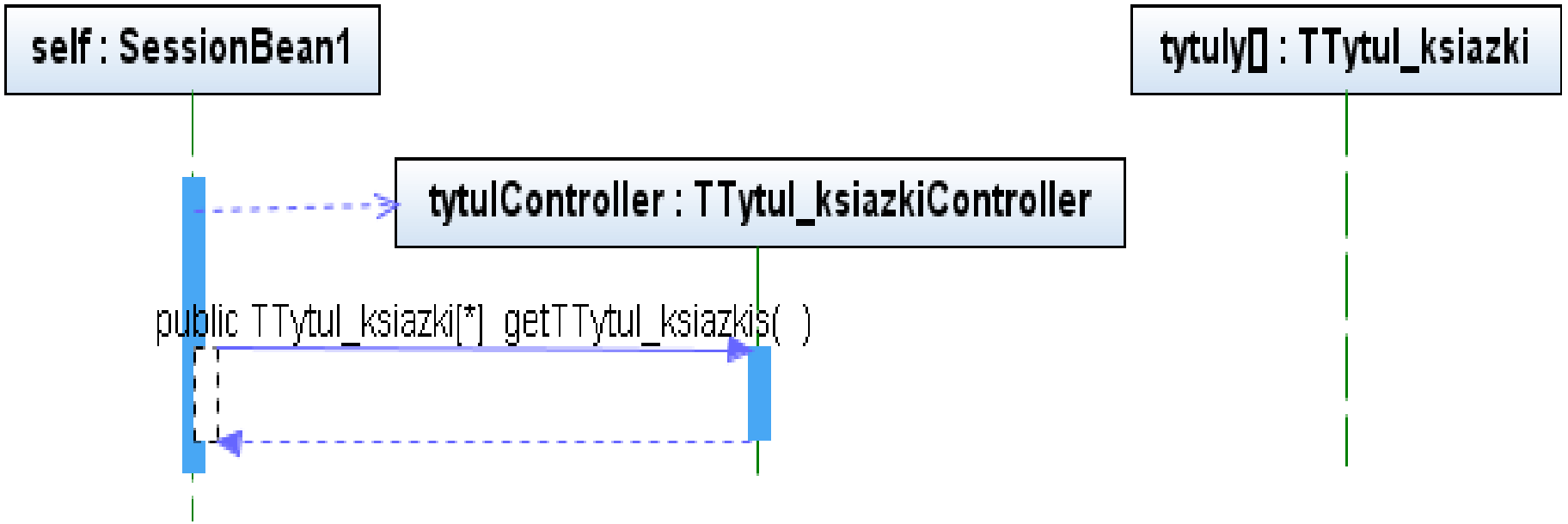


Wpisanie kodu wywołującego kod warstwy integracji (8)

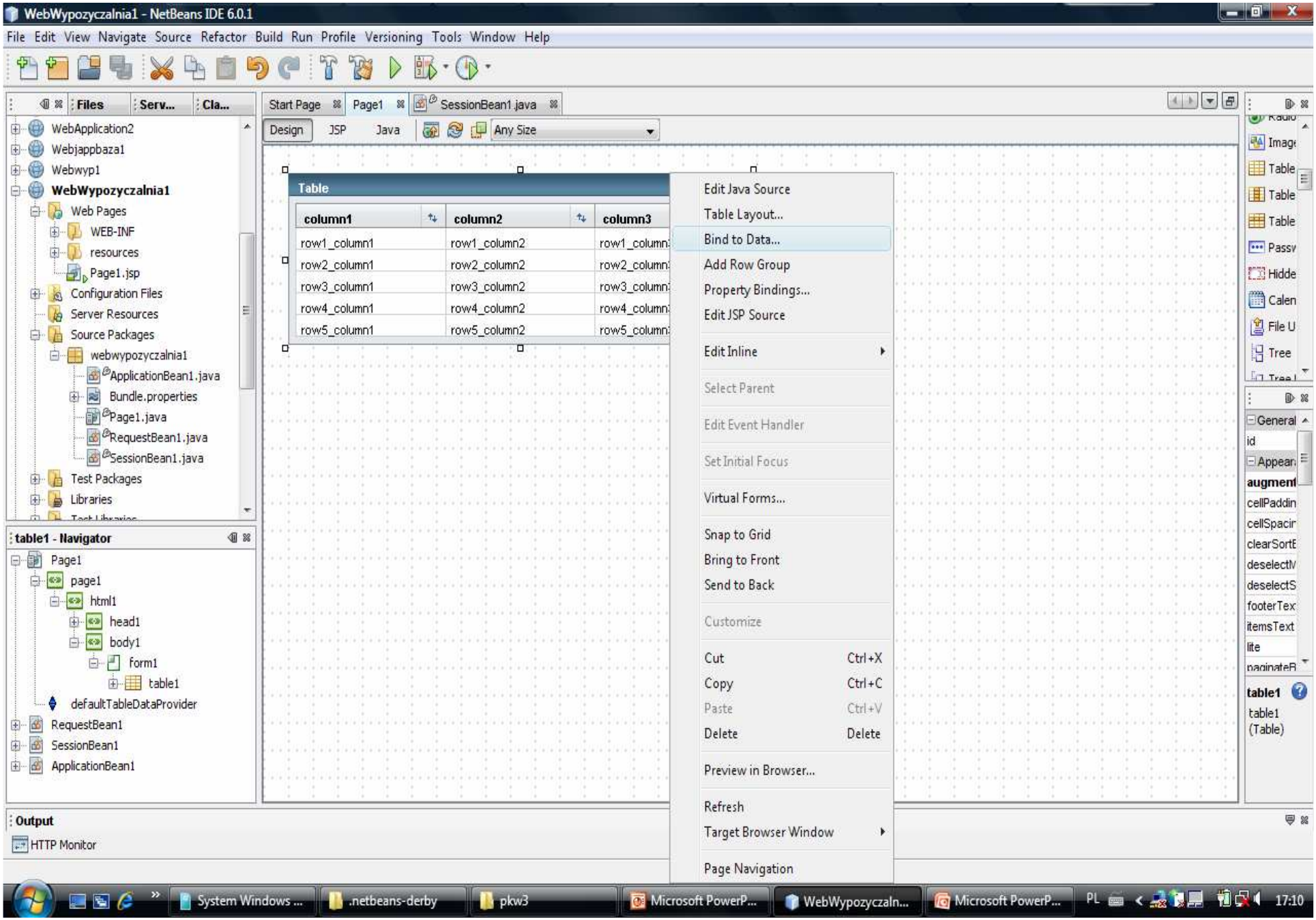


Odczyt tytułów z bazy danych (9)

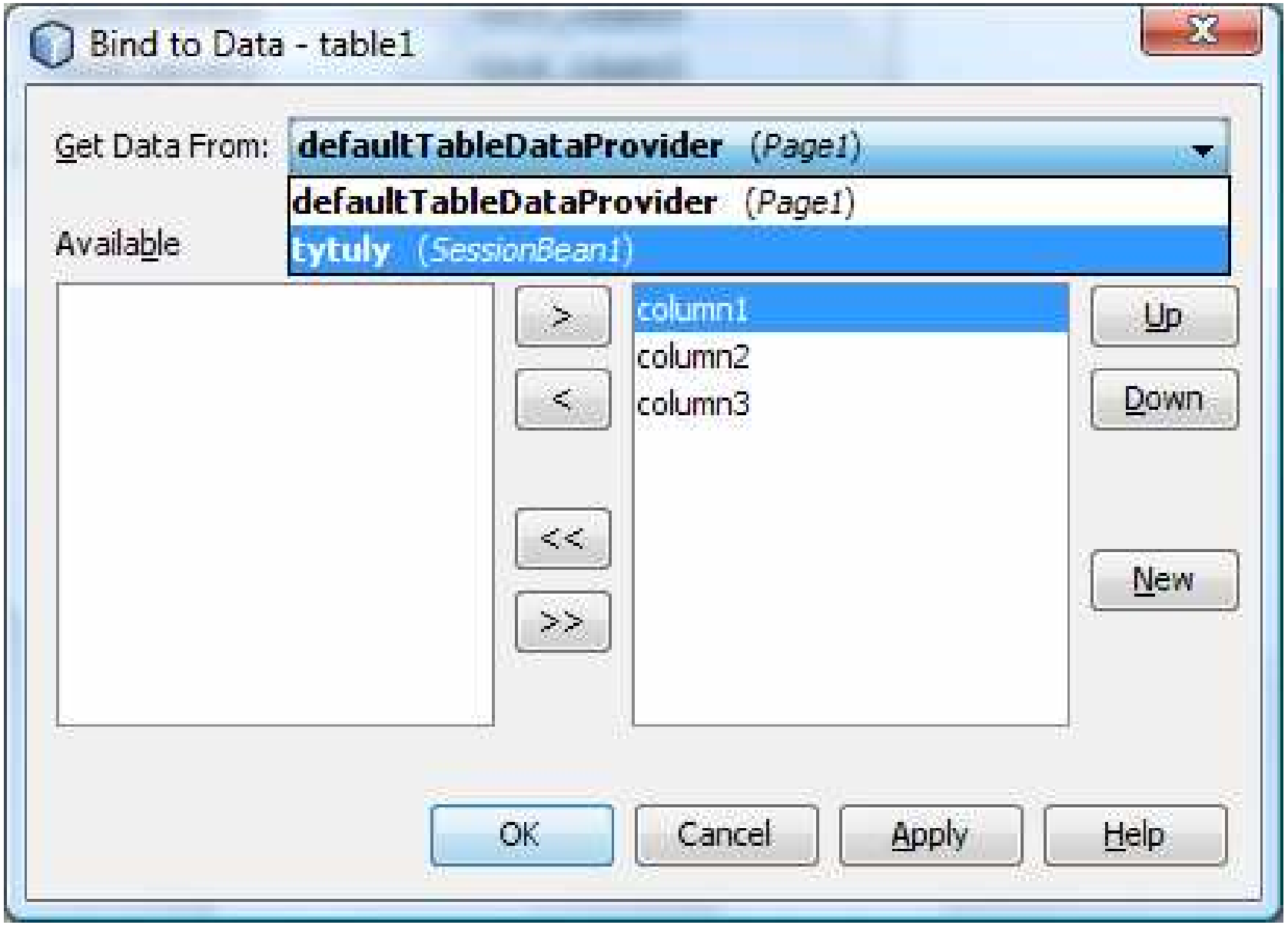
public void SessionBean1::updateTytuls()



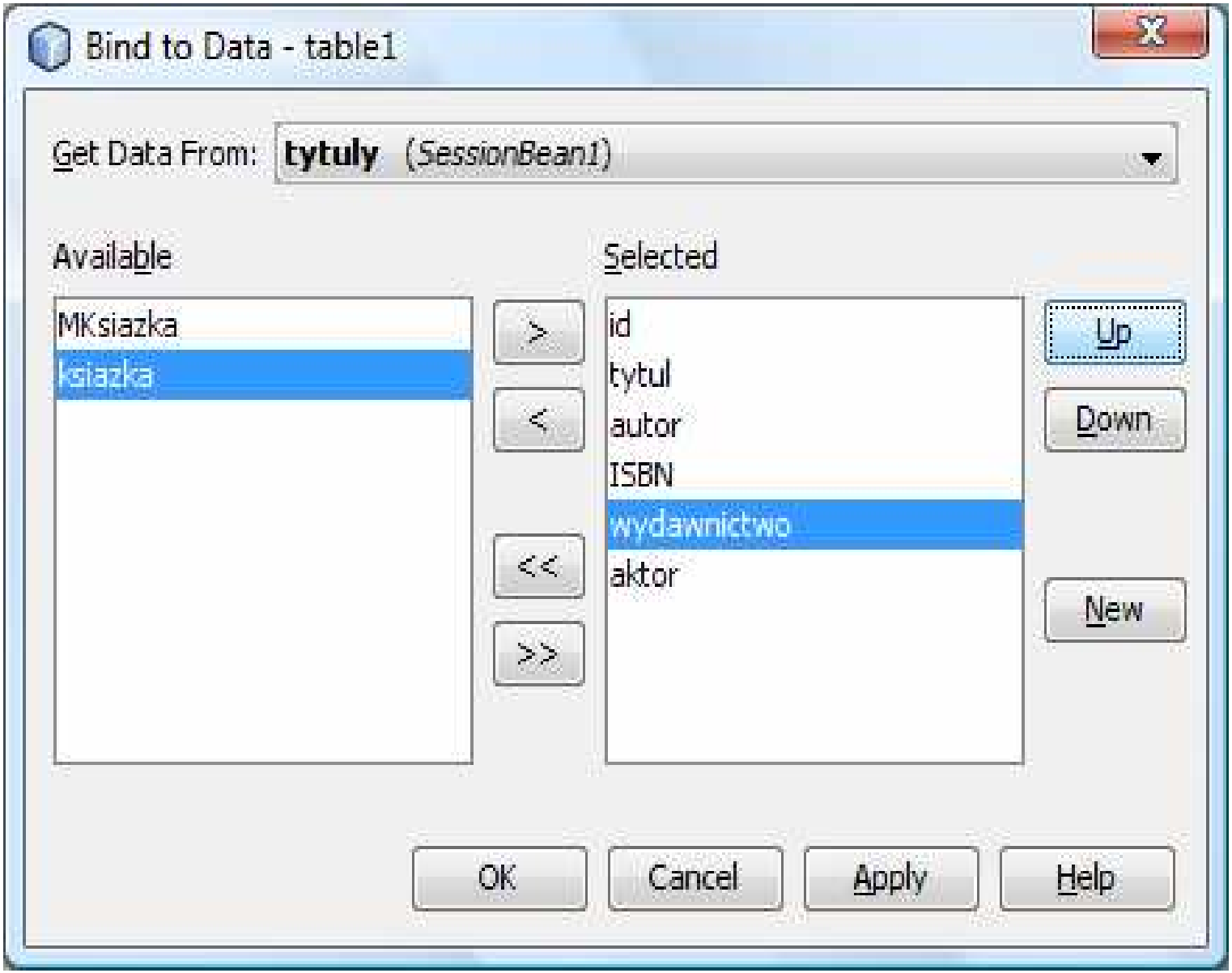
Łączenie komponentu **Table** z tablicą **TTytul_ksiazki ksiazki[ ]** – wybór **Bind to Data (10)**



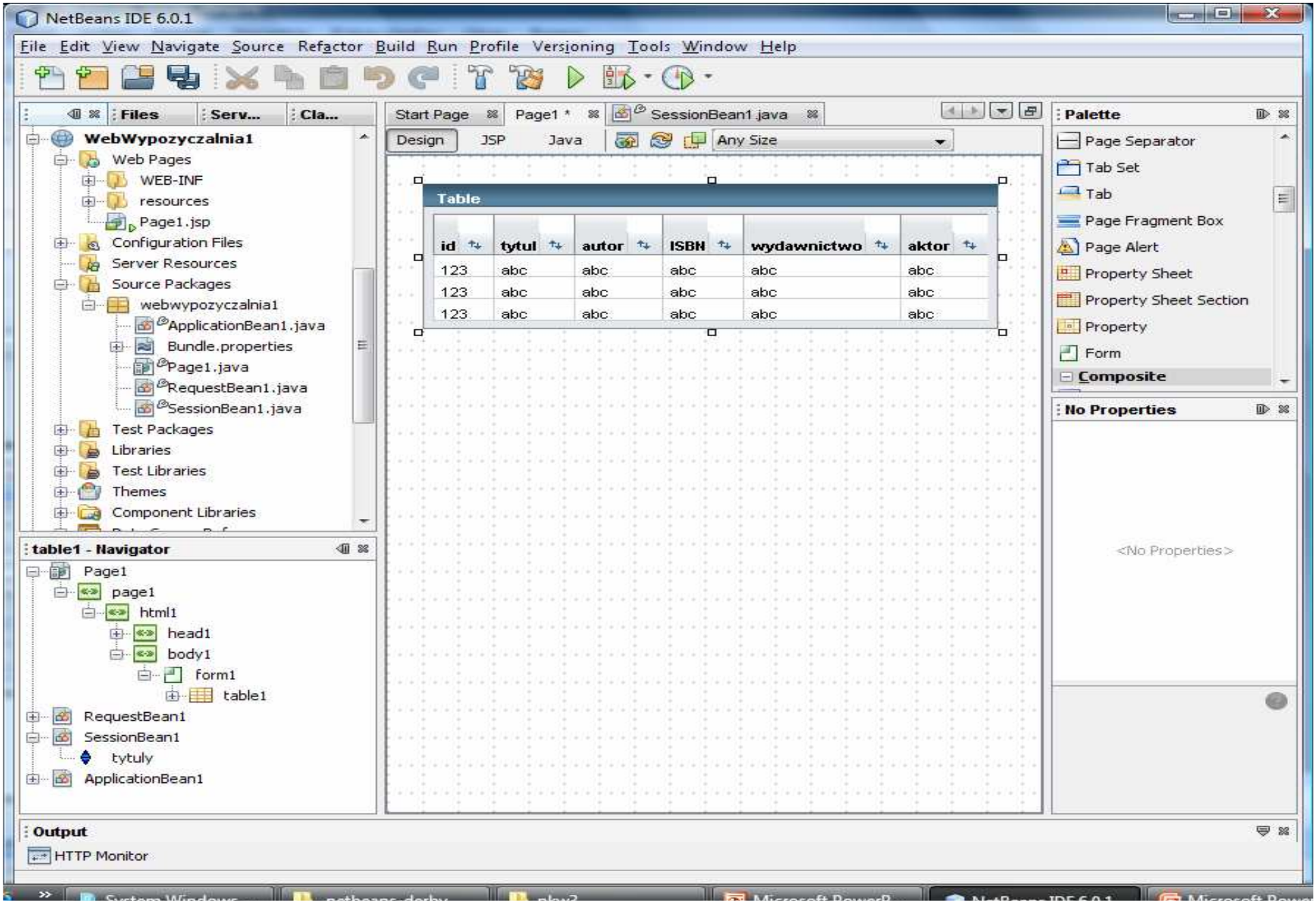
Wybór tablicy **tytuly** w okienku **Bind to Data (11)**



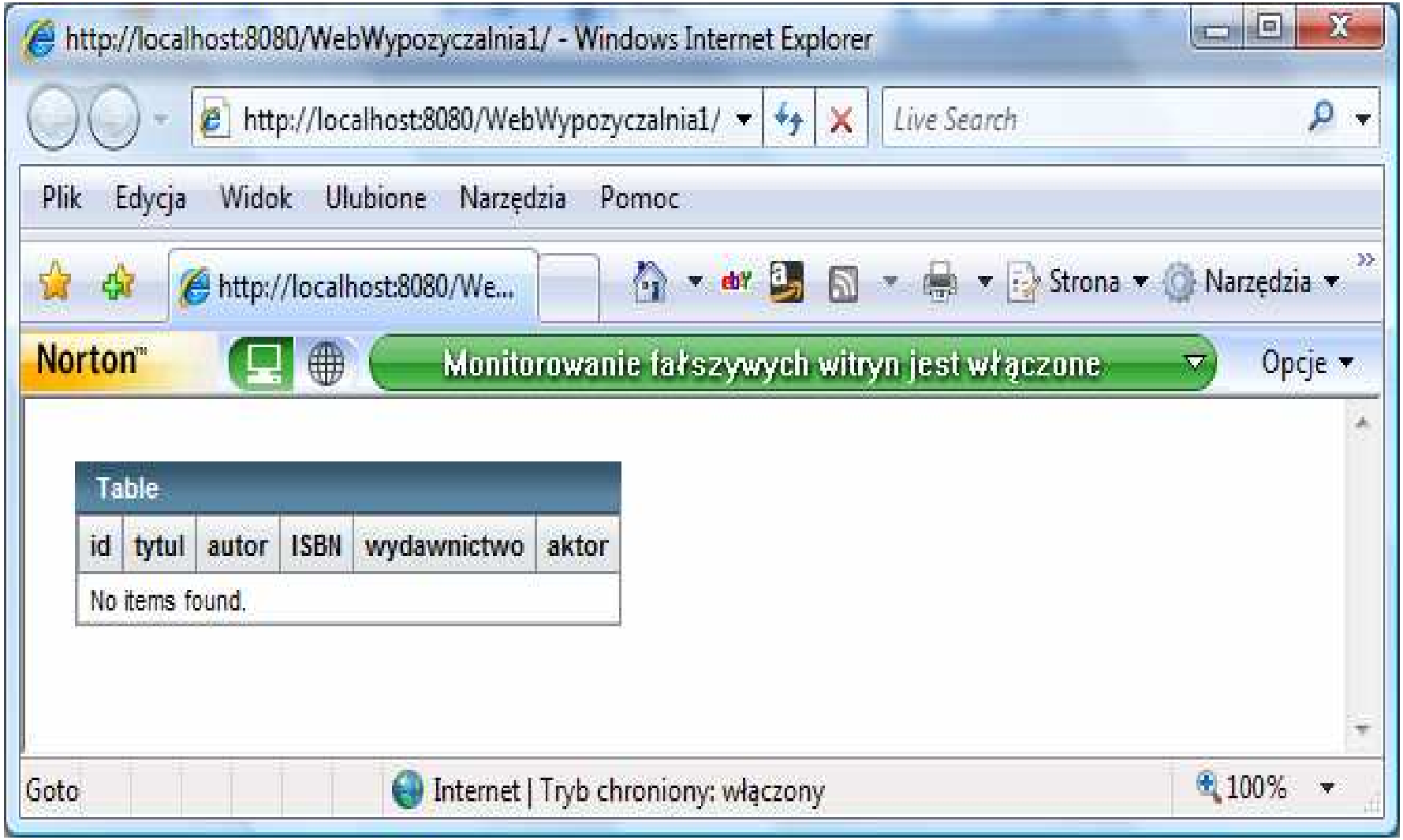
Wybór kolumn (pomijanie kolekcji ksiazka mapującej relację **OneToMany** oraz **MKsiazka** wykorzystywanej przez warstwę biznesową do gromadzenia danych o ksiazkach dla danego tytulu) (12)



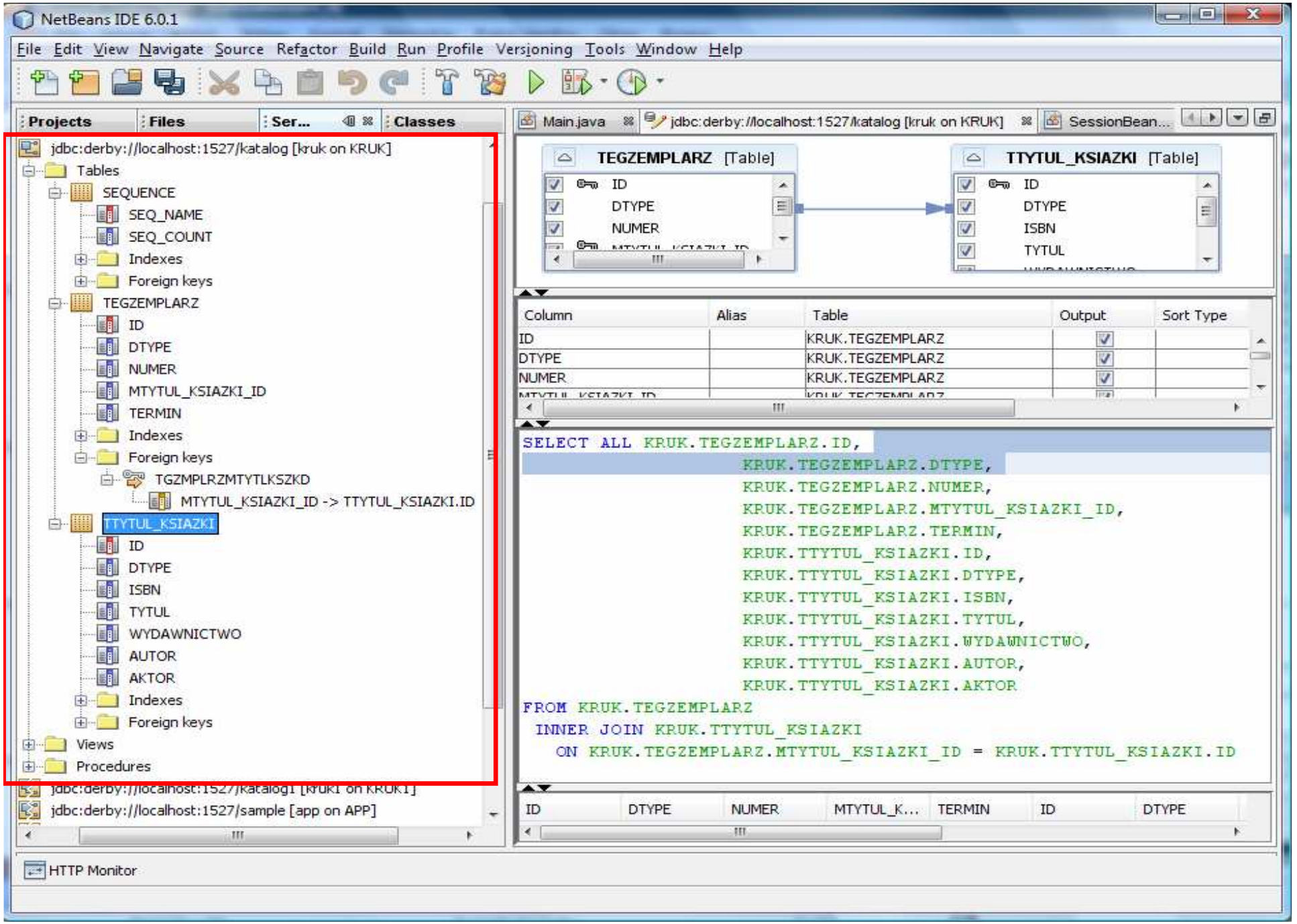
Efekt operacji Bind to Data (13)



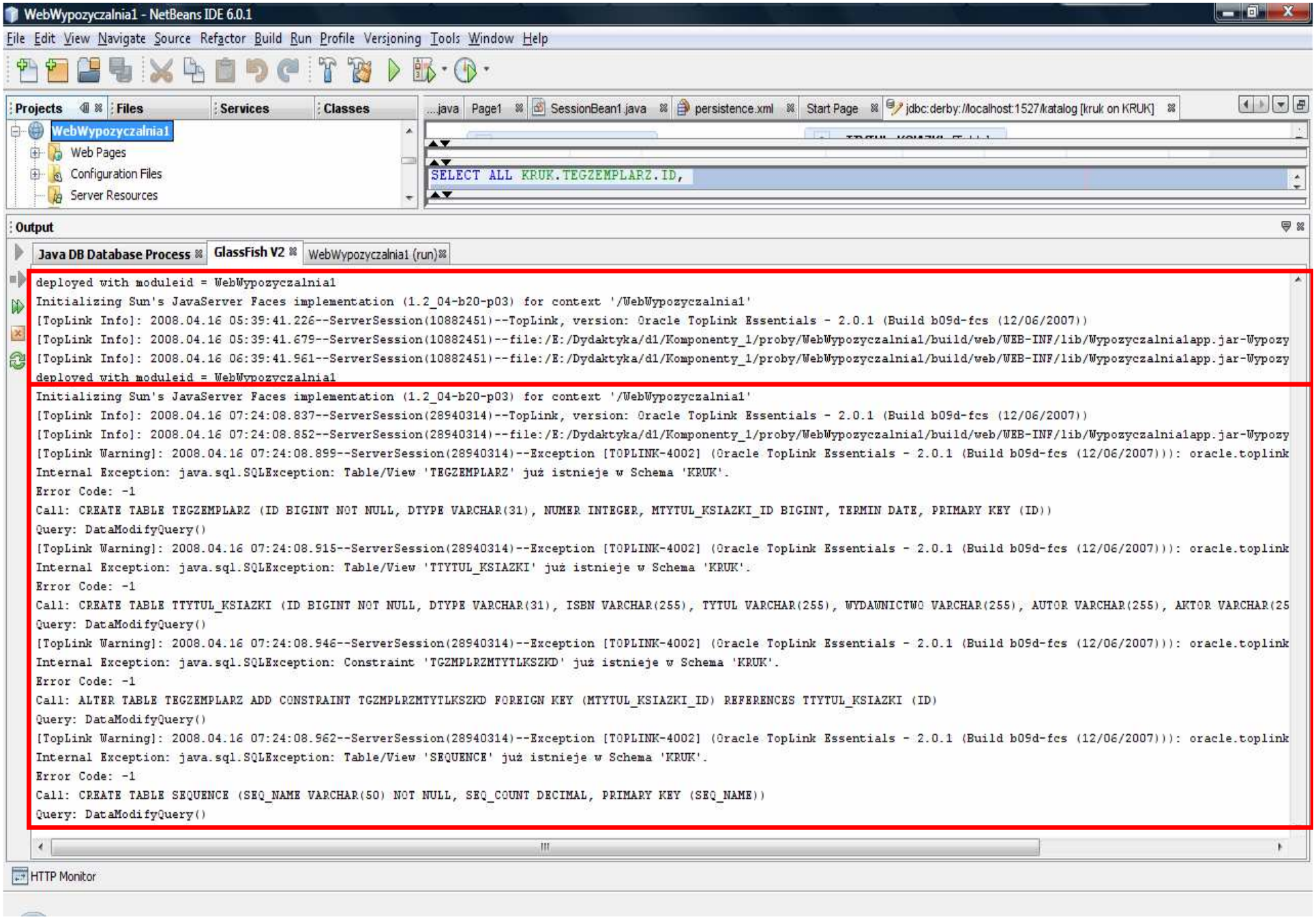
Uruchomienie aplikacji (14)



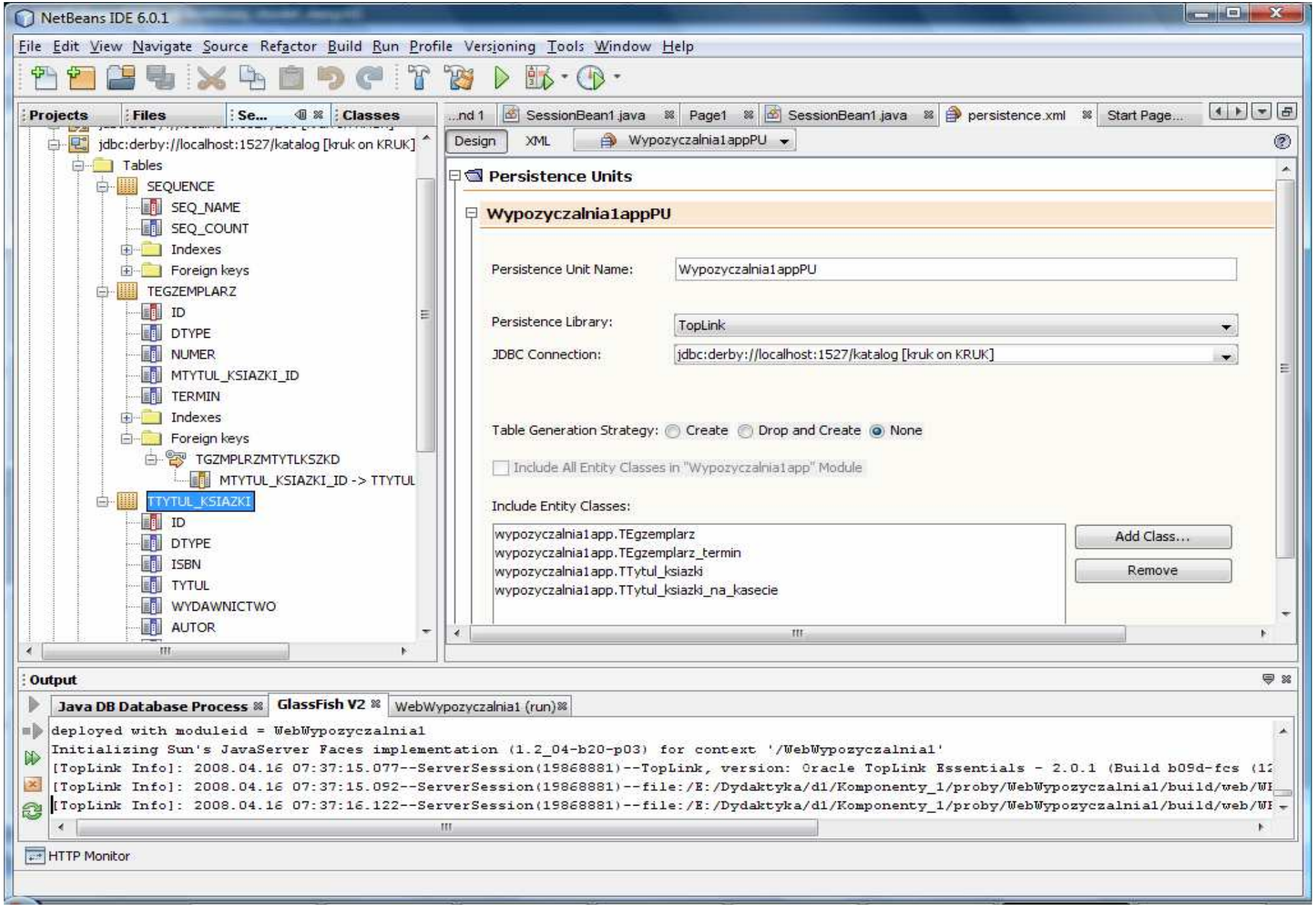
Wygenerowano tabele w bazie danych **katalog**, założonej wcześniej do utrwalania



Ponowne wywołanie aplikacji w trybie **Create (16)**



Ponowne uruchamianie aplikacji po założeniu tabel bez wykonywania skryptów SQL typu Create (17)



4. Tworzenie warstwy klienta, prezentacji cd

Integrowanie warstwy prezentacji i klienta wykonane w technologii **Java Server Faces**

- Wykonanie przypadku użycia „dodaj tytul”

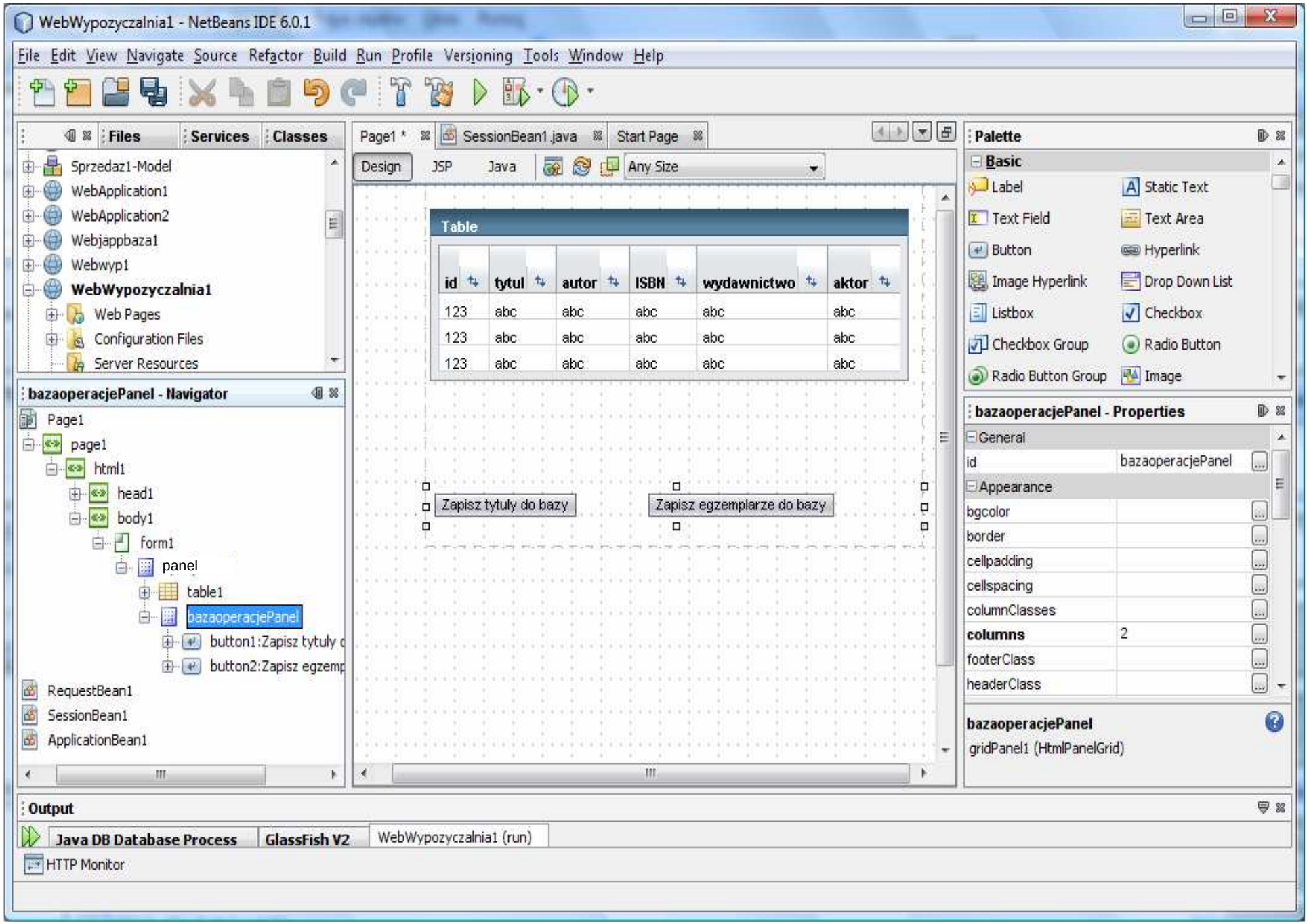
Tworzenie warstwy prezentacji dla przypadku użycia „dodaj tytul”(1)

- Slajd (1) - z palety **Layout** należy przeciągnąć komponent **Grid Panel** i nadać mu **id = panel** w okienku **Properties**. Przeciągnąć komponent **Table** do tego panela oraz kolejny **Grid Panel** i nadać mu **id = bazaoperacjePanel** oraz właściwość columns równą 2 w okienku **Properties**. Do zagnieżdżonego panela wstawić dwa przyciski typu **Button**. Nazwy widoczne w trybie **Design** na komponentach wpisać do właściwości **text** lub **title** w okienkach **Properties** tych komponentów – **komponenty można wstawiać z palety „przeciągając” je i „upuszczając” w oknie Navigator**.
- **Należy kliknąć prawym klawiszem myszy na wstawiane komponenty w Okienku Navigator lub Editor i następnie kliknąć na pozycję Add Binding Attribute (punkt 3, slajd 0)**
- Slajd(2) - wstawić kolejny **Grid Panel** zagnieżdżony w panelu **panel** i nadać mu **id = tytulyaplikacjaPanel** oraz columns=2 w okienku **Properties.** Wstawić pięć par komponentów **Label** i **Text Field**. Nazwy widoczne w trybie **Design** na komponentach wpisać do właściwości **text** lub **Title** w okienkach **Properties** tych komponentów.
- Slajd (4) – zmodyfikować zawartość panela **panel**. Wstawić **Grid Panel** do panela **panel**, nadać mu **id=bazaPanel,** przesunąć komponent **Table** do tego panela i nadać mu **id=bazatytuly, columns=2** i **title=Tytuly** (okno **Properties**)**.** Wstawić kolejny **Grid Panel** do panela **panel** i nadać mu **id=aplikacjadanePanel.** Przesunąć panel o **id= tytulyaplikacjaPanel** zagnieżdżony w panelu **panel** do panela **aplikacjadanePanel.** Wstawić kolejny **Grid Panel** do panela **aplikacjadanePanel,** nadać mu **id=aplikacjadane,** ustawić właściwość columns równą 2**,** przeciągnąć do niego komponent typu **DropDown List** i nadać mu **id = tytuly**. Wstawić kolejny **Grid Panel** zagnieżdżony w panelu **panel**, nadać mu **id = aplikacjaoperacjePanel**, ustawić właściwość columns równą 2 i przeciągnąć do niego komponent Button. Rozmieszczenie paneli ilustruje slajd (4). Uzupełnij właściwości **id, text** lub **Title** komponentów **Button** i **Text Field** w ich okienkach **Properties** zgodnie z slajdem (4).
- Napisać kod wg slajdu (3) w klasie **SessionBean1**. Wg slajdu (5) wykonać operację **Bind to Data**, łącząc komponent **tytuly** typu **DropDown List** z tablicą **Option tytuly_[].**

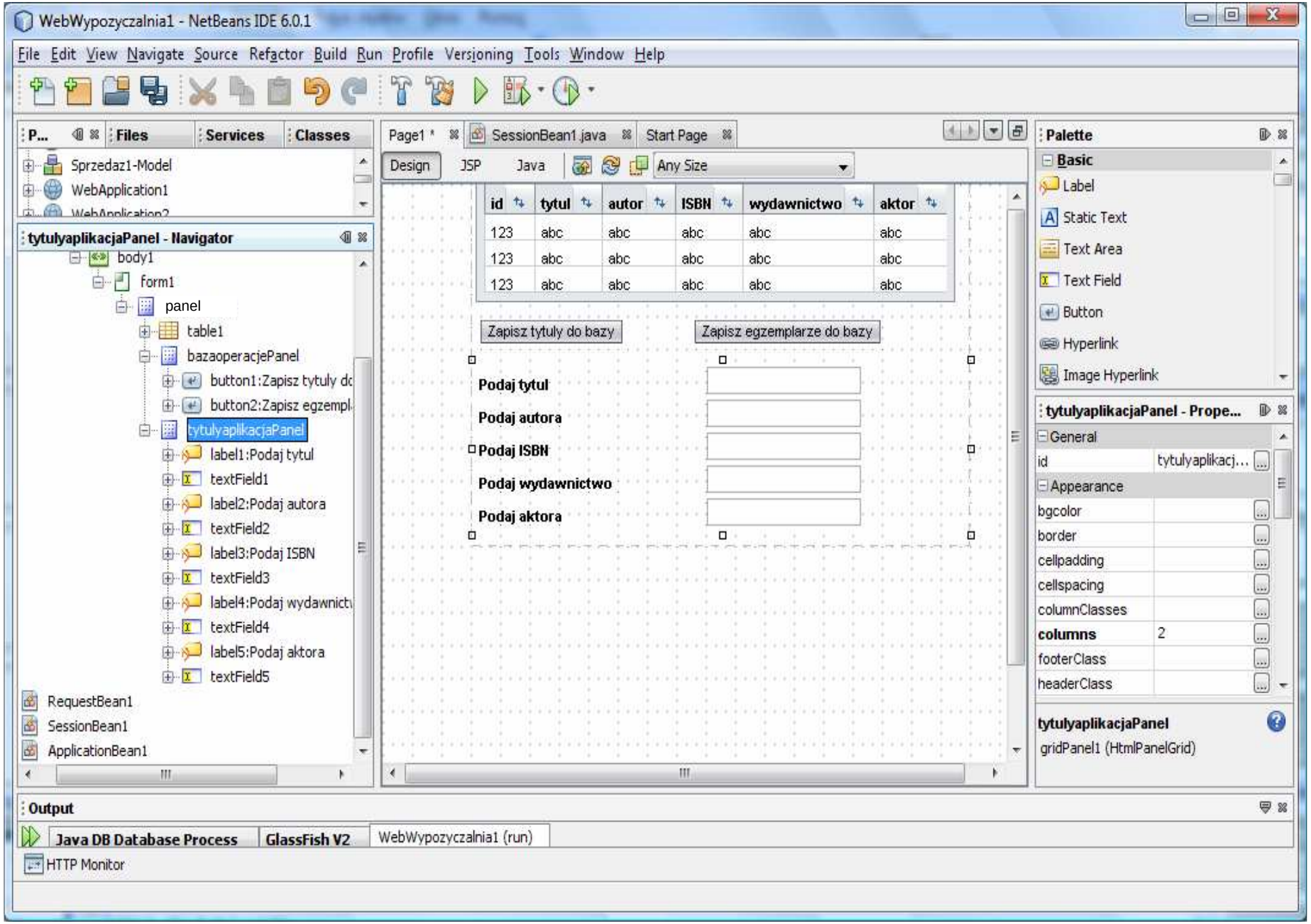
Tworzenie warstwy prezentacji dla przypadku użycia „dodaj tytul”(2)

- Napisać kod w klasie w **SessionBean1** (slajdy (6) i (7)), oznaczony ramką – oznacza ono **połączenie obiektu zdalnego** w postaci obiektu klasy **SessionBean1 do warstwy biznesowej (obiekt aplikacja).** Slajd (7) pokazuje metodę **przygotuj_tytuly(),** która pobiera dane o tytułach z obiektu **aplikacja (punkt 1)** i zapisuje je do tablicy **Option tytuly_[] (punkty 2 i 3).**
- W klasie **Page1** napisać kod wg slajdów (8) i (9). Na slajdzie (8) podano kod obsługi zdarzenia klikania na przycisk **dodajtytul_action().** Kod ten służy do pobrania danych z komponentów panela **tytulyaplikacjaPanel (punkty 1, 2) –** w punkcie 1 ustala się zawartośc pierwszego elementu tablicy dane, określające, jaki typ obiektu ma powstać 0 - obiekt typu **TTytul_ksiazki**, 3 - obiekt typu **TTytul_ksiazki_na_kasecie, w punkcie 2** ustala się zawartość tablicy **dane** (obiekt typu **Transfer Object**)**,** zapisu danych o tytułach w aplikacji metodą **dodaj_tytul** (w warstwie biznesowej) **(punkt 3)** i wyświetlenia danych o tytułach w komponencie **tytuly** typu **DropDownList** za pośrednictwem tablicy **Option tytuly []** metodą **przygotujtytuly() – (punkt 4).** Slajd (9) pokazuje zawartość metody **prerender,** która zawsze jest wywoływana przy odświeżaniu zawartości strony w fazie **Response** przetwarzania strony **Java ServerFaces**. W tej metodzie wpisuje się obiekty typu String z pustym łańcuchem do komponentów typu Text Field oraz odświeża się zawartość komponentu Table (bazatytuly)
- Slajd (10) pokazuje stan aplikacji po uruchomieniu programu
- Zgodnie ze slajdem (11) napisać w klasie **SessionBean1** kod zapisujący dane aplikacji do bazy danych metodą **addTTytul_ksiazkis** obiektu klasy **TTytul_ksiazkiController** – **punkt 2**. Na slajdzie (13) pokazano diagram sekwencji tej operacji. Zapis do bazy następuje po wywołaniu metody **findTTytul_ksiazkis** z klasy **TTytul_ksiazkiController** (slajd (12)), która zapobiega naruszeniu więzów integralności tabeli **TTytul_ksiazki – punkt 1**. Na slajdzie (14) pokazano działającą aplikację, która zapisuje dane o tytułach w aplikacji, wyświetla dane przechowywane w aplikacji za pomocą komponentu **tytuly** typu **DropDown List**, zapisuje te dane do bazy danych i wyświetla zawartość tabeli **TTytul_ksiazki** za pomocą tablicy **TTytul_ksiazki tytuly []** w komponencie typu **Table**
- Na slajdzie (15) pokazano zawartość danych bezpośrednio w tabeli **TTytul_ksiazki** w bazie danych (zakładka **Service**) – po kliknięciu prawym klawiszem myszy na nazwę tabeli **TTytul_ksiazki** wywołano z wyskakującego menu pozycję **View Data.**

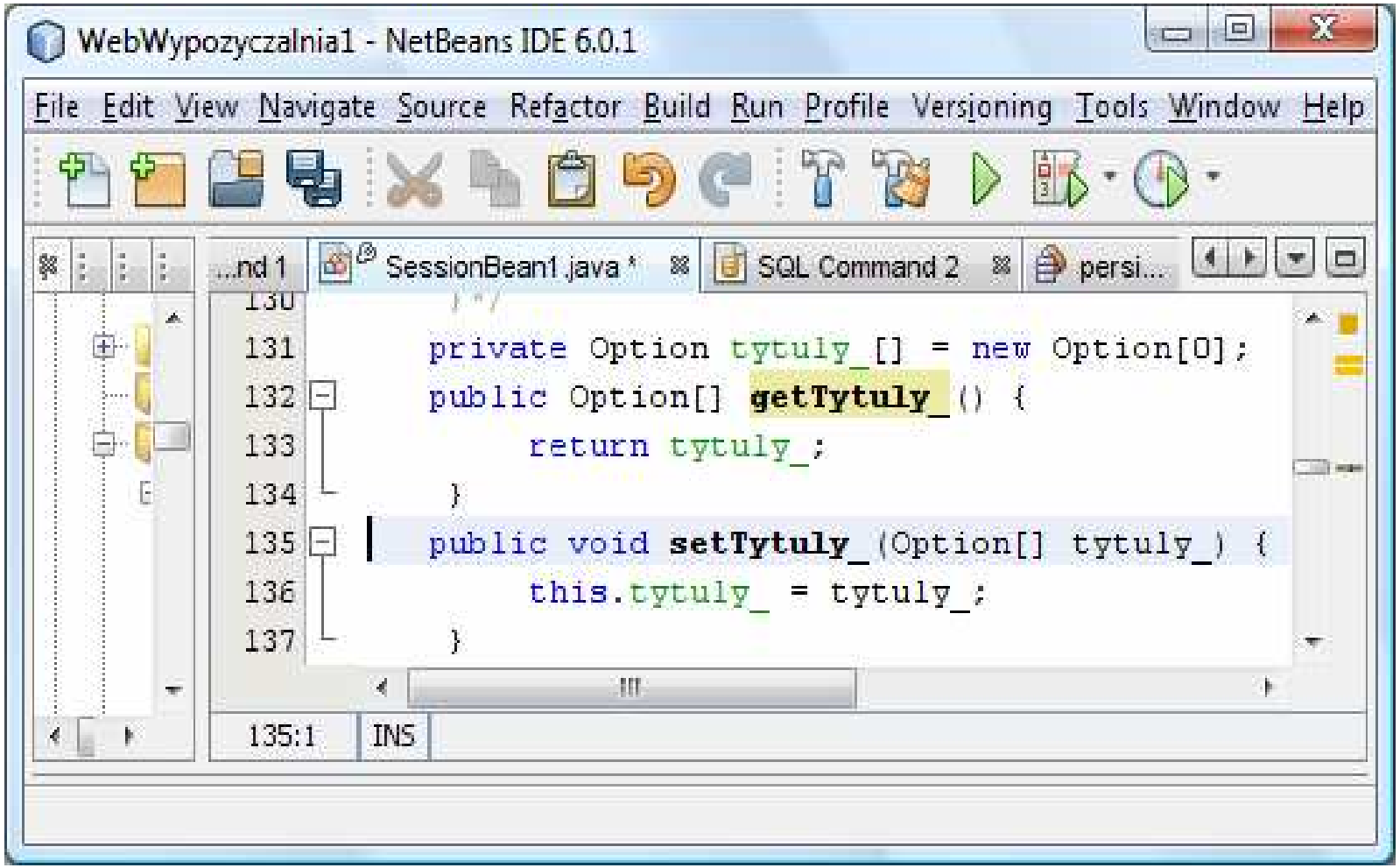
Wstawianie komponentów (1)



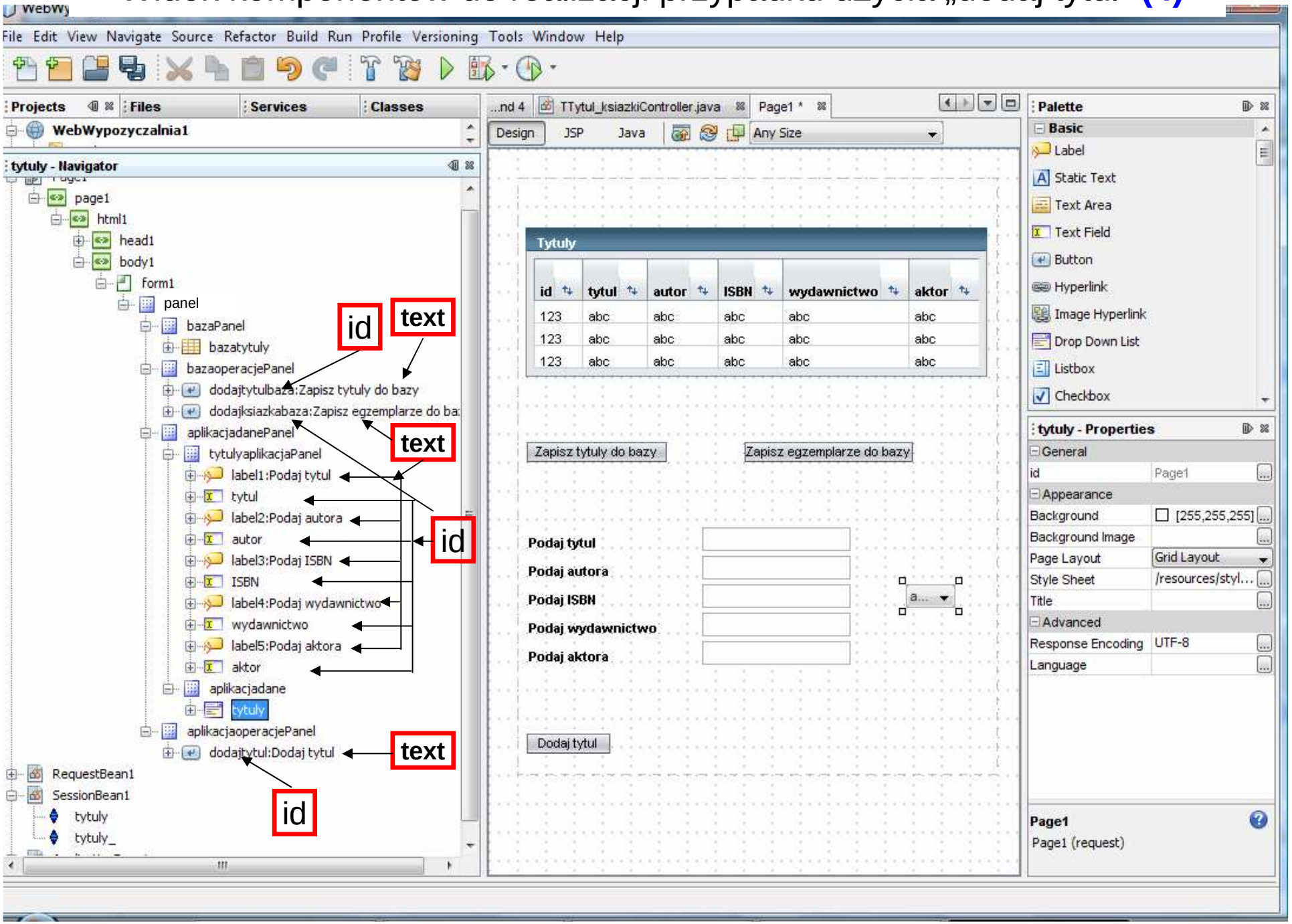
Wstawianie komponentów umieszczanych na komponencie **Grid Panel** z palety **Layout (2)**



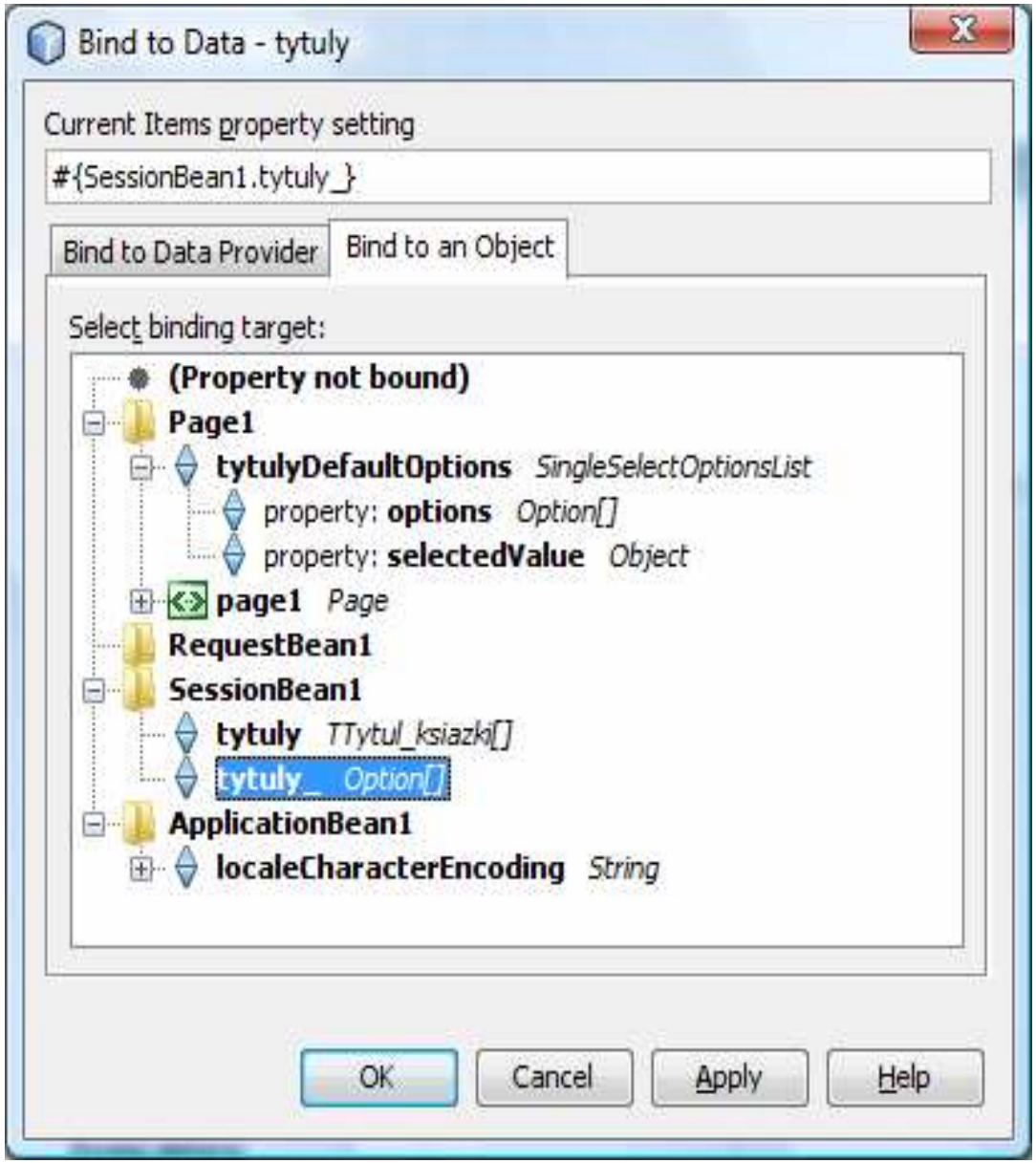
Dodanie tablicy **Option tytuly_[ ]** w klasie sesji **SessionBean1** do wykonania operacji **Bind to Data** dla komponentu **tytuly** typu **DropDownList** w celu wyświetlenia tytułów przechowywanych w aplikacji (3)



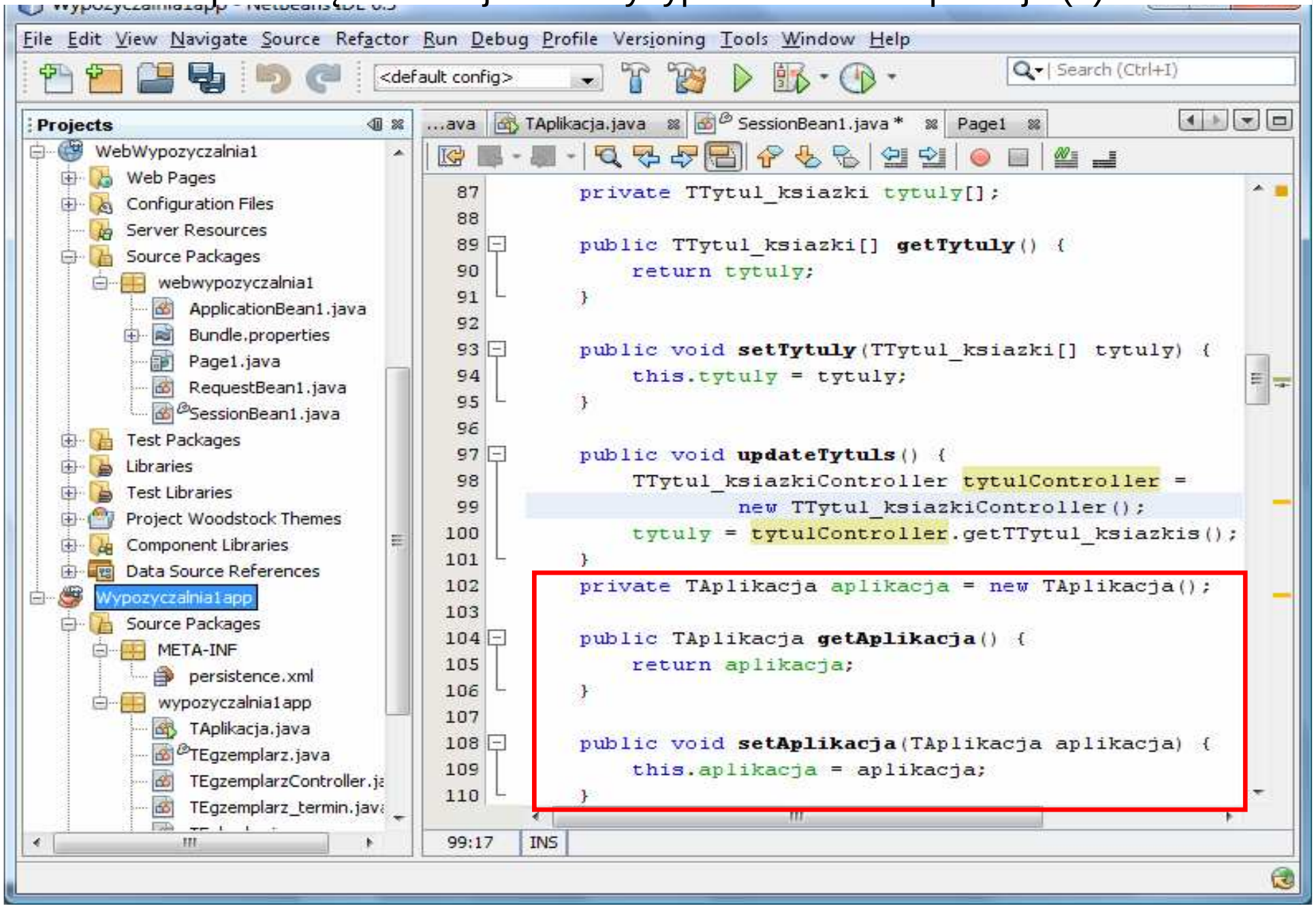
Widok komponentów do realizacji przypadku użycia „dodaj tytul” **(4)**



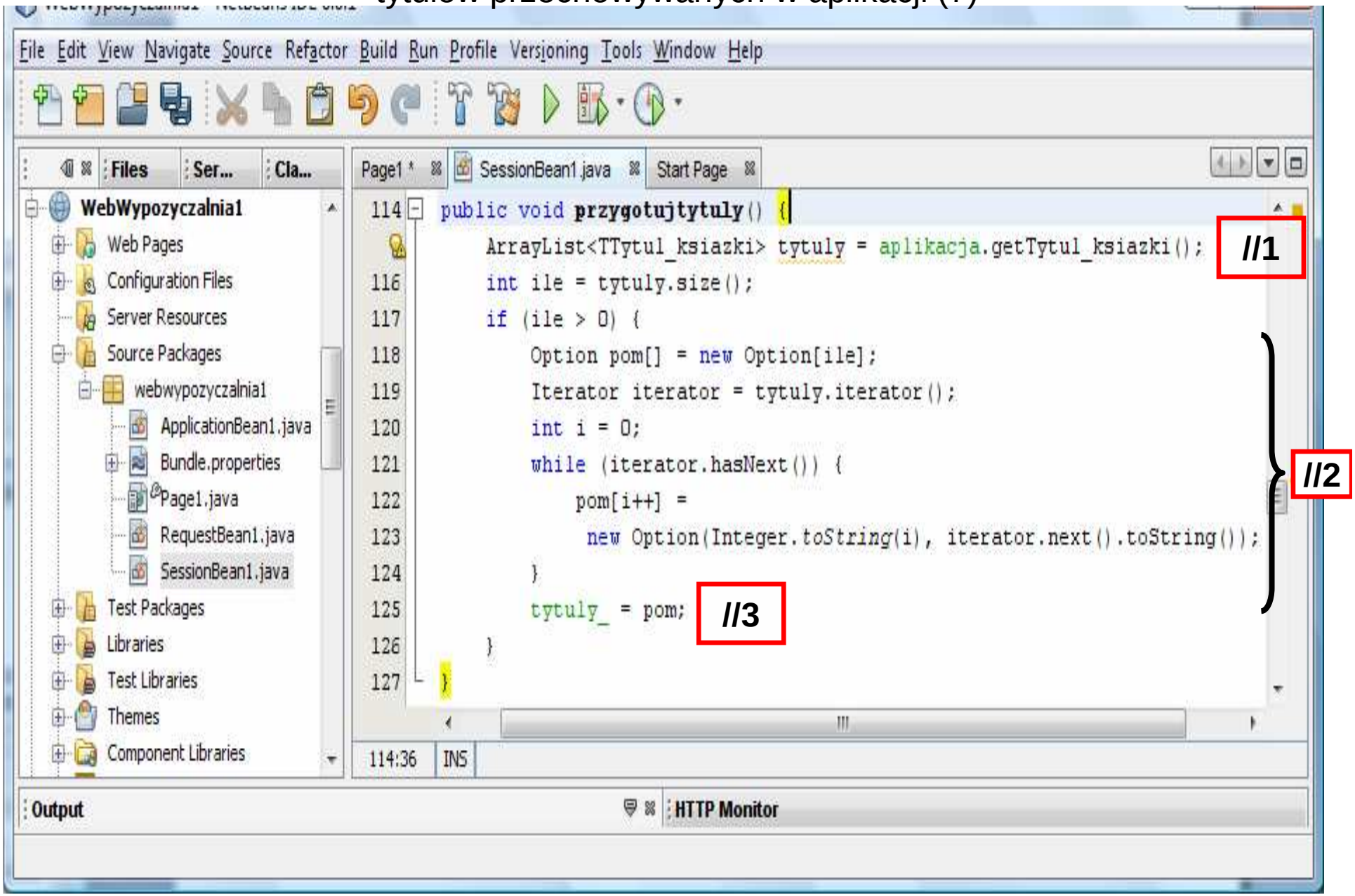
Operacja **Bind toData** dla komponentu **tytuly** typu **DropDown List (5)**



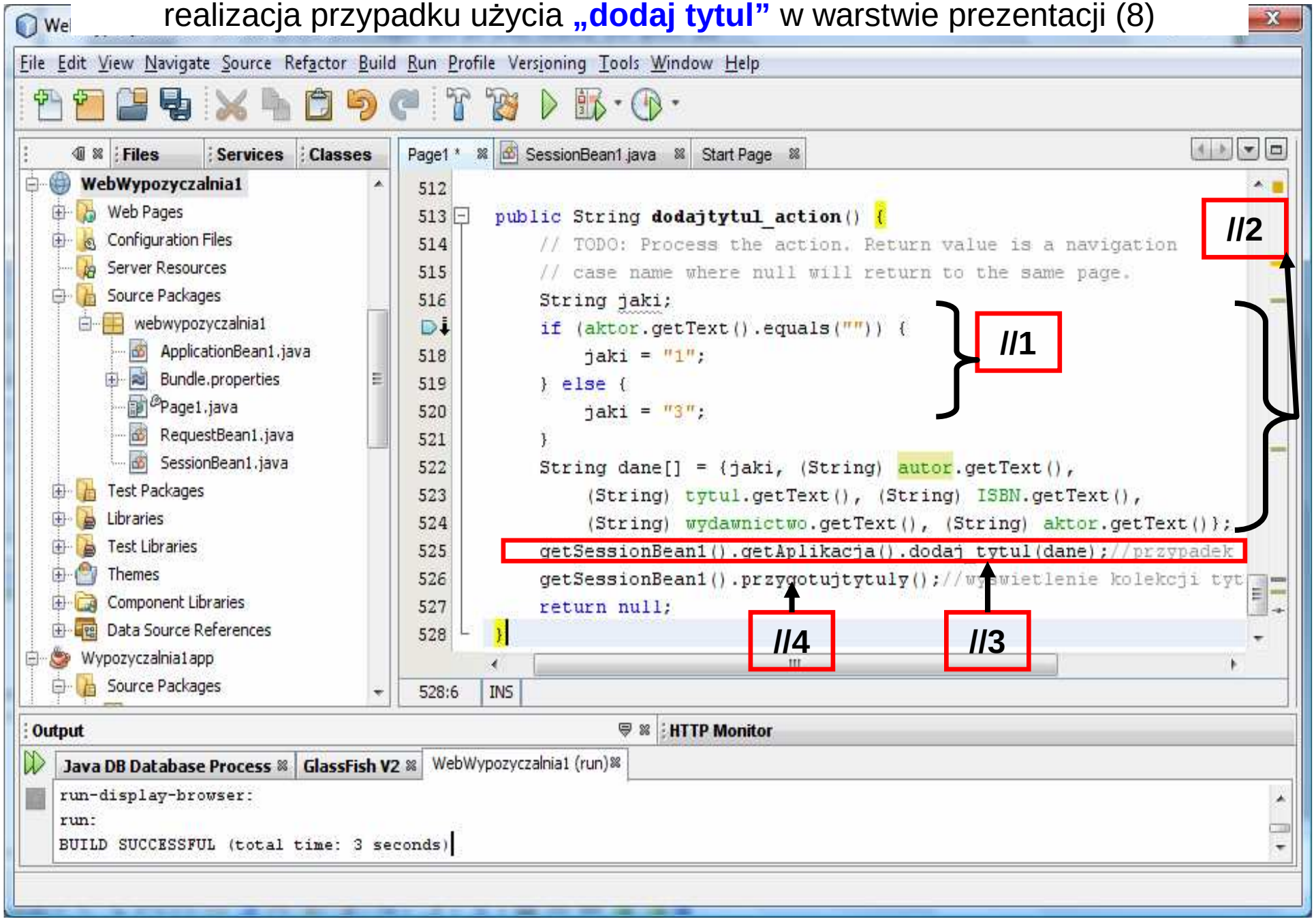
Tworzenie kodu do połączenia warstwy prezentacji z warstwą biznesową za pomocą referencji do klasy typu Fasada –TAplikacja (6)



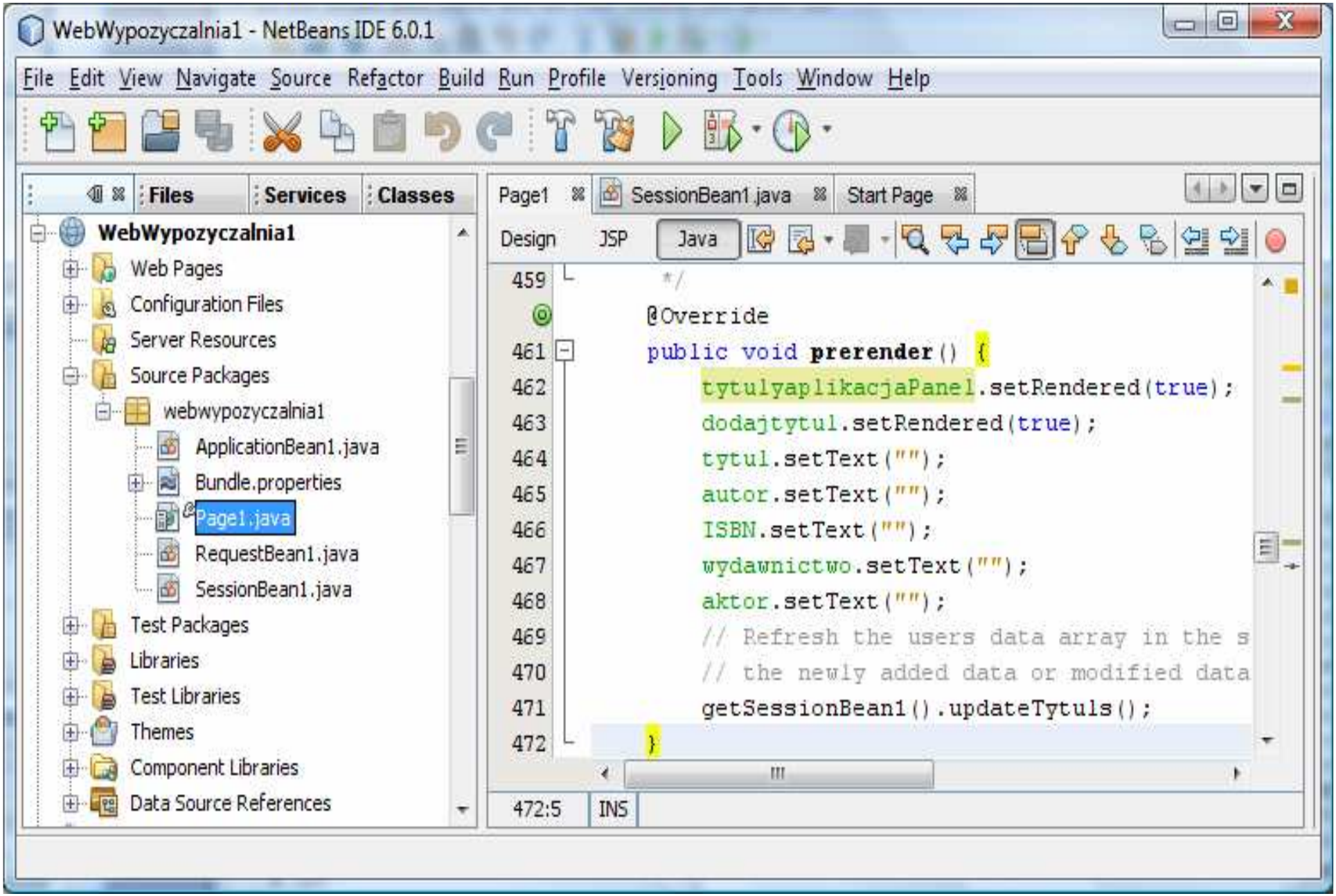
Przepisywanie zawartości kolekcji tytułów do tablicy **Option tytuly_[ ]** w celu wykonania operacji **Bind to Data** z komponentem **tytuly** typu **DropDown List** w celu wyświetlania tytułów przechowywanych w aplikacji (7)



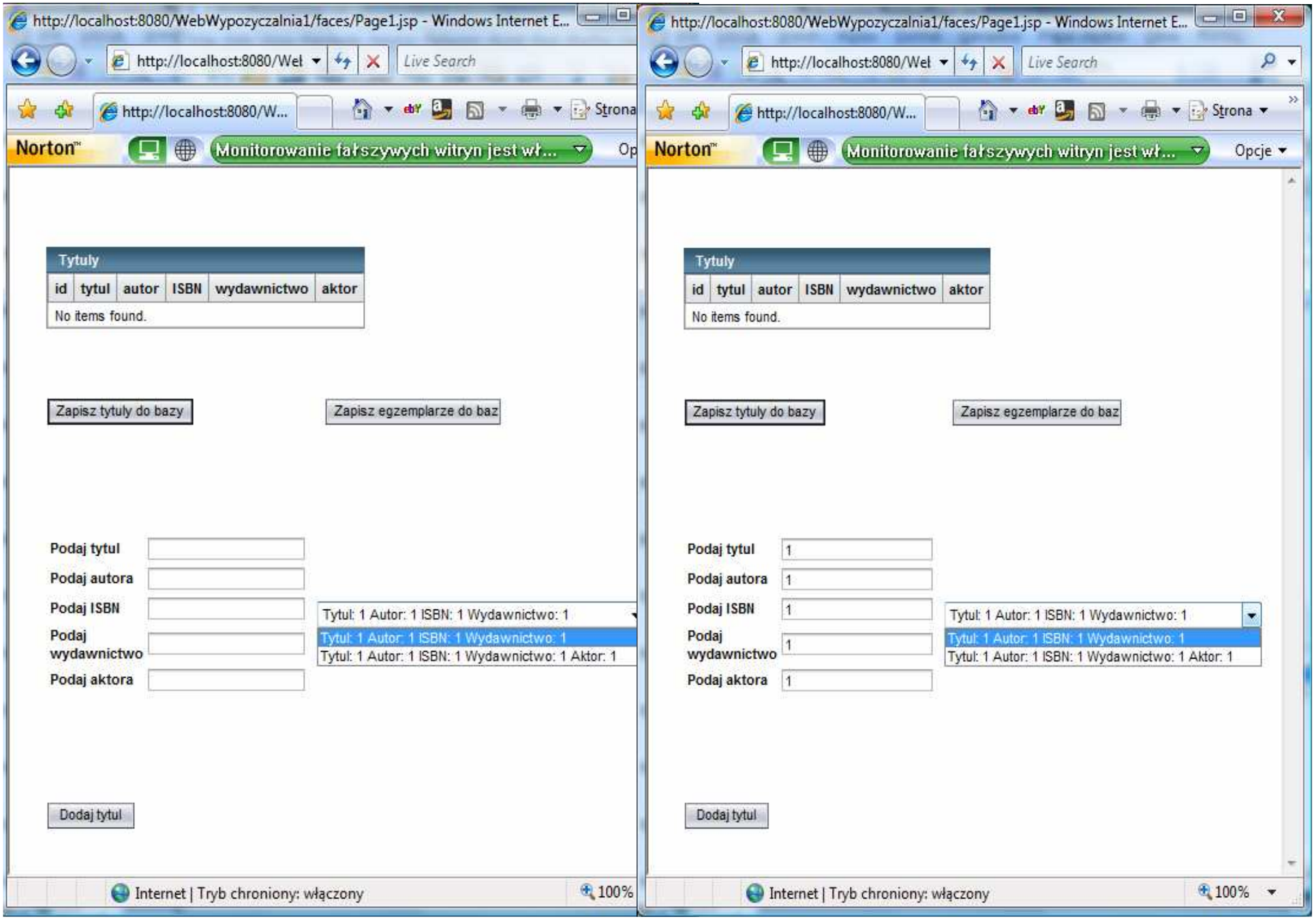
Obsługa zdarzenia klikania w przycisk **„dodajtytul”** w pliku **Page1.java** – realizacja przypadku użycia **„dodaj tytul”** w warstwie prezentacji (8)



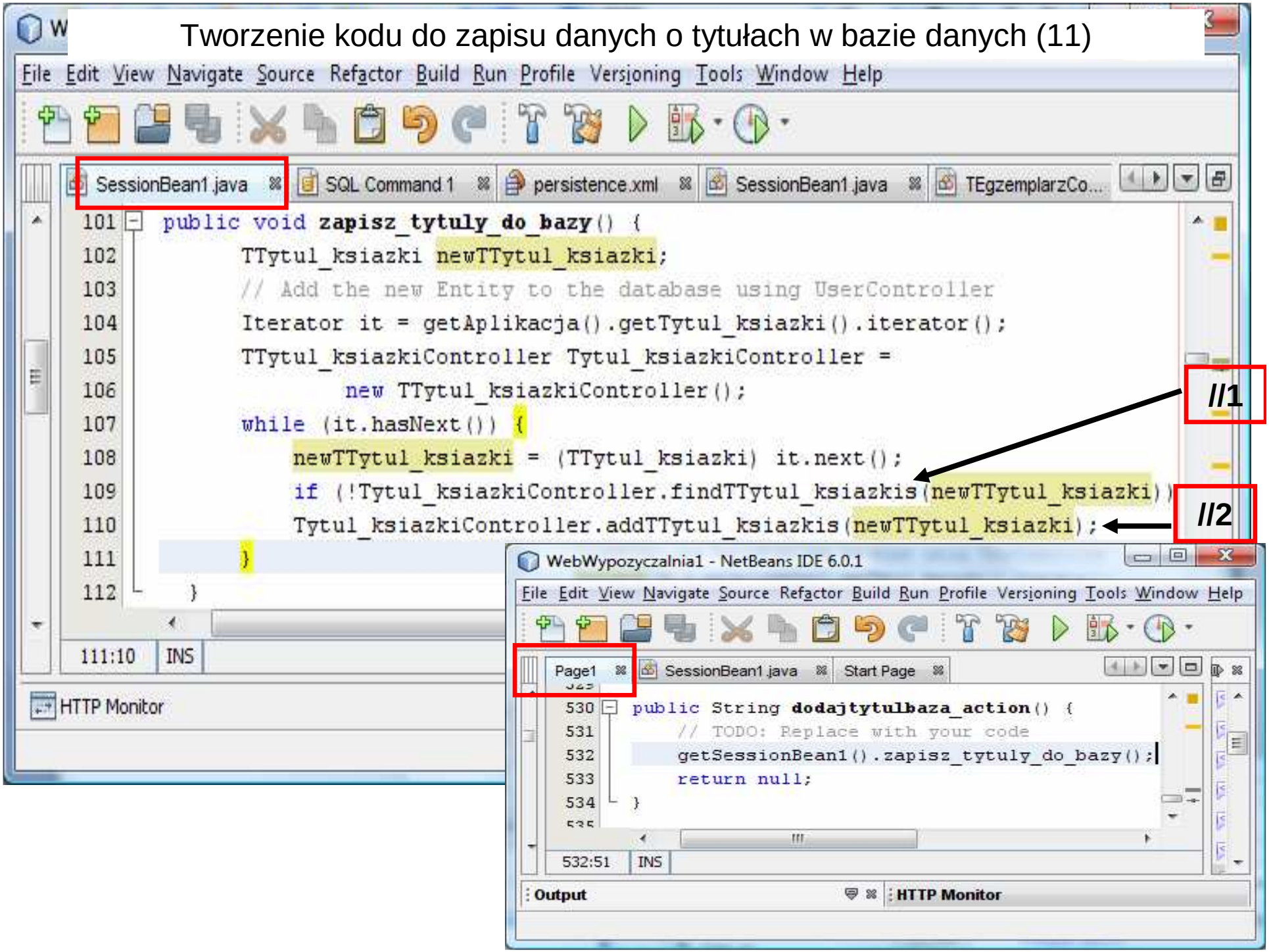
Ustawienie komponentów w metodzie **prerender (9) – jest wywoływana w fazie Response (postback) np. po zapisaniu danych do bazy danych**



Działanie przypadku użycia po stronie aplikacji, bez użycia bazy danych (10)



Tworzenie kodu do zapisu danych o tytułach w bazie danych (11)



Nowa metoda w klasie TTytul_ksiazkiController do sprawdzania, czy obiekt TTytul_ksiazki w relacji wiele do jeden zostanie zapisany ponownie naruszając więzy integralności referencyjnej tabeli TTytul_ksiazki (12)

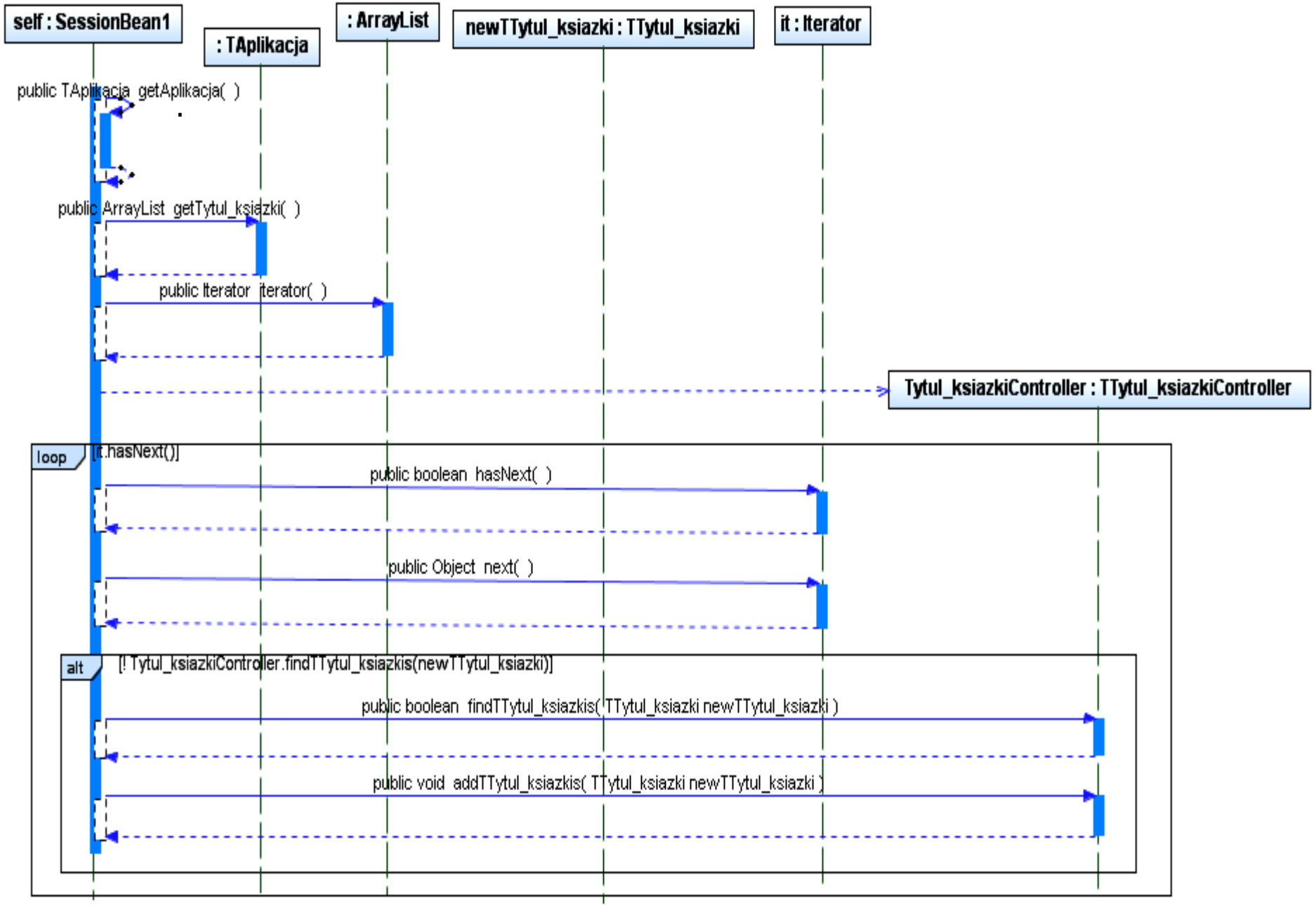
```
public boolean findTTytul_ksiazkis(TTytul_ksiazki TTytul_ksiazki)
{
 EntityManager em = getEntityManager();
 try {
       return  em.find(TTytul_ksiazki.class, TTytul_ksiazki.getId())!=null;
      }
     finally {
         em.close();
     }
 }
```

Uzupełniono kod konstruktora klasy TTytul_ksiazki– wstawia klucz główny równy 0 do danej, która jeszcze nie została zapisana do bazy danych. Wartość 0 klucza głównego jest wartością nie należąca do dziedziny wartości kluczy głównych-wtedy można przekazać taki obiekt do metody find w klasie EntityMaqnager

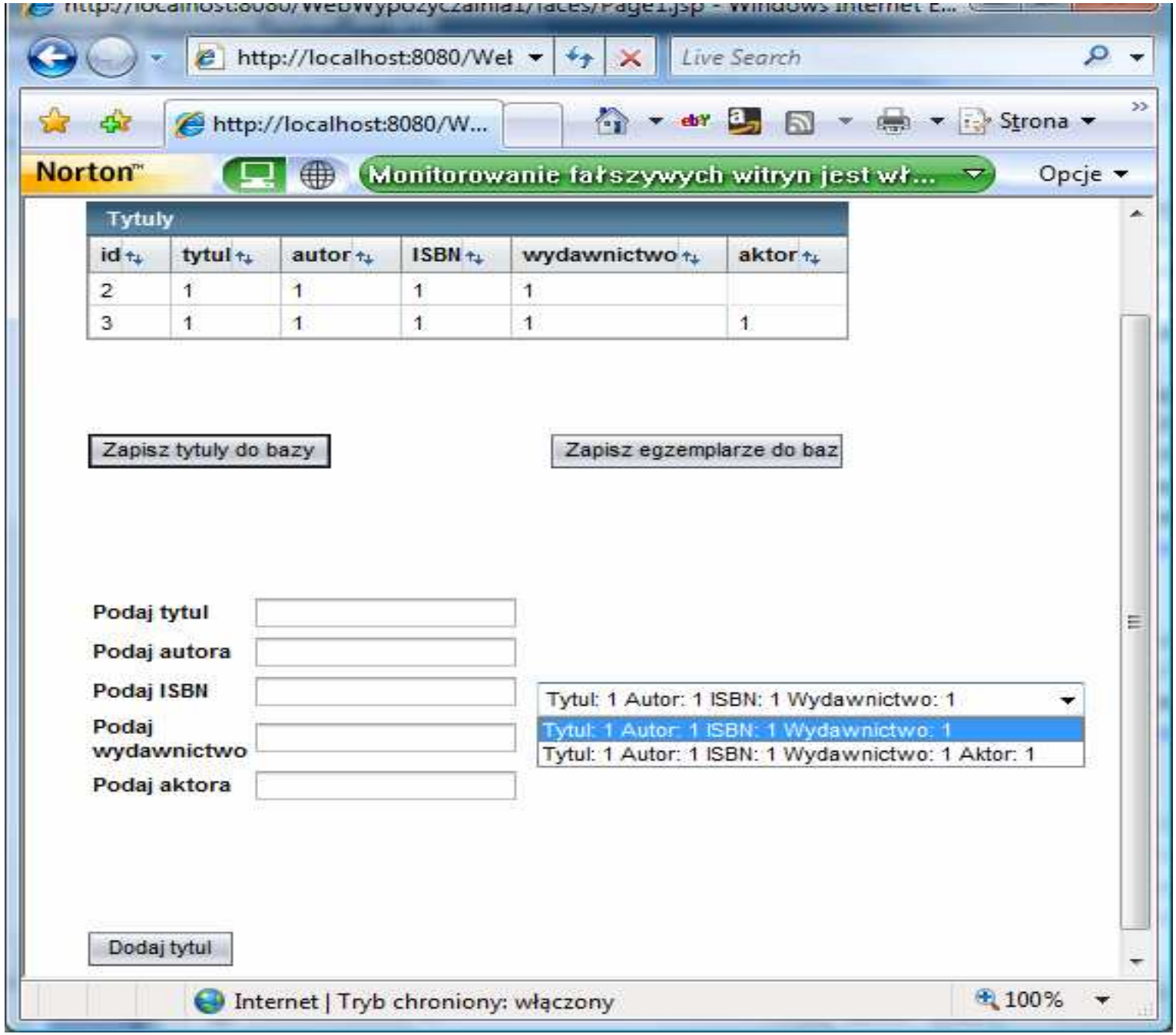
```
public  TTytul_ksiazki ()
          { id = new Long(-1); //id=null; - wersja 2 w lab7_8
          }
```

Zapis danych do bazy tytułów metodą add obiektu TTytul_ksiazkiController (13)

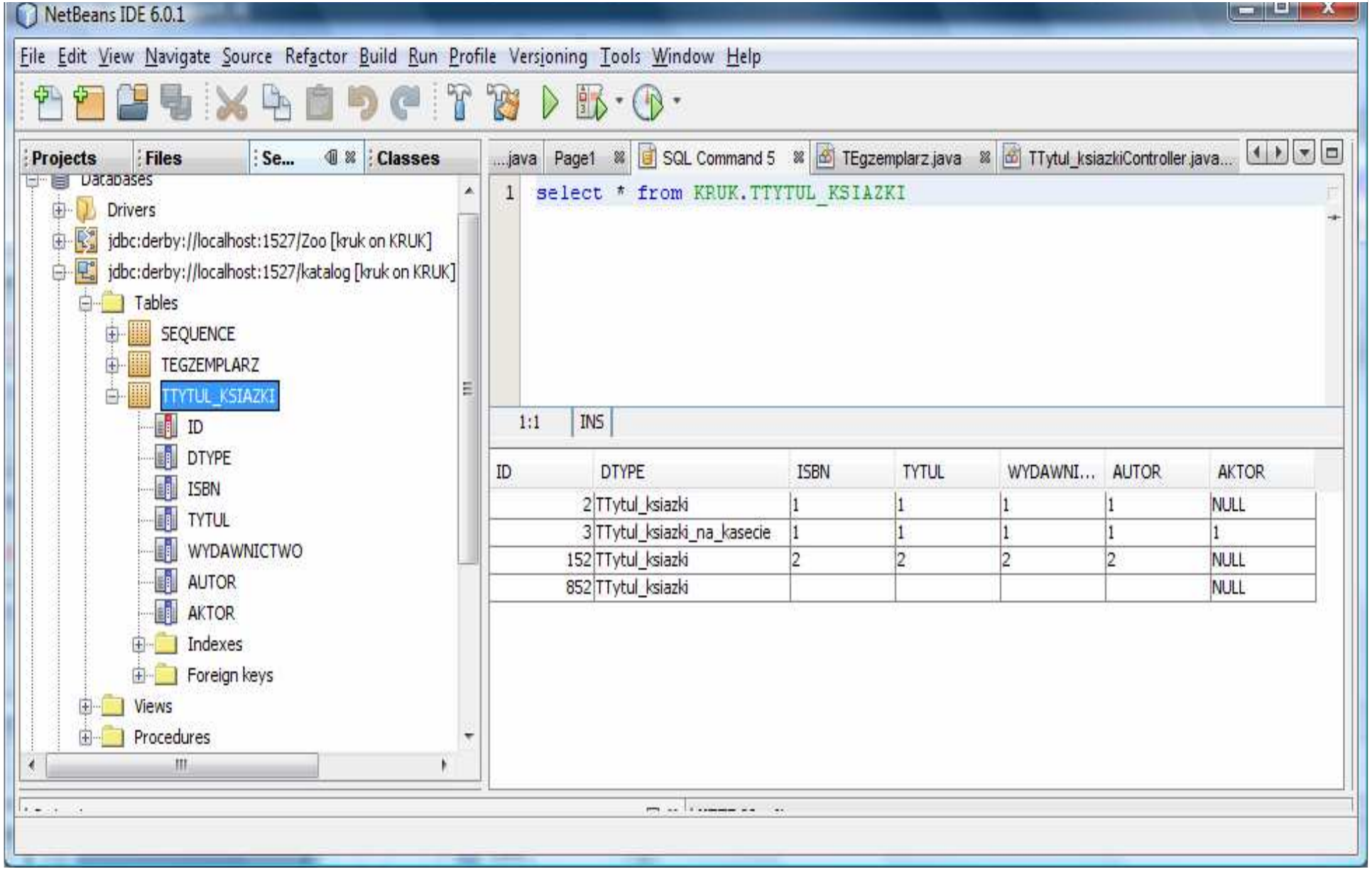
public void SessionBean1::zapisz_tytuly_do_bazy()



Uruchomienie aplikacji (14)



Widok bazy danych w zakładce **Services** – wykonanie operacji **ViewData** dostępnej z wyskakującego menu po naciśnięciu tabeli **TTytul_ksiazki** prawym klawiszem myszy (15)



5. Tworzenie warstwy klienta, prezentacji cd

Integrowanie warstwy prezentacji i klienta wykonane w technologii **Java Server Faces**

- Wykonanie przypadku użycia „dodaj egzemplarz”

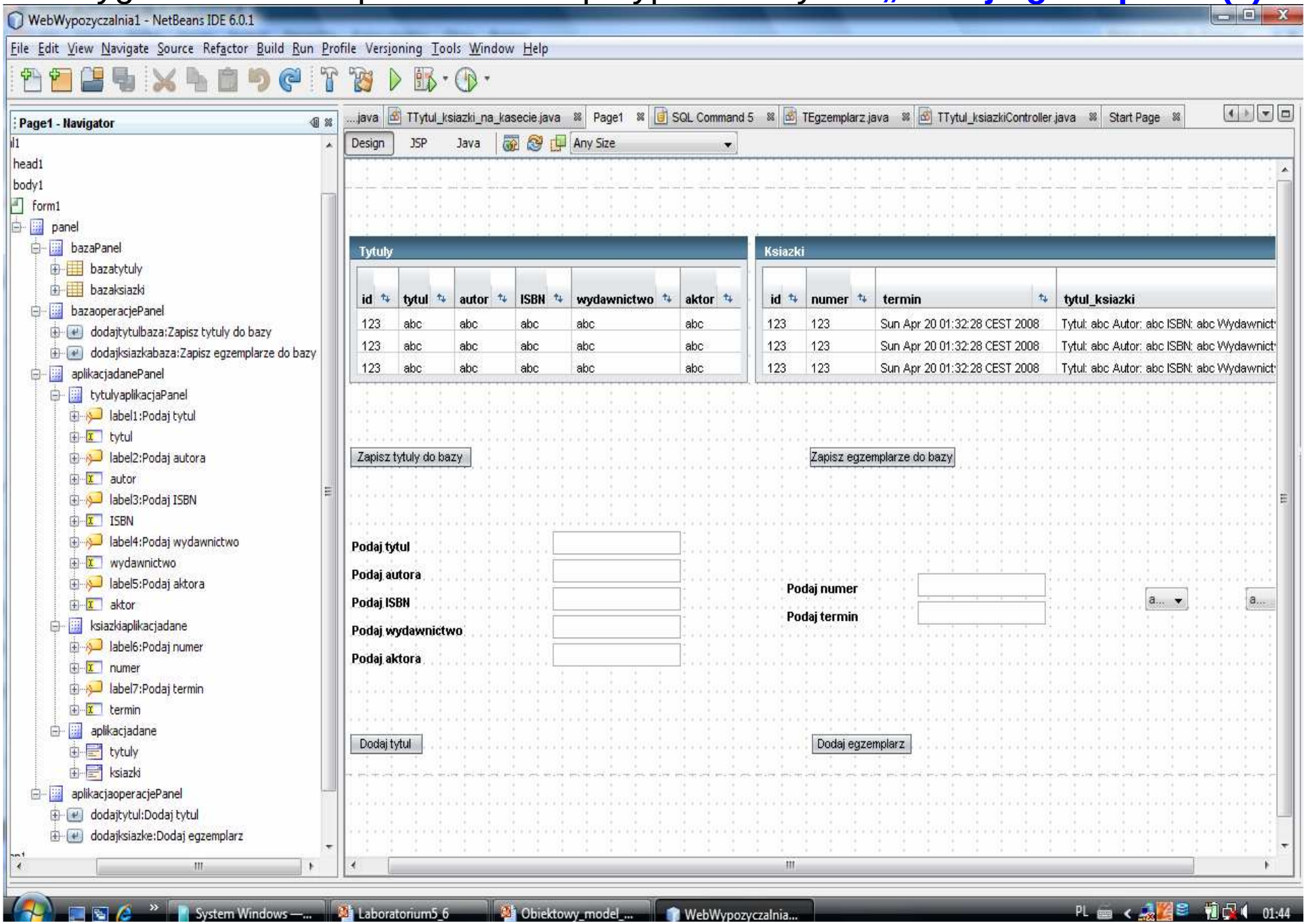
Tworzenie warstwy prezentacji dla przypadku użycia „dodaj egzemplarz” (1)

- Slajd (1) – uzupełnić komponenty strony **Page1.jsp** wg okienka **Navigator** ze slajdu (1) – **komponenty można wstawiać z palety „przeciągając” je i „upuszczając” w oknie Navigator**. Reprezentuje on zawartość strony Page1.jsp. Szczegółowo zawartość każdego z komponentów typu **Grid Panel** pokazano na slajdach: 2,3,4,5. Wg slajdu (2) należy przeciągnąć komponent **Table** do panela **bazaPanel**, nadać **id= bazaksiazki** oraz **title=Ksiazki**. Slajd (3) pokazuje nowy komponent typu **Grid Panel,** który ma **id=ksiazkiaplikacjaPanel,** jego właściwość **columns** ma być ustawiona na 2 i jest zagnieżdżony w panelu **aplikacjadanePanel.** Wg slajdu (4) wstawiono do niego dwie pary komponentów typu **Label** i **Text Field**. Ustawić właściwości **id** oraz **text** tych komponentów w ich okienkach **Properties** wg slajdu (4). Panel **aplikacjadanePanel** ma właściwość **columns** ustawioną na 3. Zgodnie ze slajdem (5) należy przeciągnąć komponent **DropDown List** do panela **aplikacjadane** (zagnieżdżonego w panelu **aplikacjadanePanel**) i nadać mu **id=ksiazki**. Należy przeciągnąć komponent **Button** do panela **aplikacjaoperacjePanel** i nadać mu **id** oraz **text** wg slajdu (5).
- **Należy kliknąć prawym klawiszem myszy na wstawiane komponenty w Okienku Navigator lub Editor i następnie kliknąć na pozycję Add Binding Attribute (punkt 3, slajd 0)**
- Napisać kod wg slajdu (6) w klasie **SessionBean1**. Slad (7) reprezentuje metodę **updateKsiazkis(),** która umożliwia z pobieranie danych z tabeli **TEgzemplarz** z bazy danych, zapisanych do tablicy **TEgzemplarz ksiazki[]** (slajd(6)) do bindowania z komponentem **bazaksiazki** typu **Table,** Wykonać operację **Bind to Data**, łącząc komponent **bazaksiazki** typu **Table** z tablicą **TEgzemplarz ksiazki[]** - slajd (8) pokazuje stan komponentów po tej operacji. Napisać kod wg slajdu (9) w klasie **SessionBean1**, wpisujący dane do tabeli **Option ksiazki_[]** dla wybranego tytułu książki z listy tytułów książek **(tytuly** typu **DropDown List)** metodą **przygotujksiazki(TTytul_ksiazki tytul))** z warstwy biznesowej**.** Wykonać operację **Bind to Data**, łącząc komponent **ksiazki** typu **DropDown List** z tablicą **Option ksiazki_[]** – slajd (10). Slajd (11) pokazuje stan komponentów strony po operacji „bindowania”

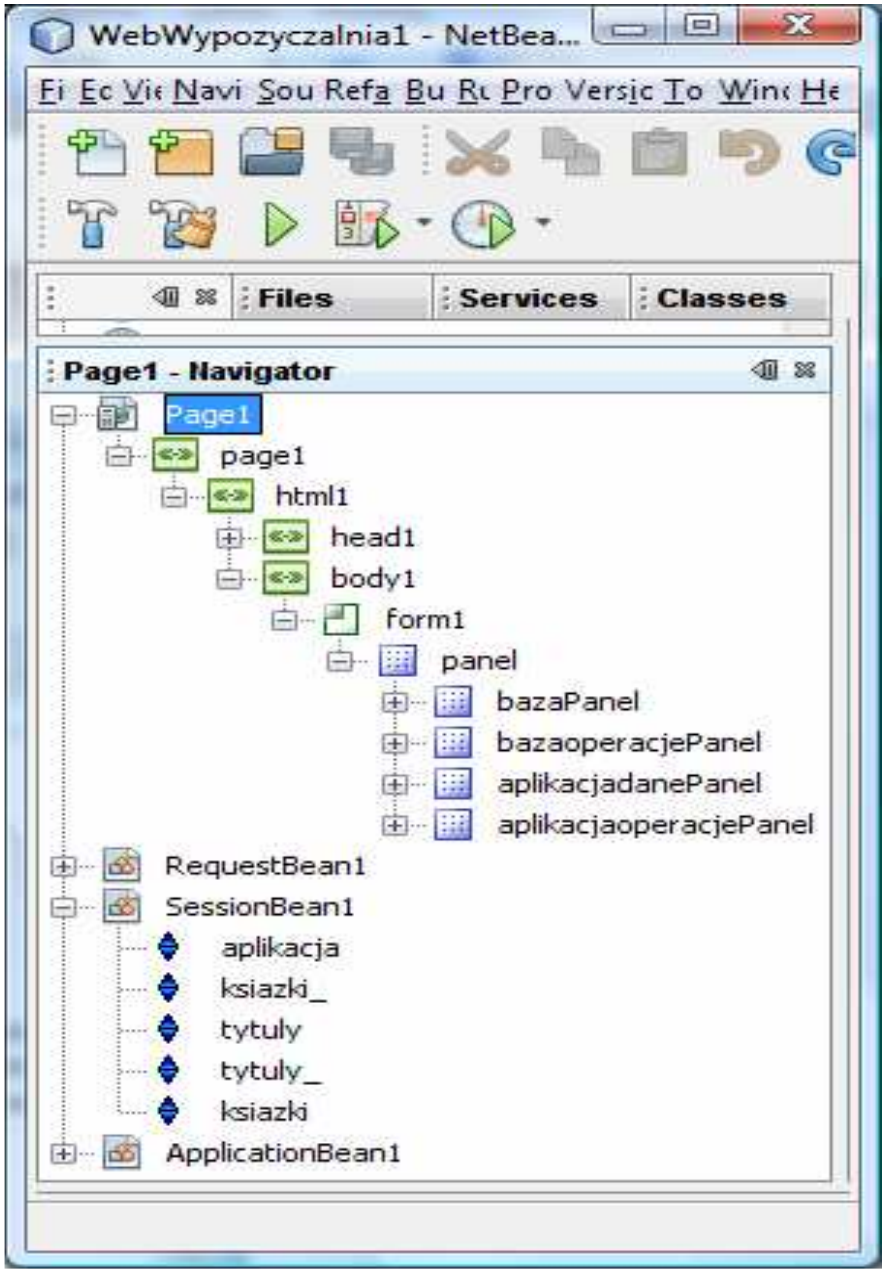
Tworzenie warstwy prezentacji dla przypadku użycia „dodaj egzemplarz” (2)

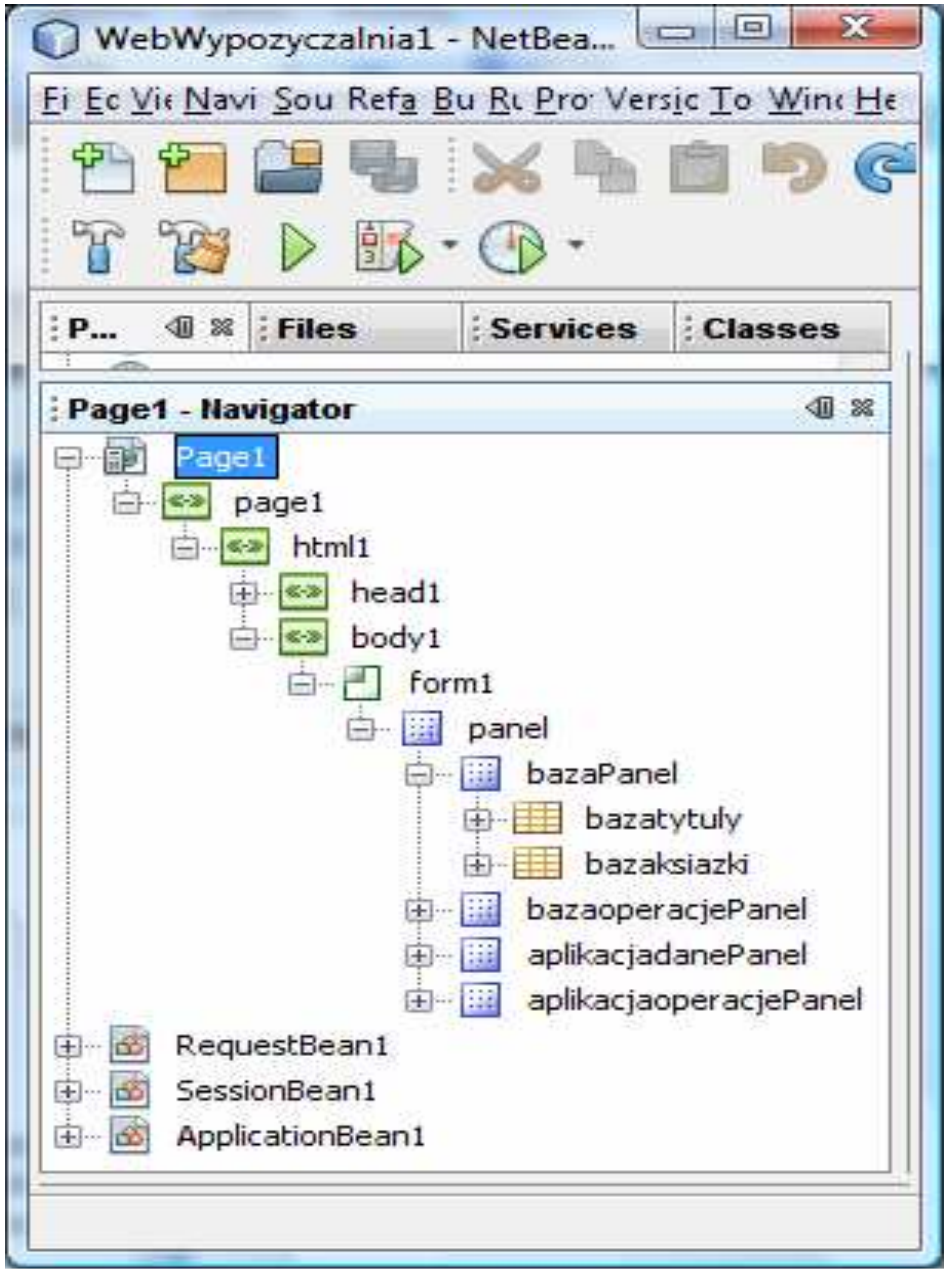
- Slajd (12) - wstawianie kodu dla przypadku użycia **„dodaj egzemplarz**” w warstwie prezentacji. Metoda **dodajksiazke_action()** jako obsługa zdarzenia klikania na przycisk **dodajksiazke** – wybór tytułu następuje w elemencie **tytuly** typu **DropDown List** przez wybór pozycji z listy (**tytuly.getSelected**), natomiast dane o numerze i ewentualnie terminie podawane są z pól tekstowych **numer** i **termin (12).** Należy uzupełnić kod metody **prerender** wg slajdu (13). Slajdy (14) i (15) pokazują uruchomioną aplikację.
- Napisać kod w klasie w **SessionBean1** wg slajdu (16). Kod metody **updateAplikacja()** służy do zapisu danych o tytułach i książkach w warstwie biznesowej aplikacji pobierając dane z bazy danych za pomocą tablic **TTytul_ksiazki tytuly[ ]** oraz **Egzemplarz ksiazki [].** Slajd (17) pokazuje diagram sekwencji metody aktualizującej zawartość warstwy biznesowej.
- Zgodnie ze slajdem (18) napisać w klasie **SessionBean1** metodę **zapisz_ksiazki_do_bazy()** zapisujący dane **zapisz_ksiazki_do_bazy()** aplikacji typu **TEgzemplarz** do bazy danych metodą **addTEgzemplarzs()** klasy **TEgzemplarzController**. Metoda jest wykorzystana do obsługi zdarzenia klikania w przyciski **dodajksiazkabaza (metoda dodajksiazkabaza_action)** w klasie **Page1**. Na slajdzie (20) pokazano diagram sekwencji tej operacji. Slajd (19) przedstawia metodę **findTEgzemplarzs** w klasie **TEgzemplarzController** do sprawdzenia, czy nie ma próby wstawienia ponownie tej samej ksiazki do tabeli TEgzemplarz naruszając więzy integralności referencyjnej.
- Na slajdzie (21) pokazano działającą aplikację, która również zapisuje dane o egzemplarzach w aplikacji
- Na slajdach 22-24 pokazano działające przypadki użycia.
- Na slajdzie (25) pokazano zawartość danych bezpośrednio w tabeli **TTytul_ksiazki** w bazie danych (zakładka **Service** i opcja **View Data** dla tabeli)

Przygotowanie komponentów dla przypadku użycia **„dodaj egzemplarz”(1)**

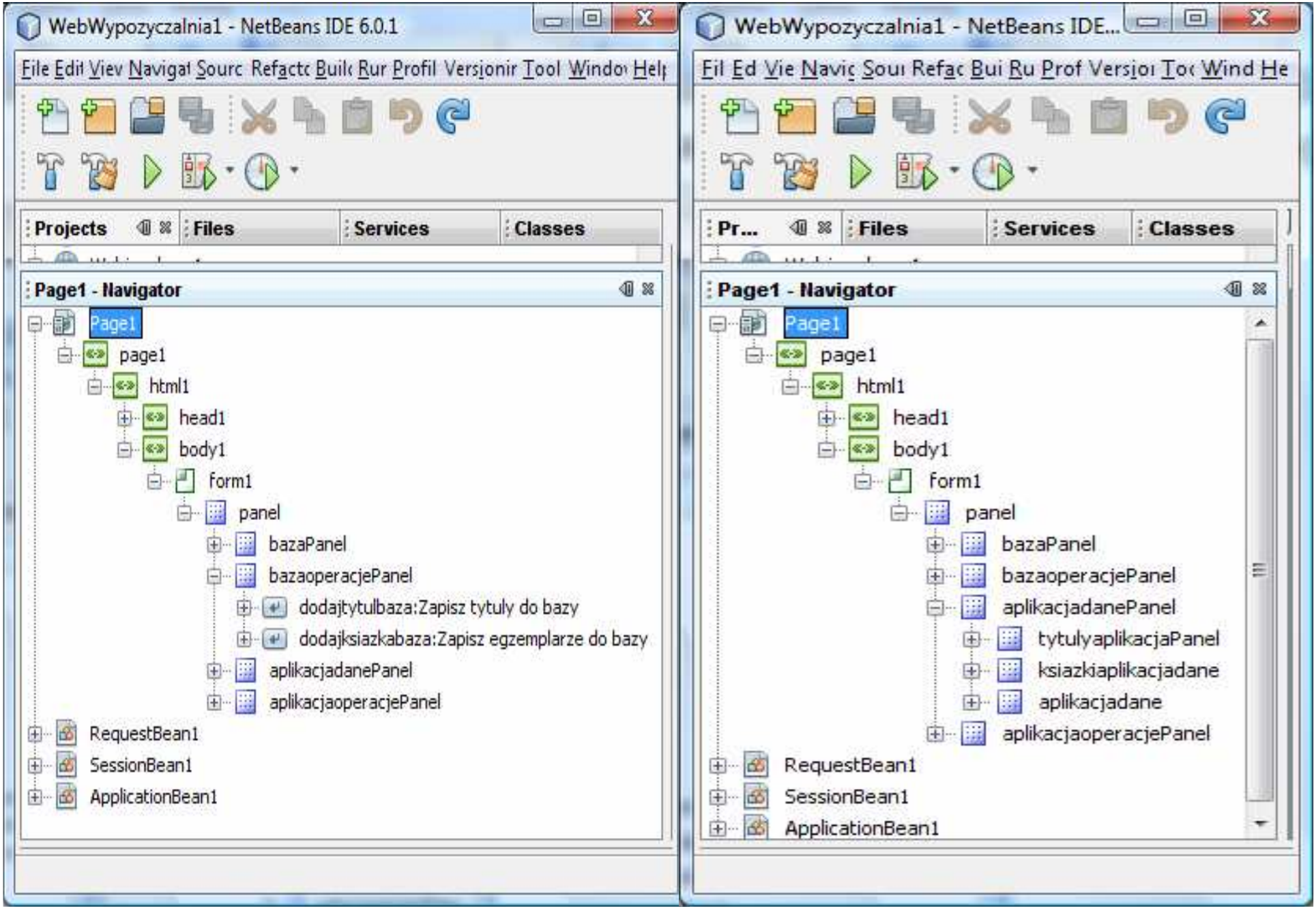


Zawartość formularza strony Page1.jsp – przeciągniąć do panela bazaPanel komponent Table – nadać mu id=bazaksiazki (2)

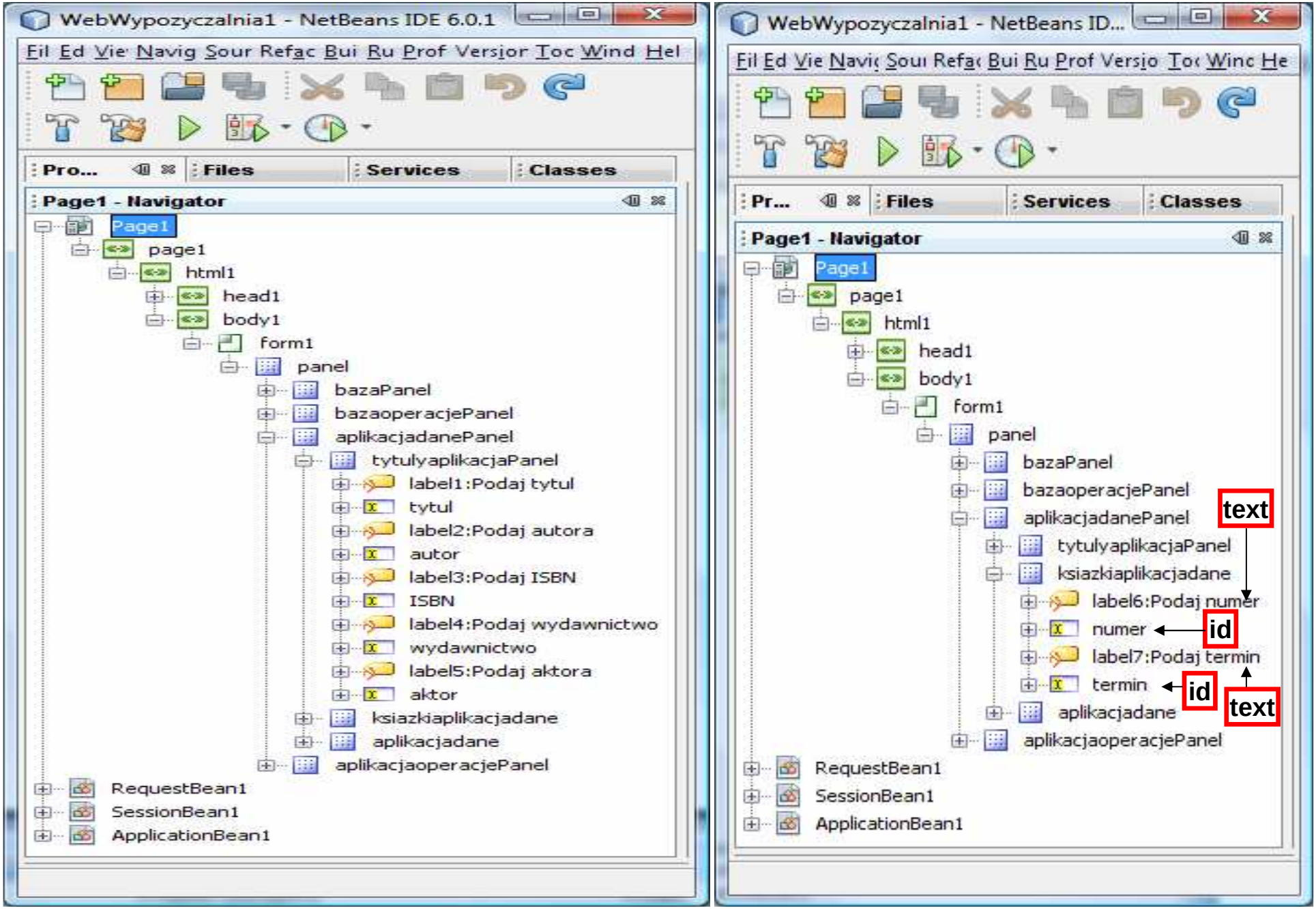




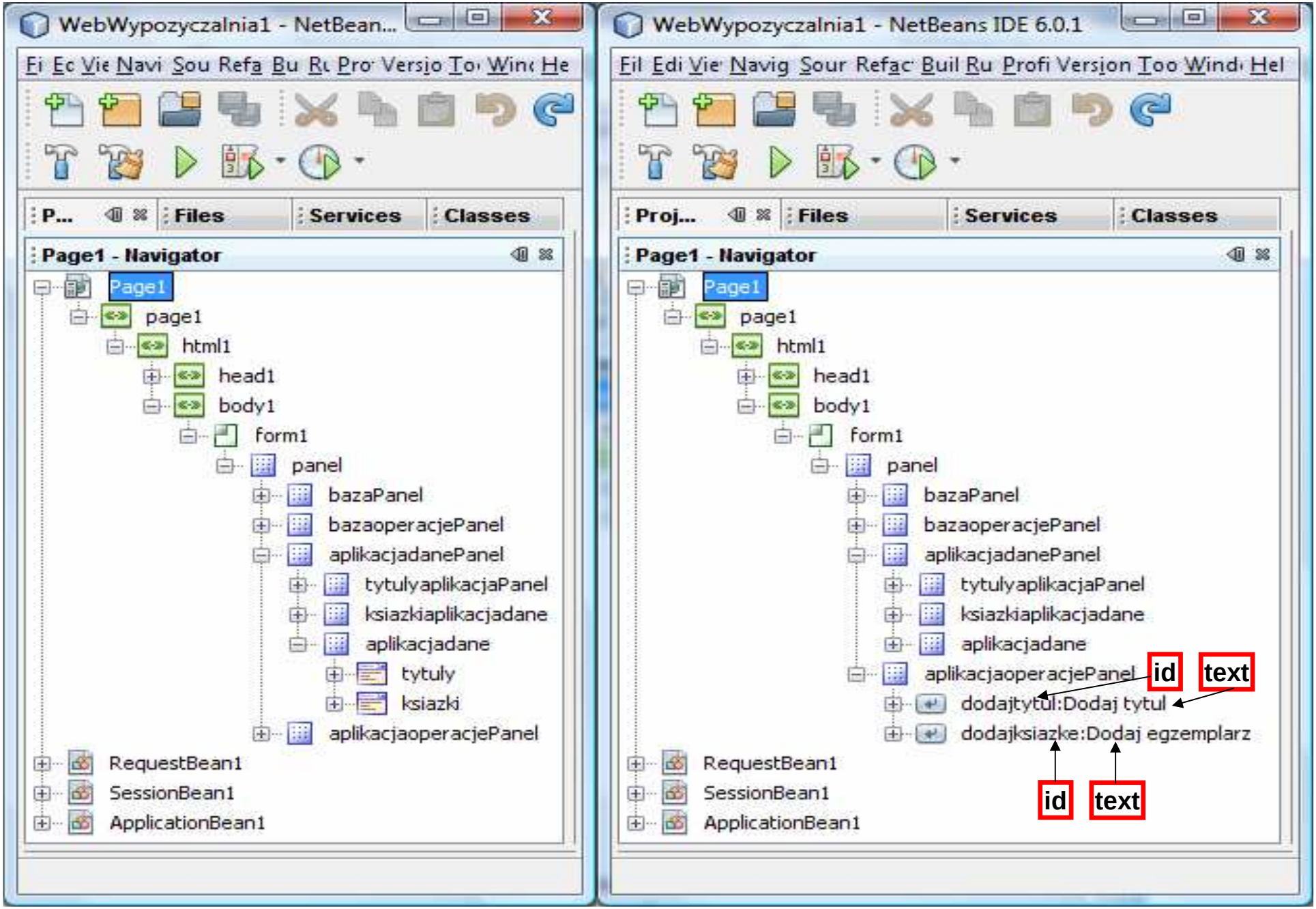
Zawartość formularza strony Page1.jsp (3)



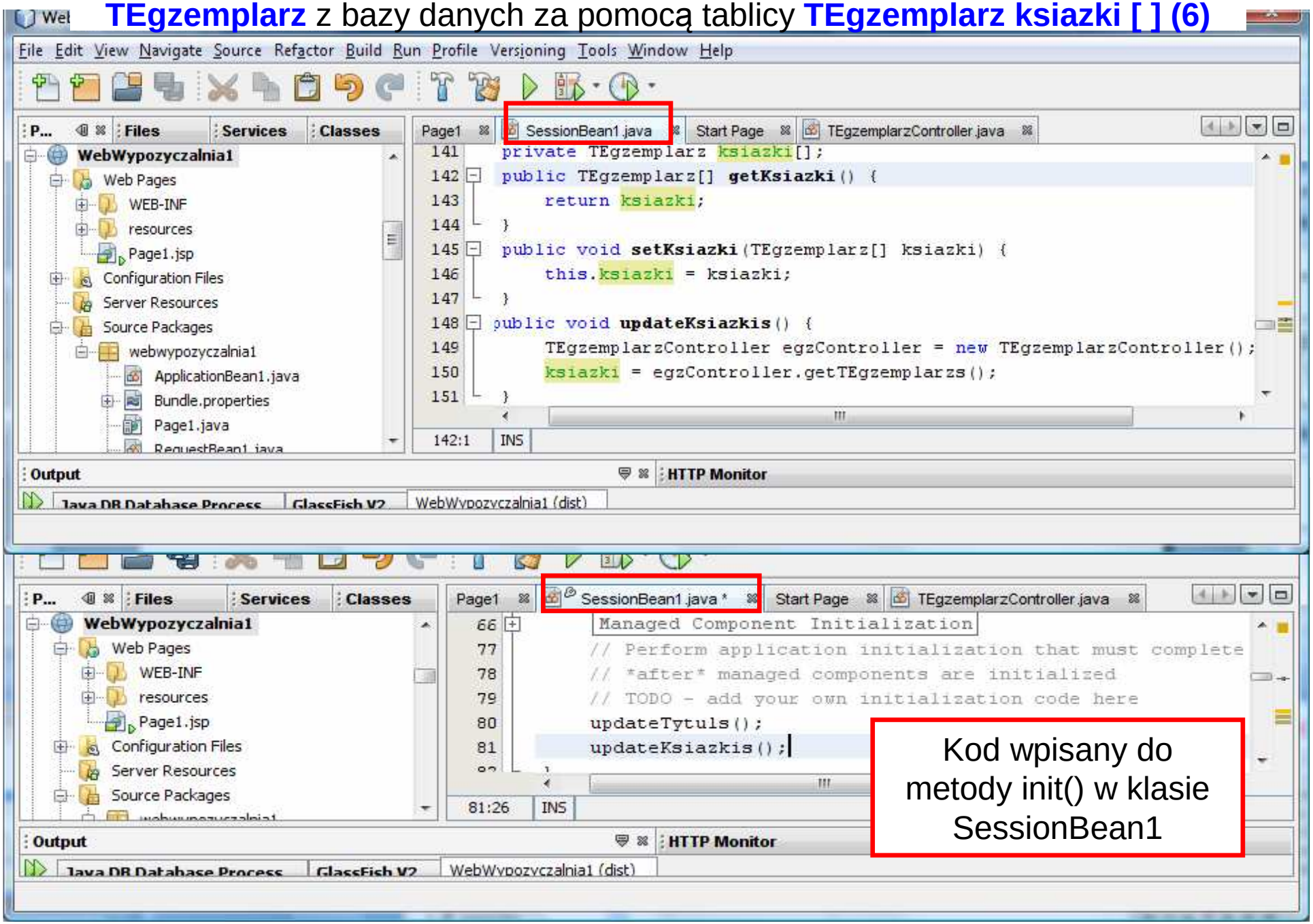
Zawartość formularza strony Page1.jsp (4)



Zawartość formularza strony Page1.jsp (5)

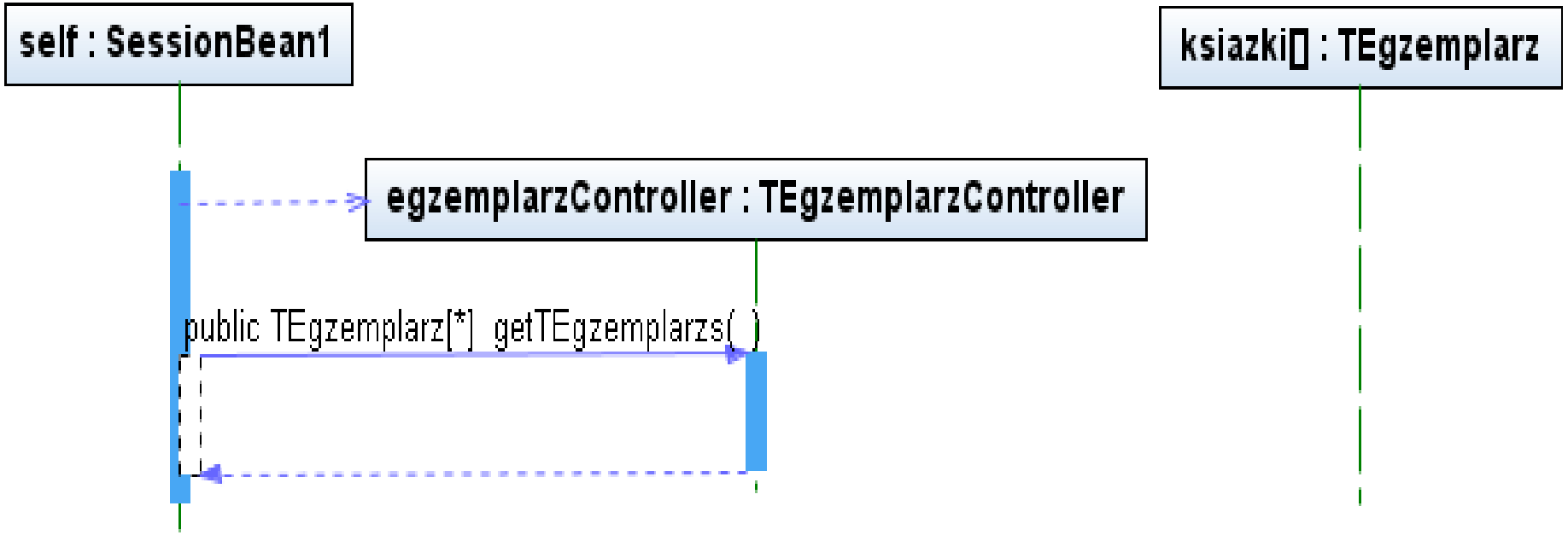


Tworzenie kodu do „bindowania” komponentu **Table** z danymi z tabeli **TEgzemplarz** z bazy danych za pomocą tablicy **TEgzemplarz ksiazki [ ] (6)**

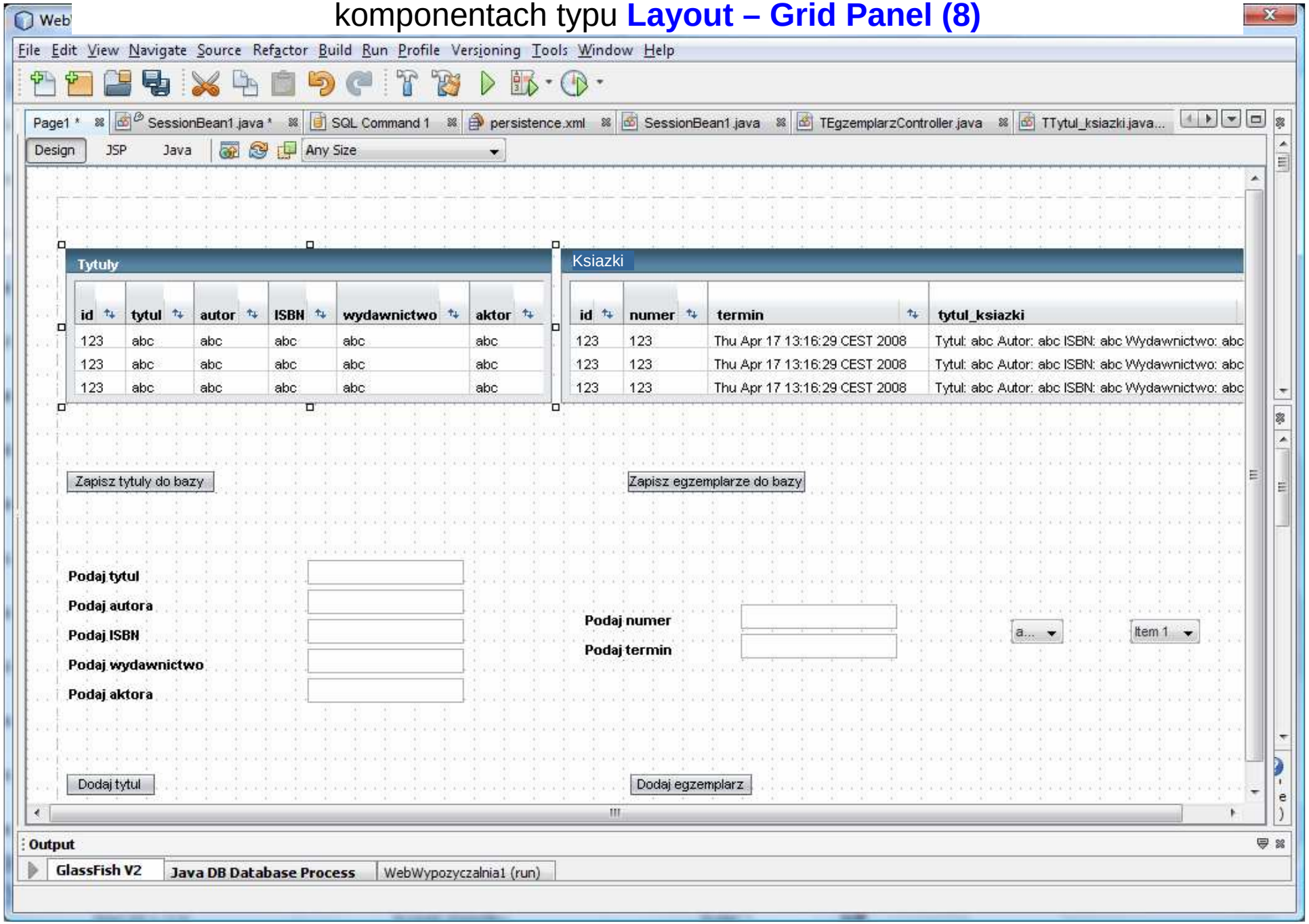


Odczyt książek z bazy danych (7)

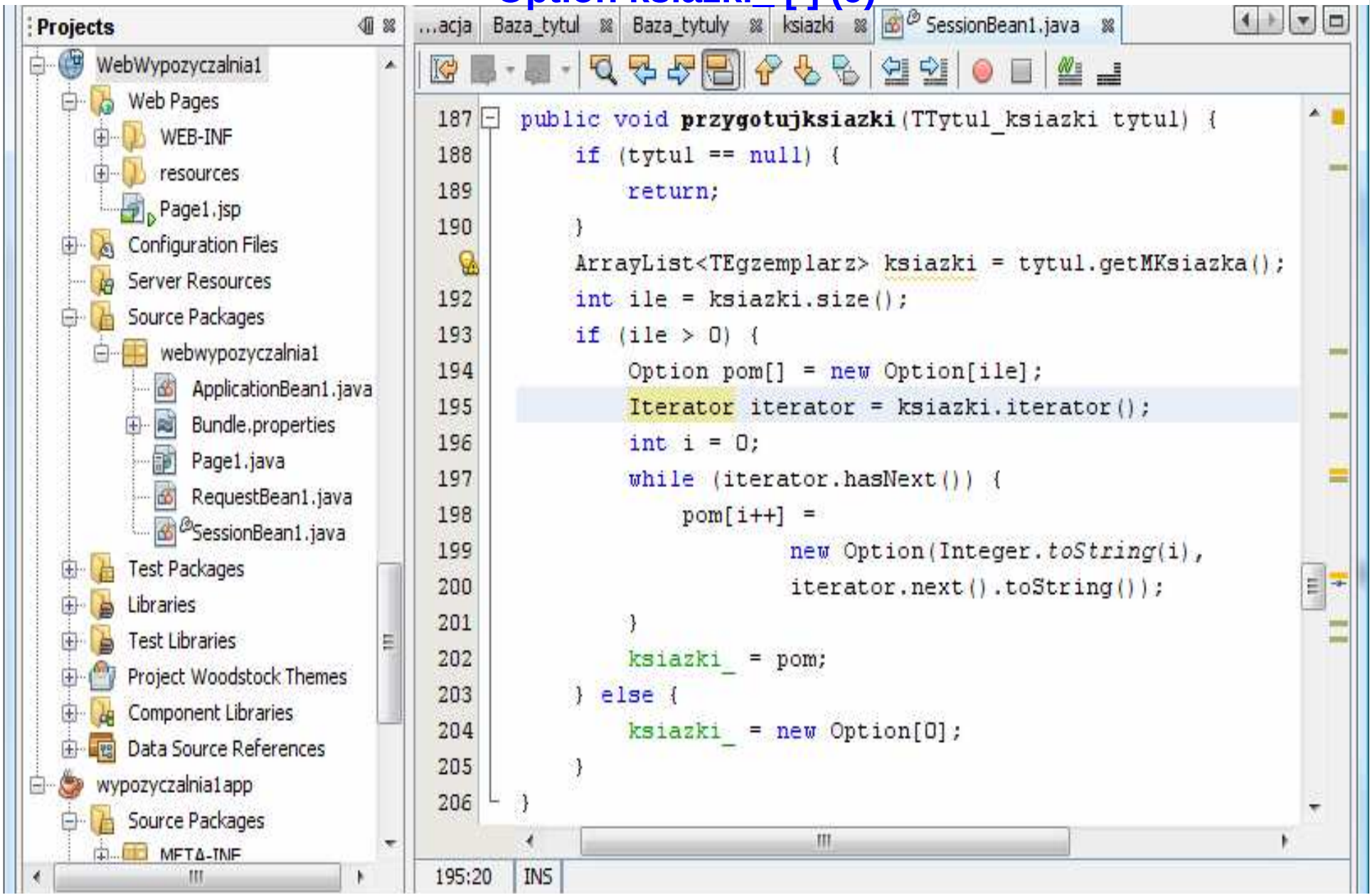
public void SessionBean1::updateKsiazkis()



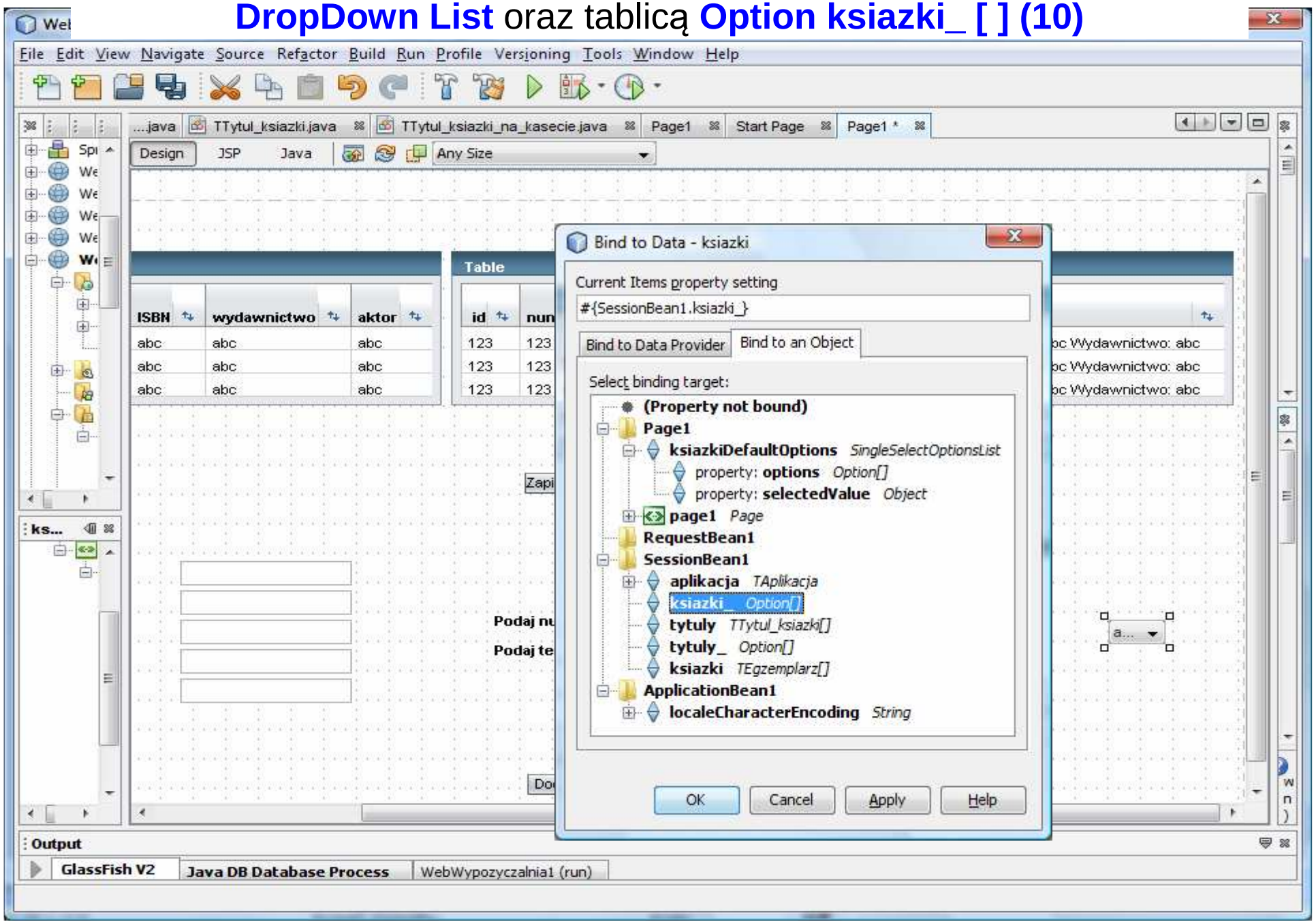
Wstawianie nowych komponentów rozpinanych na zagnieżdżanych komponentach typu **Layout – Grid Panel (8)**



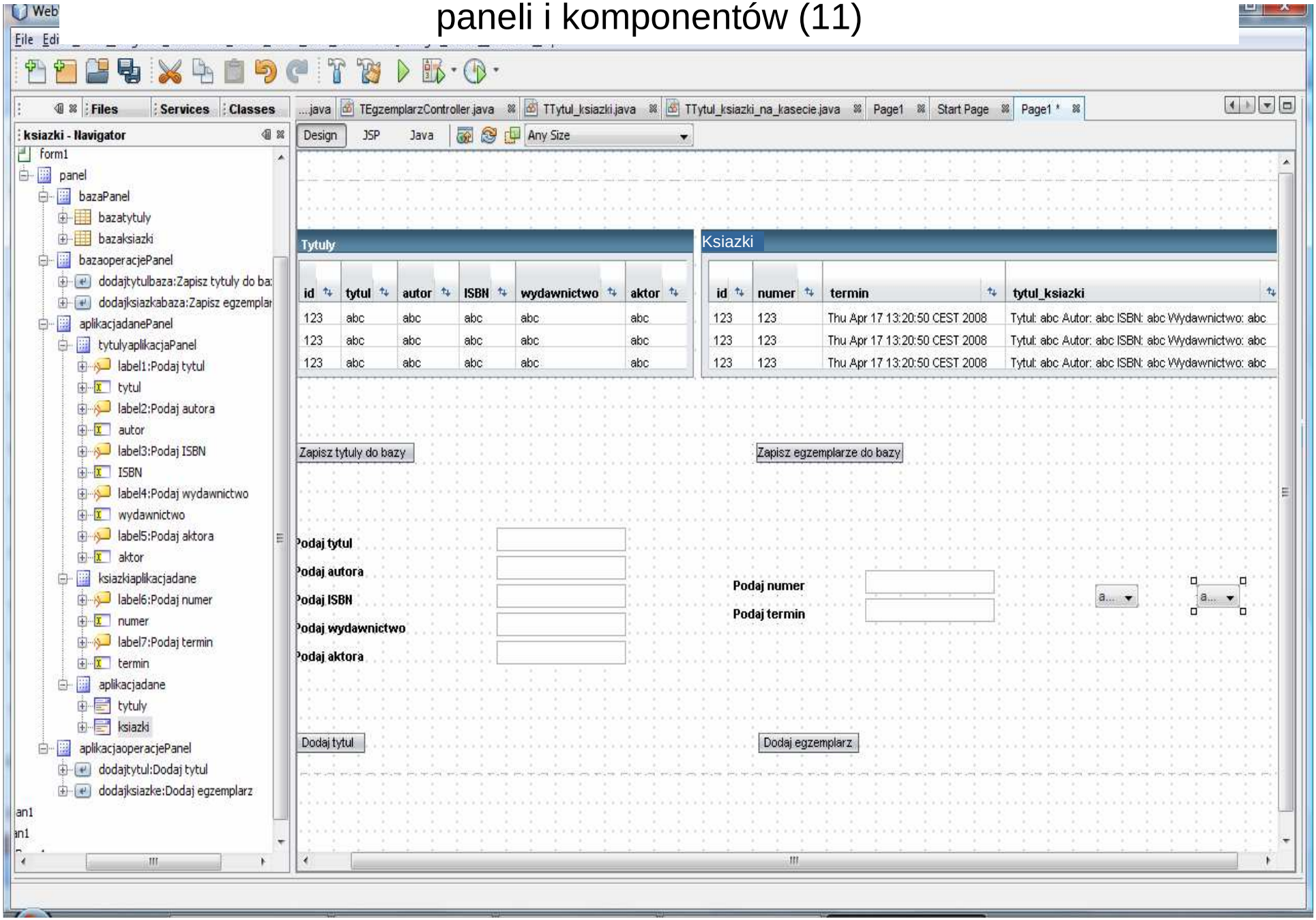
Wstawienie kodu do wyświetlania danych o egzemplarzach przechowywanych w aplikacji w elemencie **ksiazki** typu **DropDown List** za pośrednictwem tablicy **Option ksiazki_ [ ] (9)**



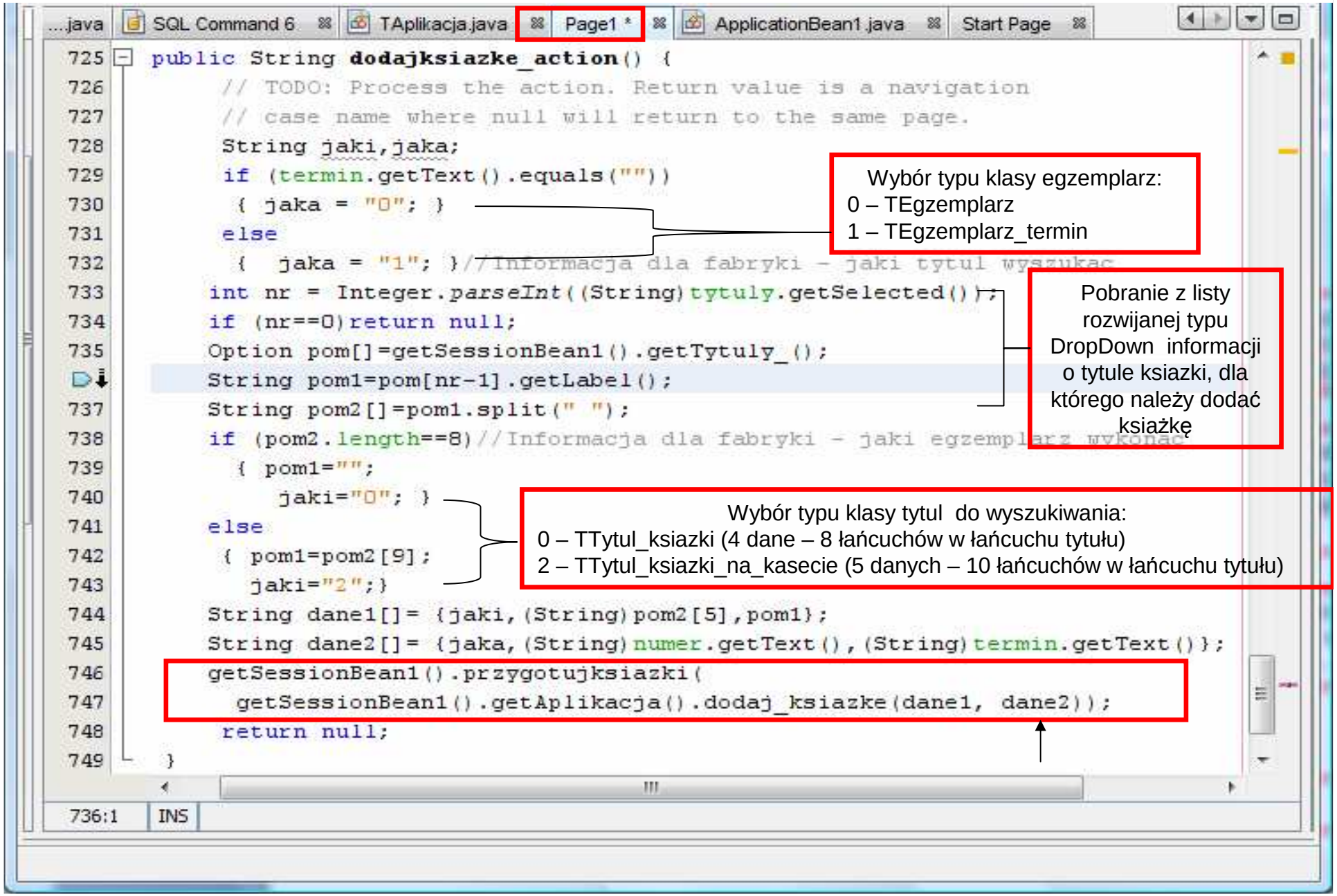
Wykonanie operacji **Bind to Data** między elementem **ksiazki** typu **DropDown List** oraz tablicą **Option ksiazki_ [ ] (10)**



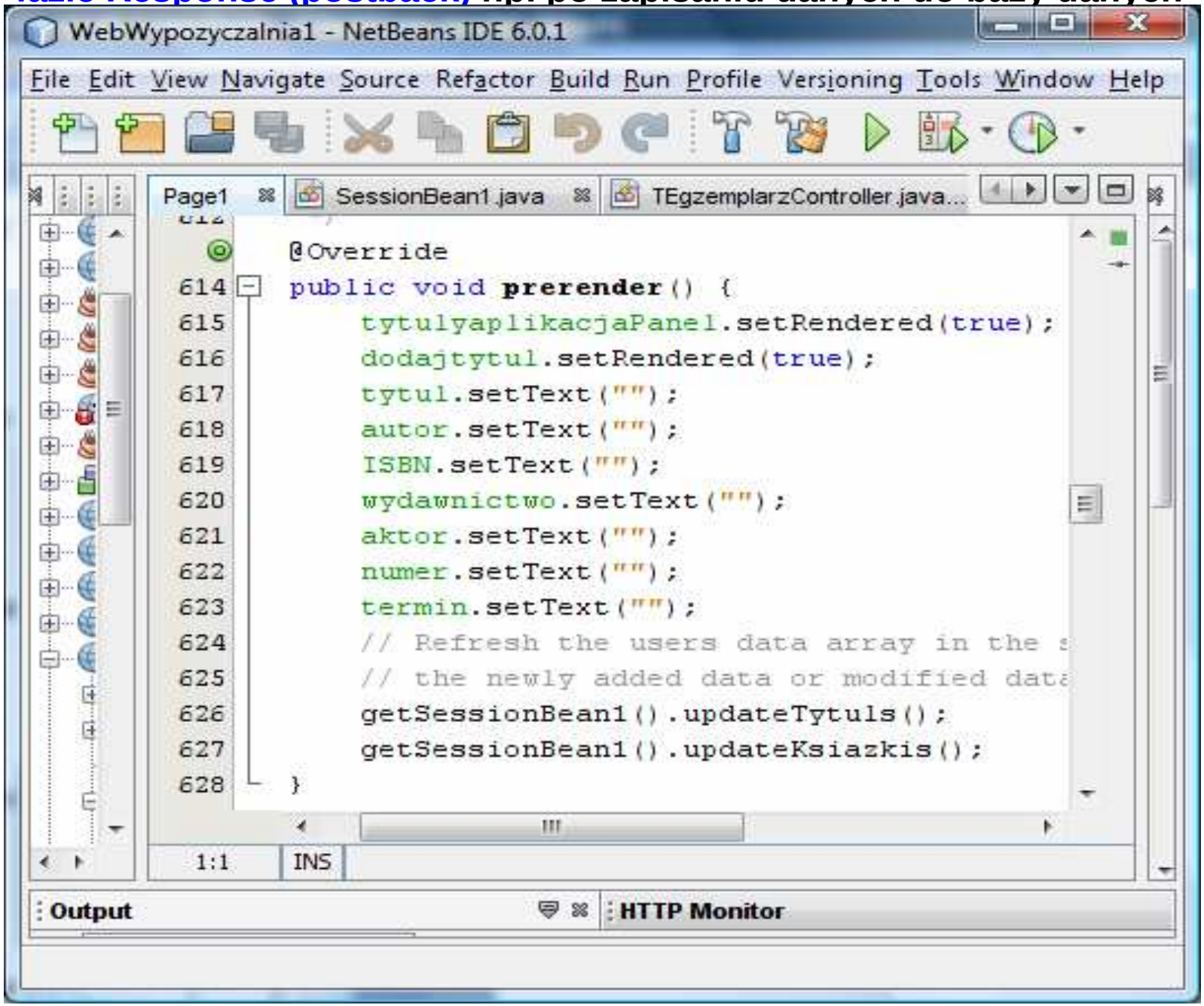
Efekt operacji **Bind to Data** – okienko **Navigator** przedstawia strukturę paneli i komponentów (11)



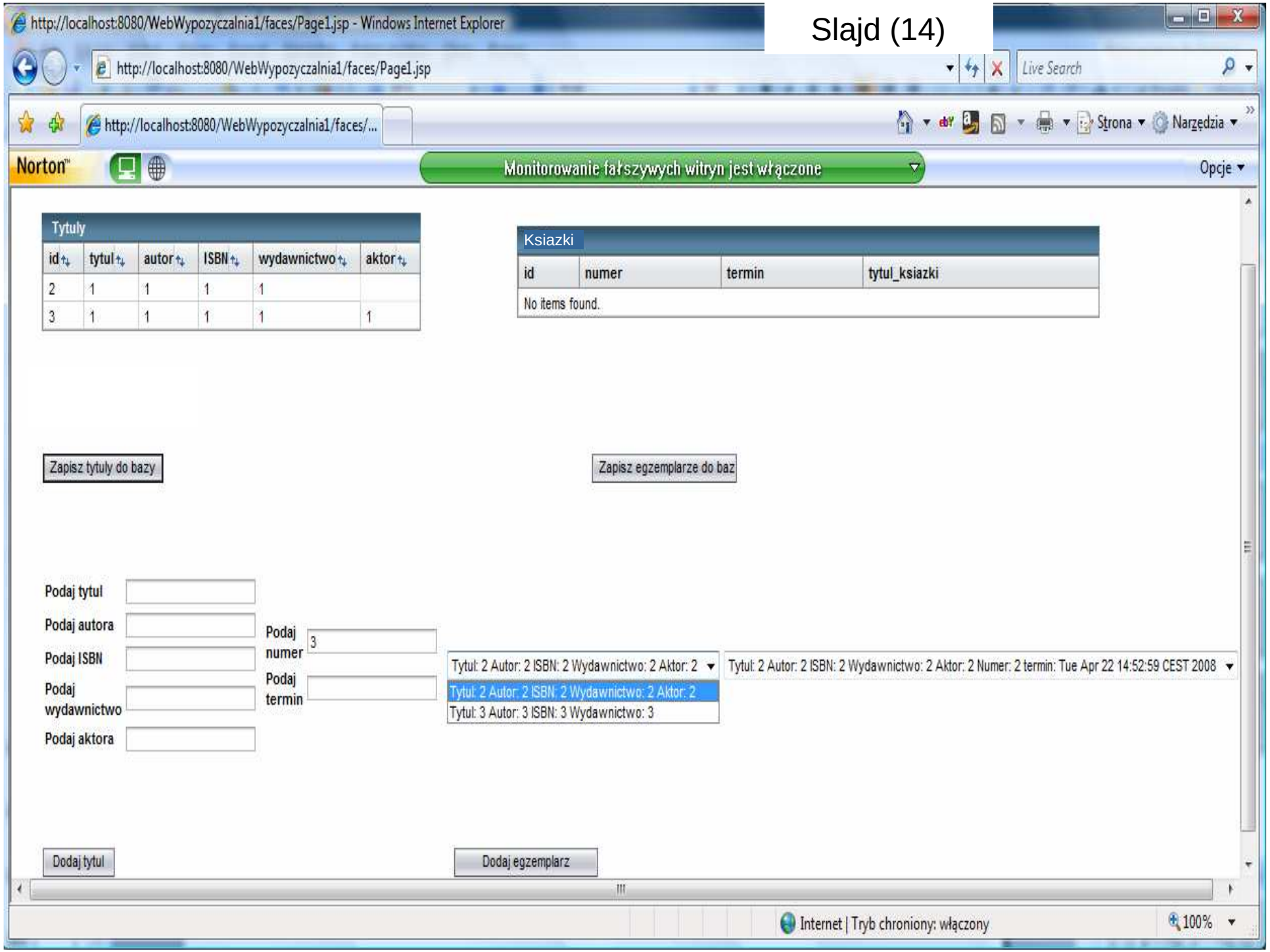
Wstawianie kodu dla przypadku użycia **„dodaj egzemplarz”** w warstwie prezentacji – wybór tytułu następuje w elemencie **tytuly** typu **DropDown List** przez wybór pozycji z listy (tytuly.getSelected), natomiast dane o numerze i ewentualnie terminie podawane są z pól tekstowych **numer** i **termin (12)**



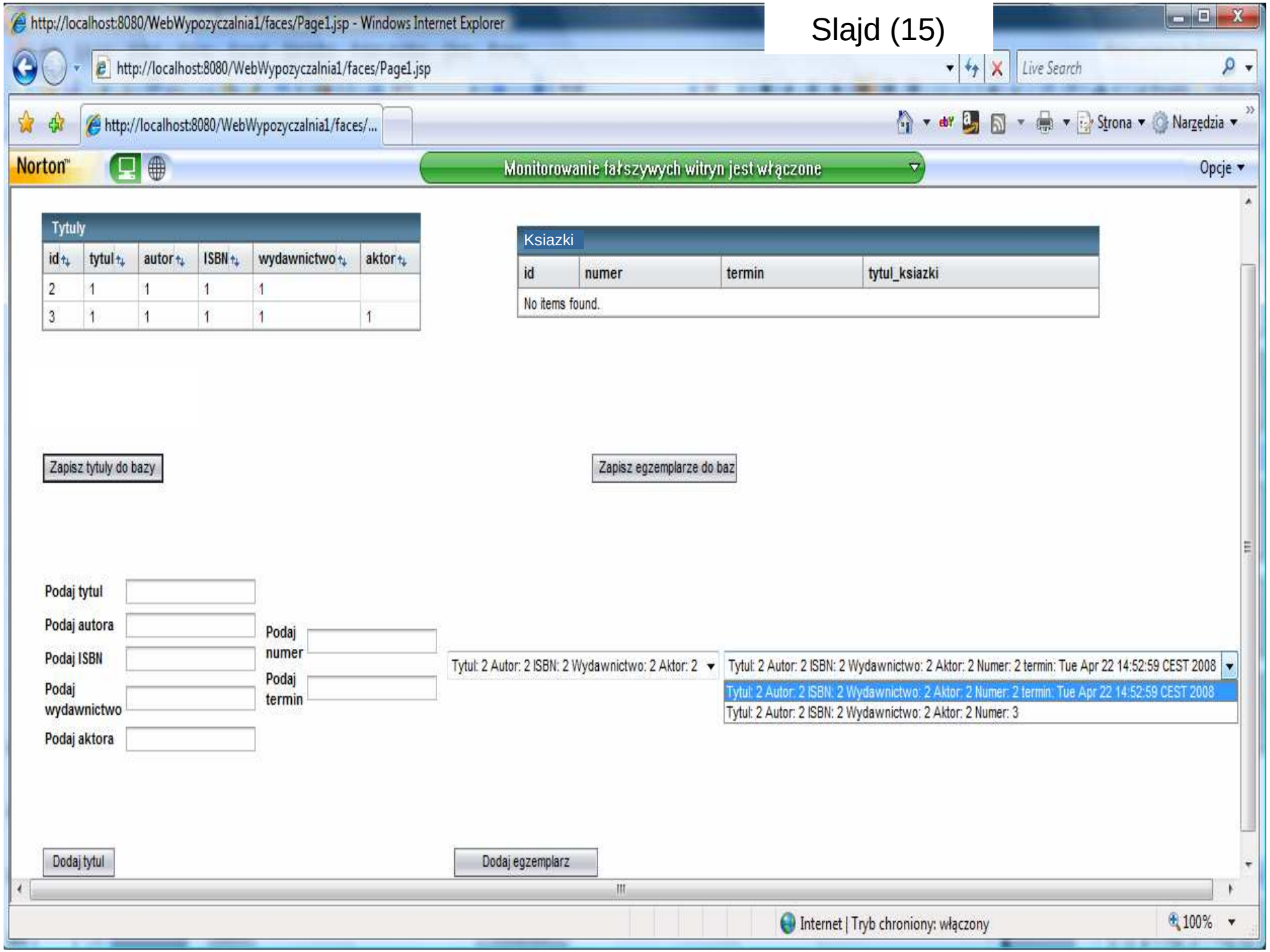
Uzupełnienie kodu metody **prerender** w klasie **Page1.java (13) – jest wywoływana w fazie Response (postback) np. po zapisaniu danych do bazy danych**



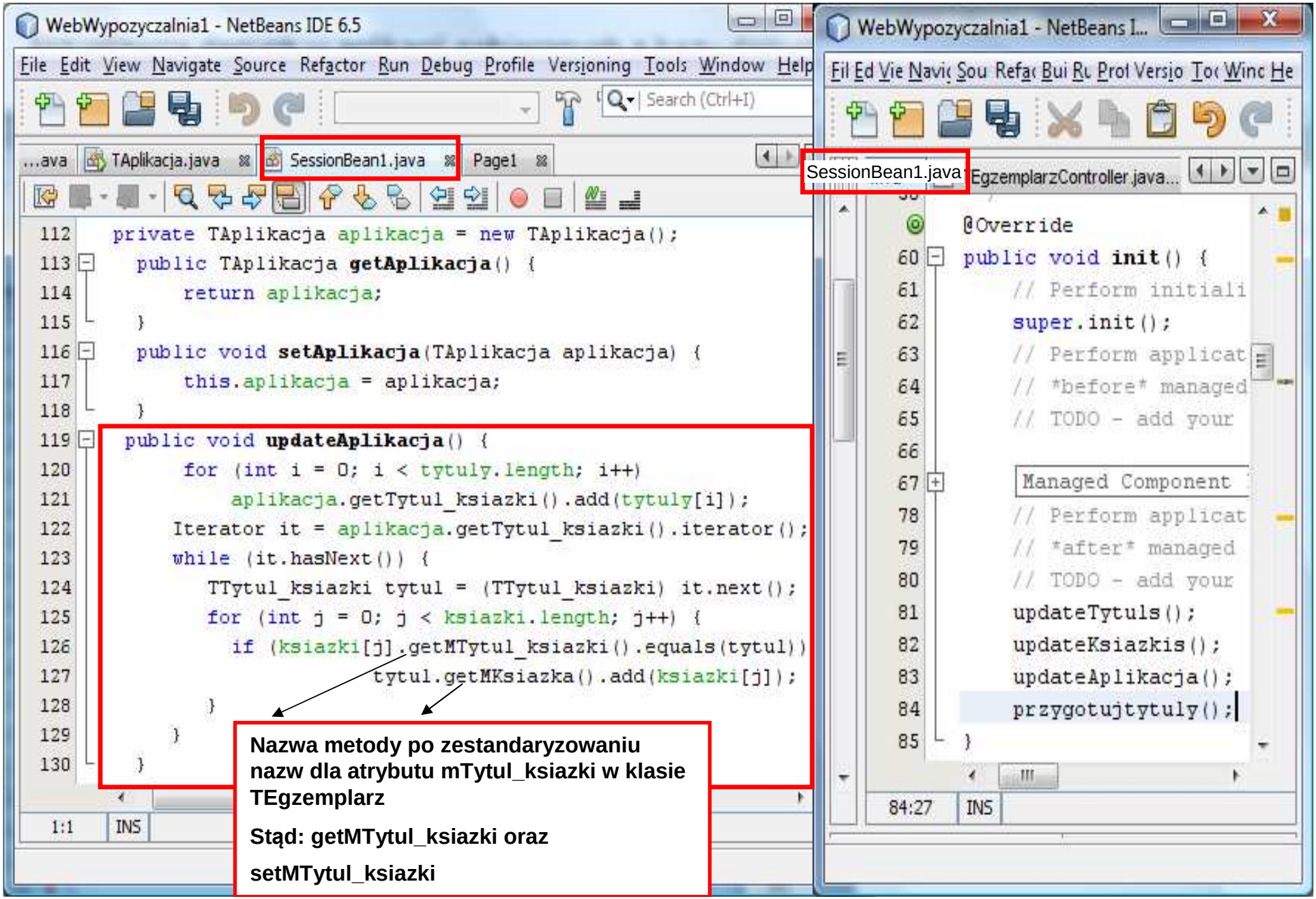
Slajd (14)


http://localhost:8080/WebWypozyczalnia1/faces/Page1.jsp - Windows Internet Explorer
http://localhost:8080/WebWypozyczalnia1/faces/Page1.jsp
Live Search
http://localhost:8080/WebWypozyczalnia1/faces/...
Strona
Narzędzia
Norton
Monitorowanie fałszywych witryn jest włączone
Opcje
Tytuly
id
tytul
autor
ISBN
wydawnictwo
aktor
2 1 1 1 1
3 1 1 1 1 1
Ksiazki
id
numer
termin
tytul_ksiazki
No items found.
Zapisz tytuly do bazy
Zapisz egzemplarze do baz
Podaj tytul
Podaj autora
Podaj ISBN
Podaj wydawnictwo
Podaj aktora
Podaj numer
3
Podaj termin
Tytul: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2
Tytul: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2
Tytul: 3 Autor: 3 ISBN: 3 Wydawnictwo: 3
Tytul: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2 Numer: 2 termin: Tue Apr 22 14:52:59 CEST 2008
Dodaj tytul
Dodaj egzemplarz
Internet | Tryb chroniony: włączony
100%

Slajd (15)

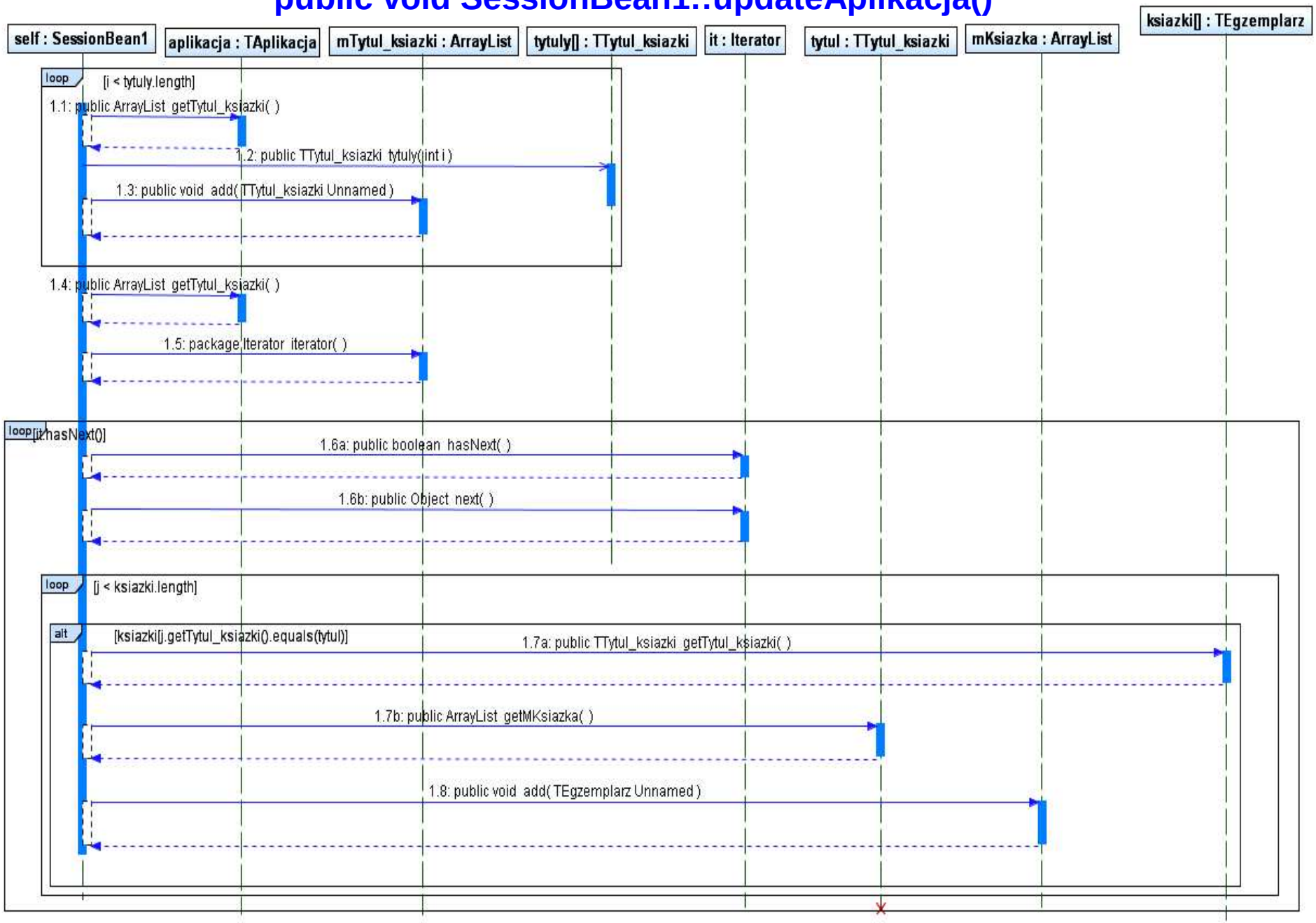

http://localhost:8080/WebWypozyczalnia1/faces/Page1.jsp - Windows Internet Explorer
http://localhost:8080/WebWypozyczalnia1/faces/Page1.jsp
Live Search
http://localhost:8080/WebWypozyczalnia1/faces/...
Strona
Narzędzia
Norton
Monitorowanie fałszywych witryn jest włączone
Opcje
Tytuly
id
tytul
autor
ISBN
wydawnictwo
aktor
2 1 1 1 1
3 1 1 1 1 1
Ksiazki
id
numer
termin
tytul_ksiazki
No items found.
Zapisz tytuly do bazy
Zapisz egzemplarze do baz
Podaj tytul
Podaj autora
Podaj ISBN
Podaj wydawnictwo
Podaj aktora
Podaj numer
Podaj termin
Tytul: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2
Tytul: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2 Numer: 2 termin: Tue Apr 22 14:52:59 CEST 2008
Tytul: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2 Numer: 2 termin: Tue Apr 22 14:52:59 CEST 2008
Tytul: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2 Numer: 3
Dodaj tytul
Dodaj egzemplarz
Internet | Tryb chroniony: włączony
100%

Aktualizacja danych w aplikacji pobieranych z bazy danych za pomocą tablic **TTytul_ksiazki tytuly[ ]** oraz **TEgzemplarz ksiazki [ ]** (16)

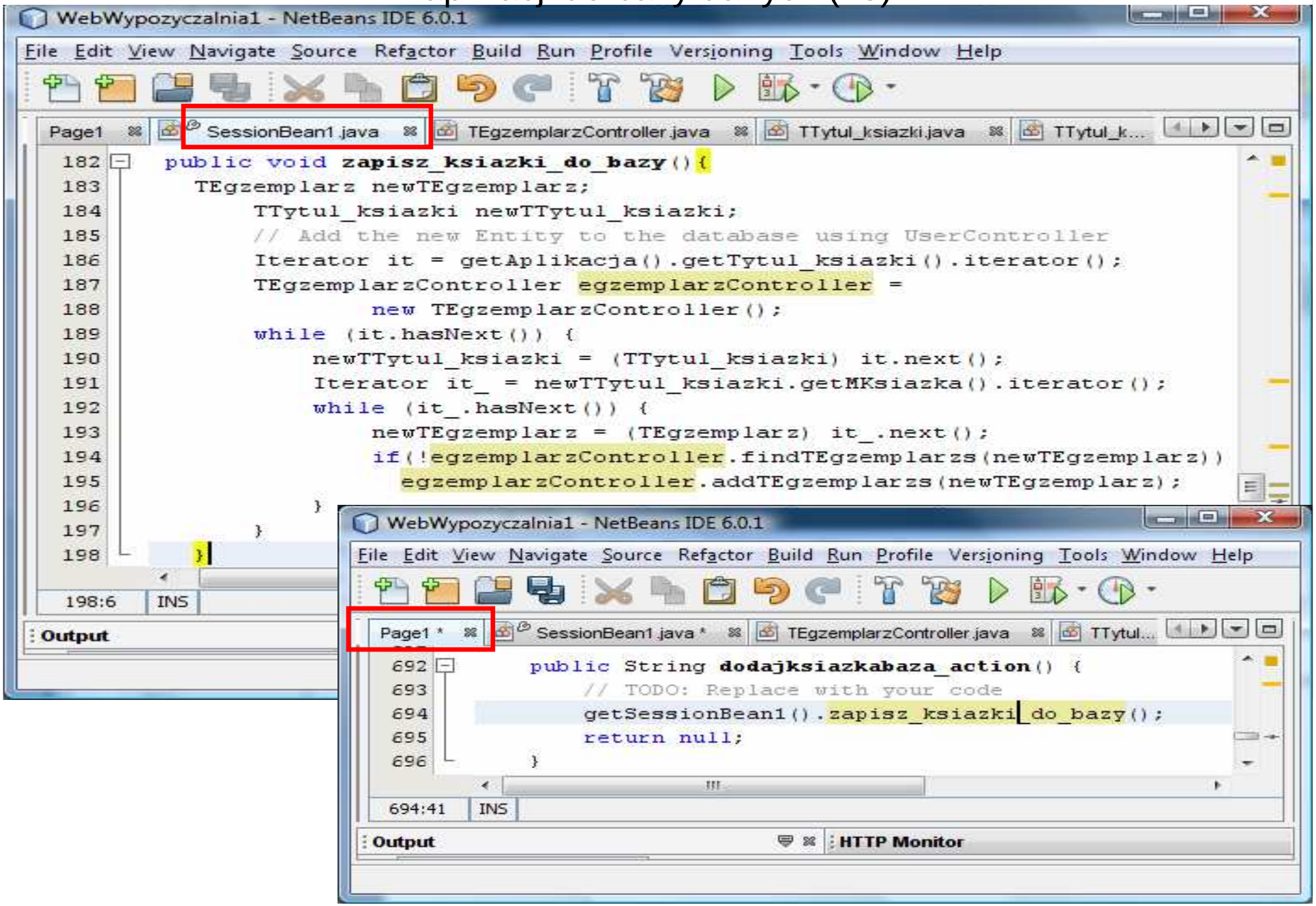


Odczyt danych z bazy tytułów oraz książek za pomocą tablic encji tytuly i ksiazki (17)

public void SessionBean1::updateAplikacja()



Wykonanie kodu zapisującego dane o egzemplarzach przechowywanych w aplikacji do bazy danych (18)



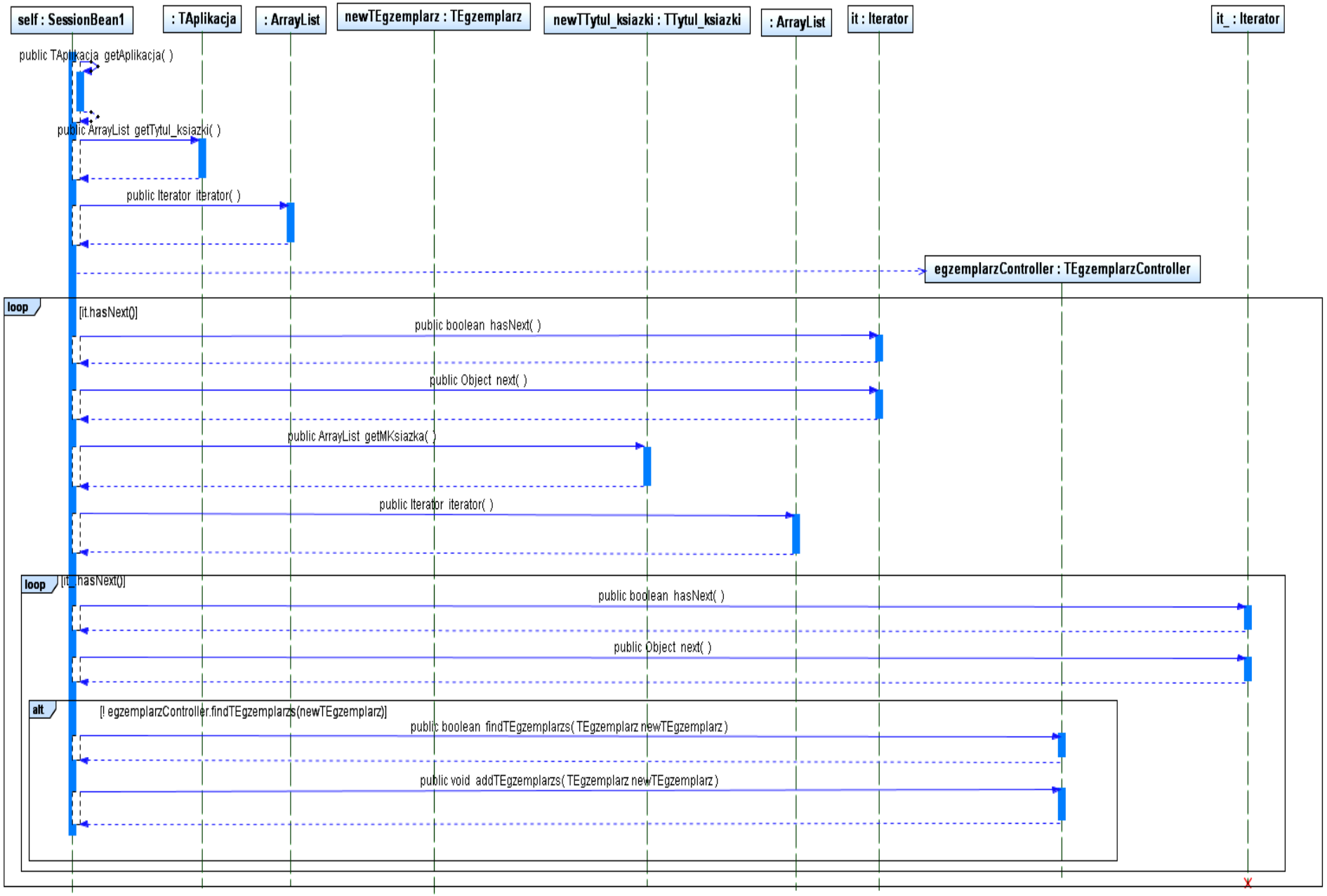
Nowa metoda w klasie TEgzemplarzController do sprawdzania, czy obiekt TEgzemplarz w relacji wiele do jeden zostanie zapisany ponownie naruszając więzy integralności referencyjnej tabeli TEgzemplarz (19)

```
public boolean findTEgzemplarzs(TEgzemplarz TEgzemplarz)
{
  EntityManager em = getEntityManager();
  try {
   return em.find(TEgzemplarz.class, TEgzemplarz.getId())!=null; }
   finally {
     em.close();
     }
}
```

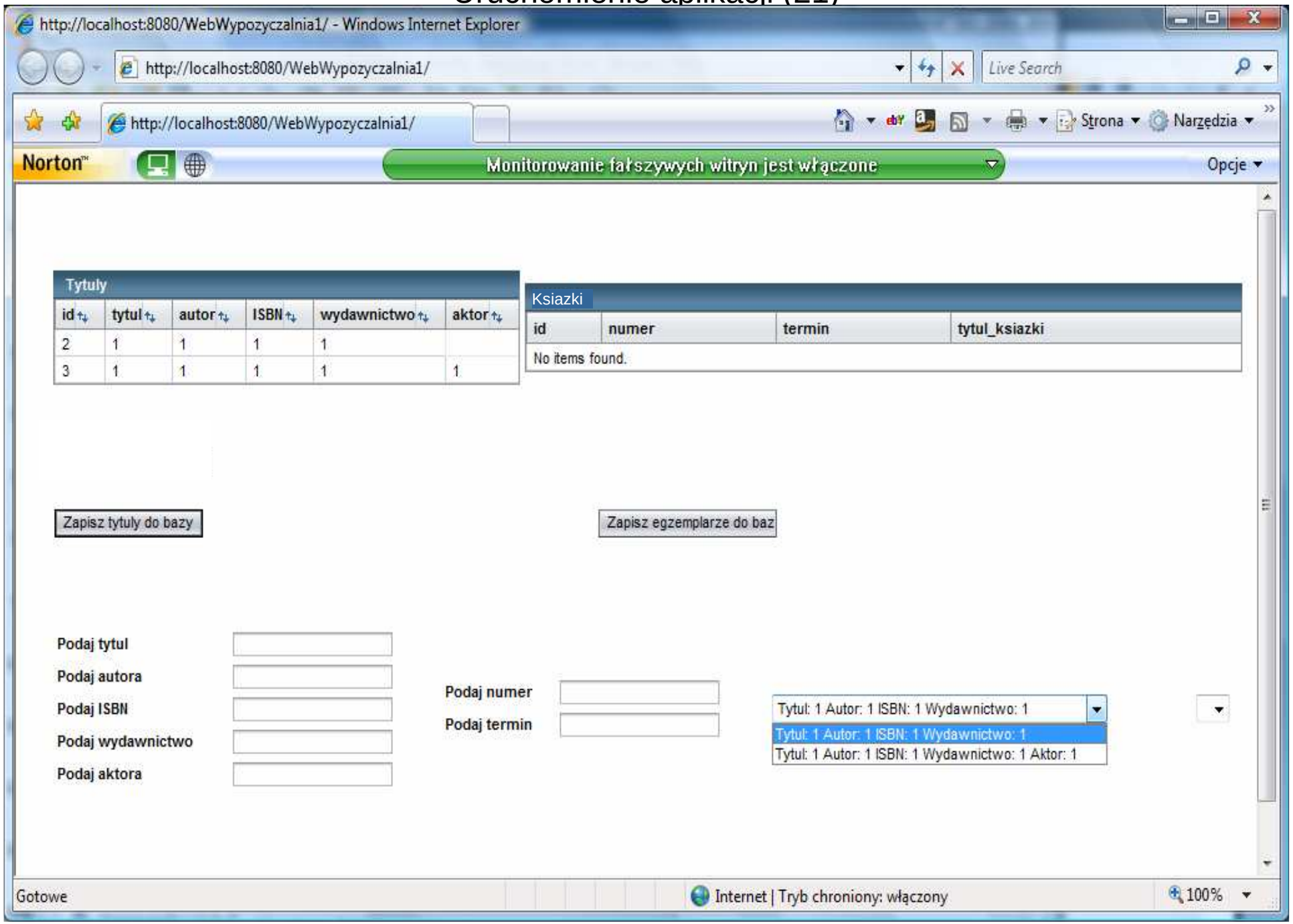
Uzupełniono kod konstruktora klasy TEgzemplarz – wstawia klucz główny równy 0 do danej, która jeszcze nie została zapisana do bazy danych. Wartość 0 klucza głównego jest wartością nie należąca do dziedziny wartości kluczy głównych- wtedy można przekazać taki obiekt do metody find w klasie EntityMaqnager

```
 public TEgzemplarz() {
       id = new Long(-1); // id = null; dla wersji 2 w lab.7_8
}
```

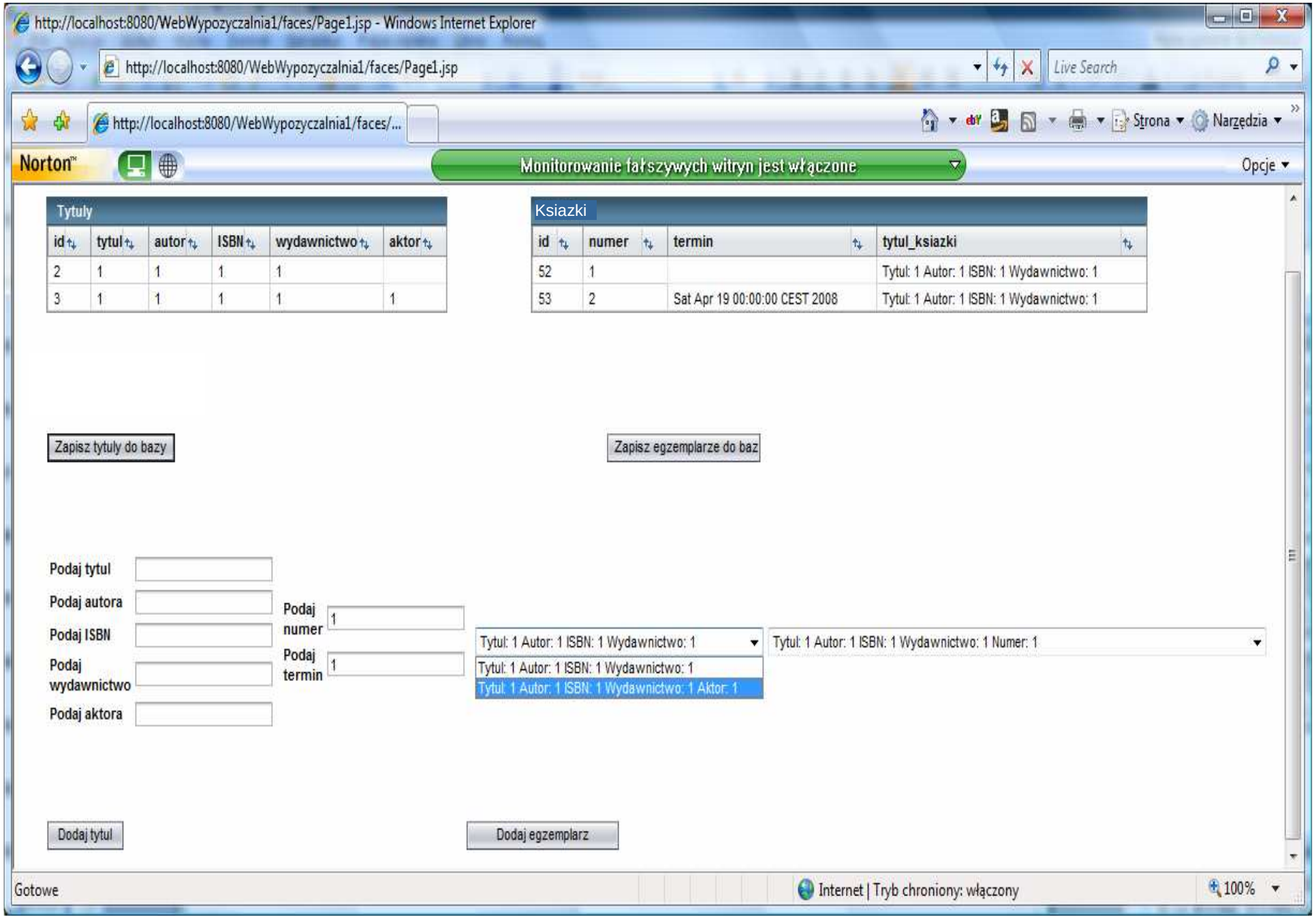
Zapis danych do bazy książek metodą add obiektu TEgzemplarzController (20)
public void SessionBean1::zapisz_ksiazki_do_bazy()



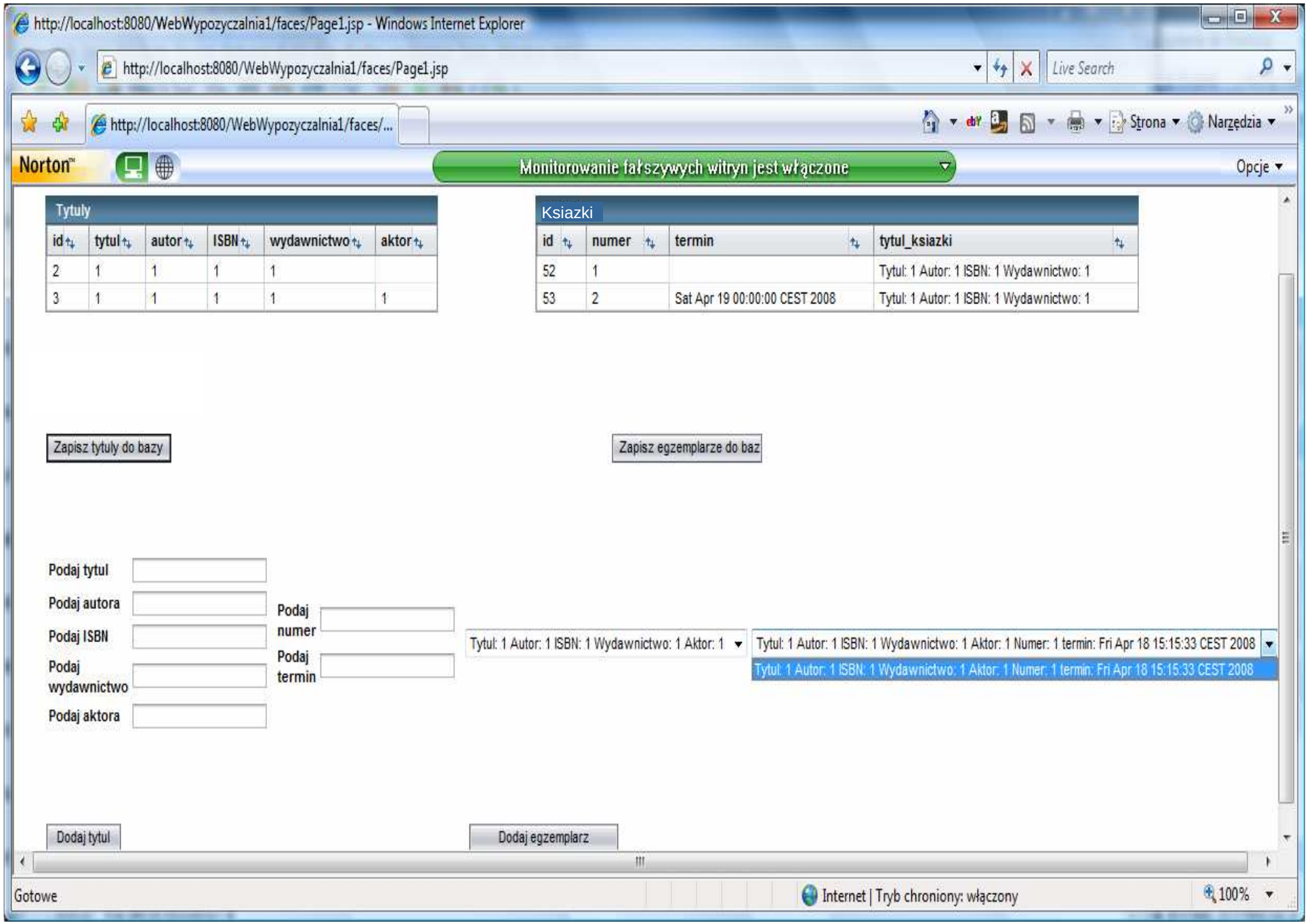
Uruchomienie aplikacji (21)



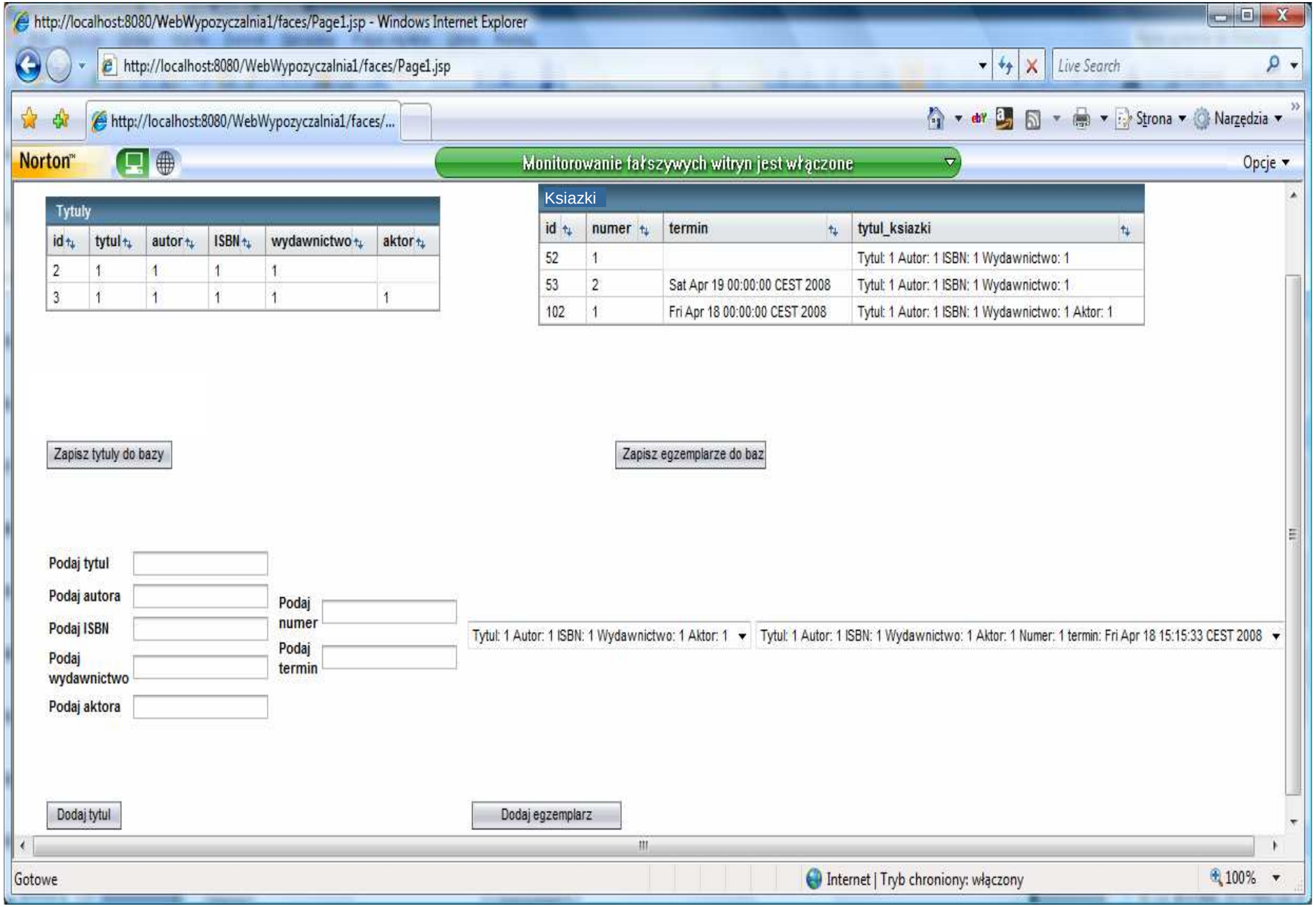
Wstawianie danych z aplikacji do tabeli **TEgzemplarz** w bazie danych (1) (22)



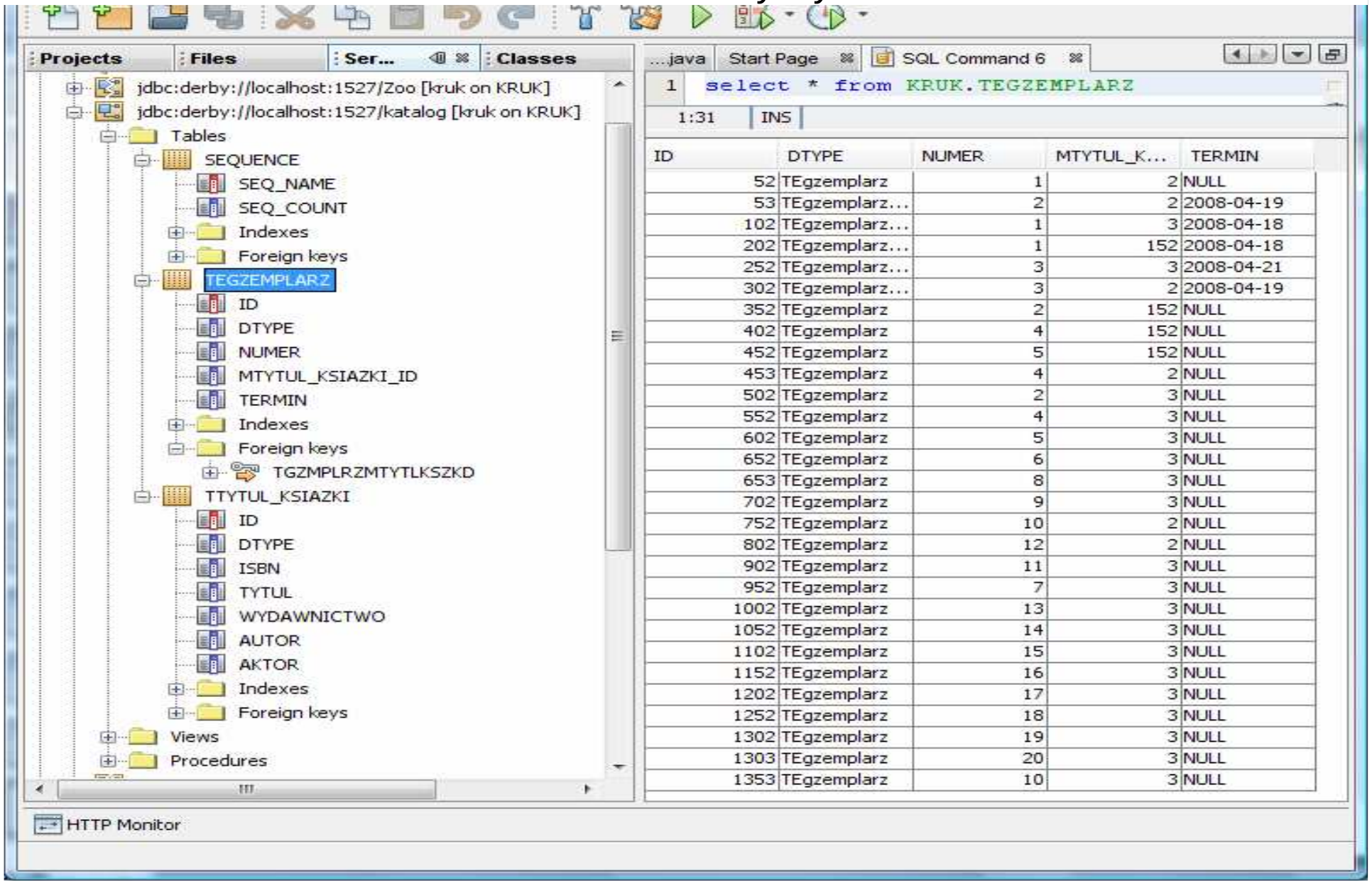
Wstawianie danych z aplikacji do tabeli **TEgzemplarz** w bazie danych (2) (23)



Wstawianie danych z aplikacji do tabeli **TEgzemplarz** w bazie danych (3) (24)



Widok bazy danych w zakładce **Services** – wykonanie operacji **ViewData** dostępnej z wyskakującego menu po naciśnięciu tabeli **TEgzemplarz** prawym klawiszem myszy



ID	DTYPE	NUMER	MTYTUL_K...	TERMIN
52	TEgzemplarz	1	2	NULL
53	TEgzemplarz...	2	2	2008-04-19
102	TEgzemplarz...	1	3	2008-04-18
202	TEgzemplarz...	1	152	2008-04-18
252	TEgzemplarz...	3	3	2008-04-21
302	TEgzemplarz...	3	2	2008-04-19
352	TEgzemplarz	2	152	NULL
402	TEgzemplarz	4	152	NULL
452	TEgzemplarz	5	152	NULL
453	TEgzemplarz	4	2	NULL
502	TEgzemplarz	2	3	NULL
552	TEgzemplarz	4	3	NULL
602	TEgzemplarz	5	3	NULL
652	TEgzemplarz	6	3	NULL
653	TEgzemplarz	8	3	NULL
702	TEgzemplarz	9	3	NULL
752	TEgzemplarz	10	2	NULL
802	TEgzemplarz	12	2	NULL
902	TEgzemplarz	11	3	NULL
952	TEgzemplarz	7	3	NULL
1002	TEgzemplarz	13	3	NULL
1052	TEgzemplarz	14	3	NULL
1102	TEgzemplarz	15	3	NULL
1152	TEgzemplarz	16	3	NULL
1202	TEgzemplarz	17	3	NULL
1252	TEgzemplarz	18	3	NULL
1302	TEgzemplarz	19	3	NULL
1303	TEgzemplarz	20	3	NULL
1353	TEgzemplarz	10	3	NULL