

# **Tworzenie systemów informatycznych**

**Inżynieria oprogramowania  
Zofia Kruczkiewicz**

# Zagadnienia

1. Wielowarstwowa architektura systemu informatycznego
2. Refaktoryzacja architektury wielowarstwowej systemu informatycznego
3. Wzorce projektowe
4. Przykład **warstwy biznesowej** stosującej wzorce obiektowe
5. Przykłady architektury wielowarstwowej w środowisku Visual Web Java server Faces
6. Tworzenie **warstwy zasobów** - bazy danych w systemie baz danych Derby
7. Tworzenie **warstwy integracji** w projekcie Java Application. Zastosowanie wzorców projektowych typu **Domain Store** i **Transfer Object**.
8. Tworzenie **warstwy prezentacji**  
Pierwszy etap – tworzenie stron typu JSP
9. Uwierzytelnianie i autoryzacja oprogramowania  
Drugi etap tworzenia warstwy prezentacji

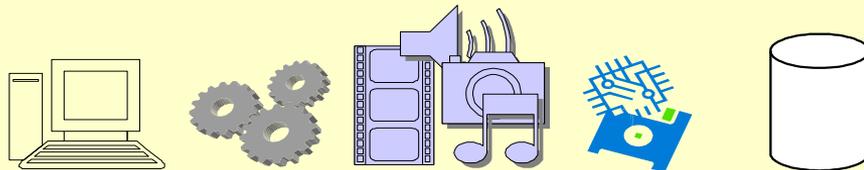
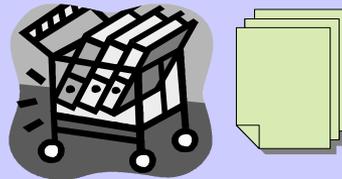
# **1. Wielowarstwowa architektura systemu informatycznego**

# Definicja systemu informatycznego

**Nieformalny system informacyjny:**  
zasoby osobowe - ludzie



**Formalny system informacyjny:**  
procedury zarządzania,  
bazy wiedzy



**Techniczny system informacyjny:**

- Sprzęt
- Oprogramowanie
- Bazy danych, bazy wiedzy

**System informatyczny**

jest to zbiór powiązanych ze sobą elementów

**nieformalnych,**

**formalnych i**

**technicznych,**

którego funkcją

jest przetwarzanie danych

Techniczny system informacyjny

- zorganizowany zespół środków technicznych (komputerów, oprogramowania, urządzeń teletransmisyjnych itp.)
- służący do gromadzenia, przetwarzania i przesyłania informacji

# Pięciowarstwowy model logicznego rozdzielania zadań

(wg. D.Alur, J.Crupi, D. Malks, Core J2EE. Wzorce projektowe.)

## Warstwa klienta

Klienci aplikacji, aplety, aplikacje i inne elementy z graficznym interfejsem użytkownika

Interakcja z użytkownikiem, urządzenia i prezentacja interfejsu użytkownika

## Warstwa prezentacji

Strony JSP, serwlety i inne elementy interfejsu użytkownika

Logowanie, zarządzanie sesją, tworzenie zawartości, formatowanie i dostarczanie

## Warstwa biznesowa

Komponenty EJB i inne obiekty biznesowe

Logika biznesowa, transakcje, dane i usługi

## Warstwa integracji

JMS, JDBC, konektory i połączenia z systemami zewnętrznymi

Adaptory zasobów, systemy zewnętrzne, mechanizmy zasobów, przepływ sterowania

## Warstwa zasobów

Bazy danych, systemy zewnętrzne i pozostałe zasoby

Zasoby, dane i usługi zewnętrzne

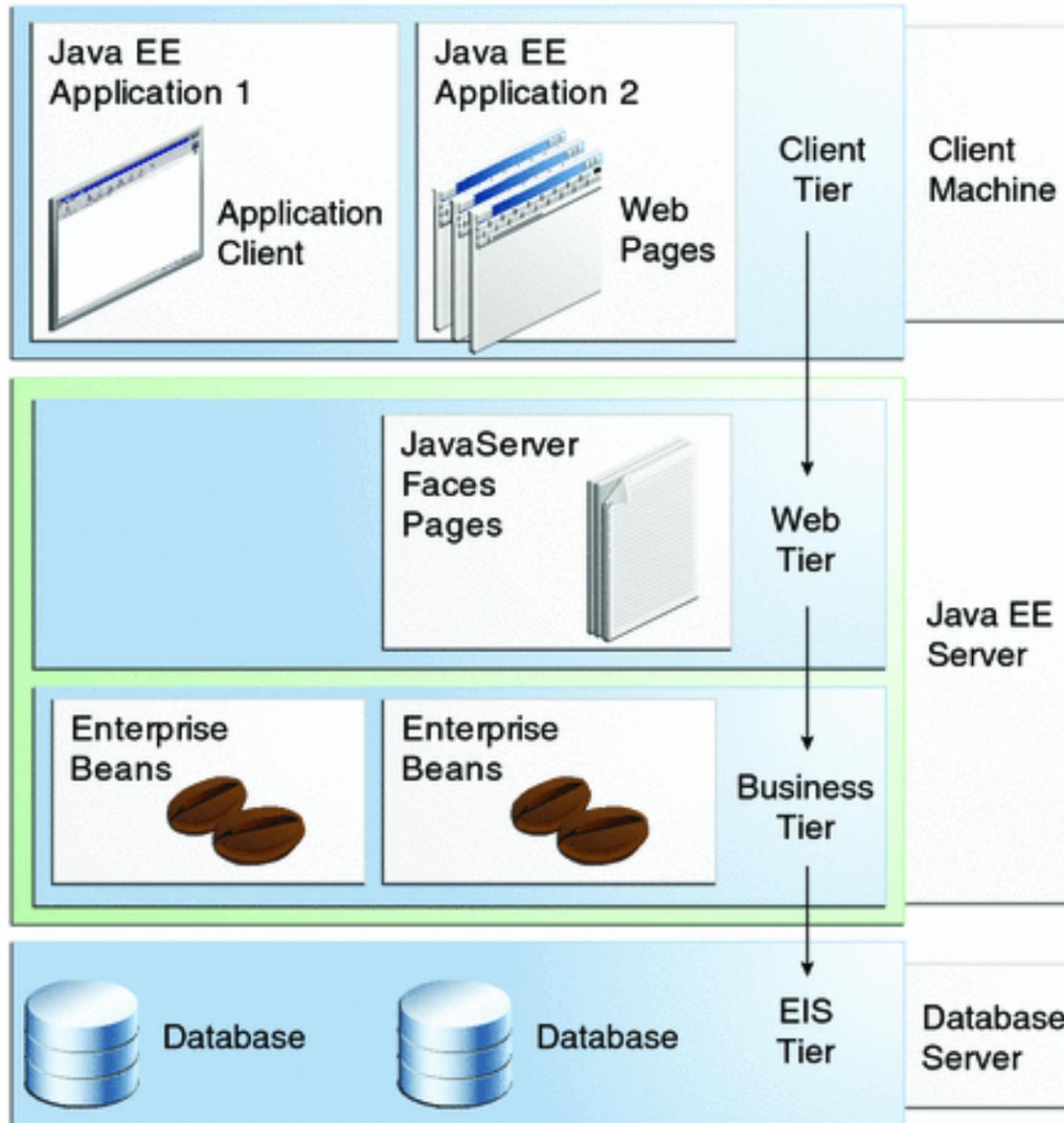
# (1) Warstwy aplikacji (Java EE)

## **Komponent:**

- **skompilowany moduł programowy,**
- **funkcjonalność dostarczana za pomocą interfejsu,**
- **zdolny do współdziałania z innymi komponentami oraz innymi częściami systemu informatycznego.**

# (2) Warstwy aplikacji (Java EE)

<http://download.oracle.com/javaee/6/tutorial/doc/bnaay.html>



# 2. Cel refaktoryzacji architektury wielowarstwowej systemu informatycznego

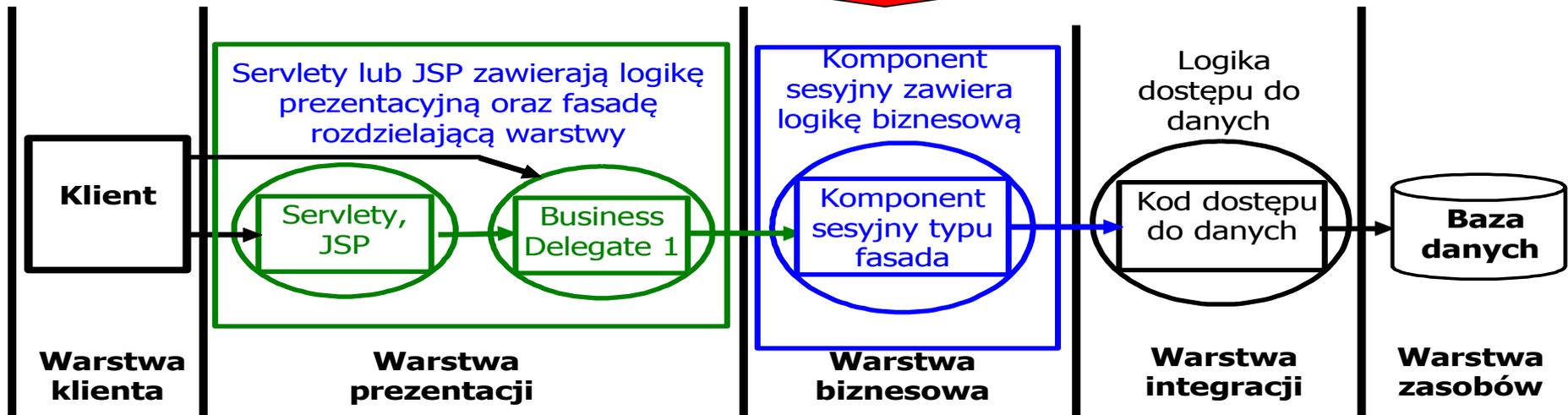
(wg. D.Alur, J.Crupi, D. Malks, Core J2EE. Wzorce projektowe.)

**Refaktoryzacja to poprawa struktury oprogramowania bez utraty funkcjonalności – w celu poprawy:**

- **wydajności**
- **funkcjonalności**
- **kosztu**
- **jakości oprogramowania:**
  - **Testowalności**
  - **Pielęgnowalności**
  - **Wieloużywalności**
  - **Zrozumiałości**
  - **Stopnia osiągniętej abstrakcji**

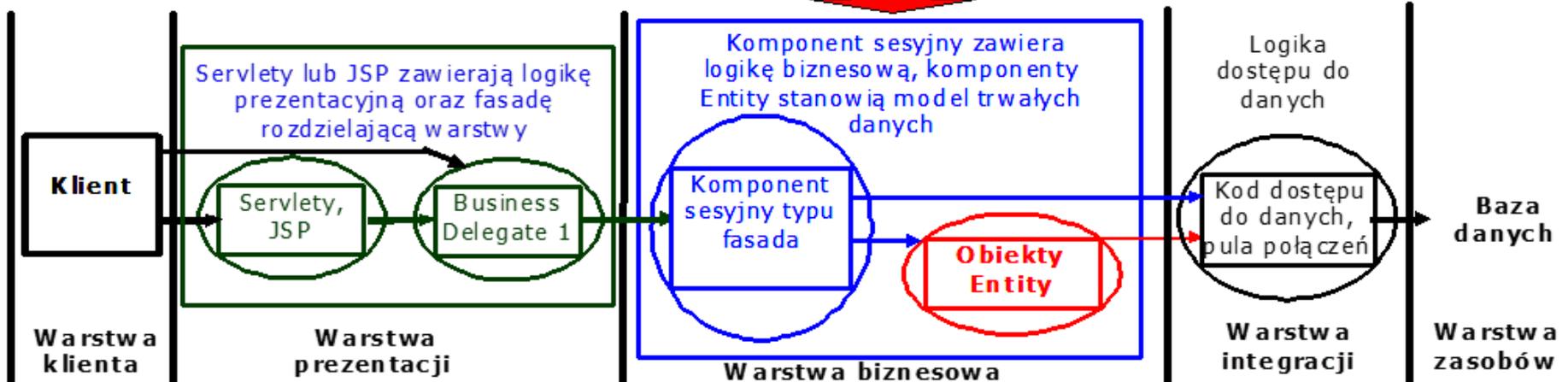
# Refaktoryzacja architektury wielowarstwowej 1

Należy przenieść kod dostępu do danych logicznie lub fizycznie bliżej rzeczywistego źródła danych, a logikę przetwarzania z klienta i warstwy prezentacji do warstwy biznesowej zawierającej **fasadowe komponenty sesyjne typu „Control”**.  
**Komponenty Business Delegate typu „Control”** hermetyzują dostęp do warstwy biznesowej z warstwy prezentacji – stanowią przedłużenie warstwy biznesowej.



## Refaktoryzacja architektury wielowarstwowej 2

Należy przenieść kod dostępu do danych logicznie lub fizycznie bliżej rzeczywistego źródła danych, a złożoną logikę przetwarzania z klienta i warstwy prezentacji typu do warstwy biznesowej zawierającą **obiekty danych typu „Entity”** i **hermetyzujące dostęp do tych komponentów fasadowe komponenty sesyjne typu „Control”**. **Komponenty Business Delegate typu „Control”** hermetyzują dostęp do warstwy biznesowej z warstwy prezentacji.



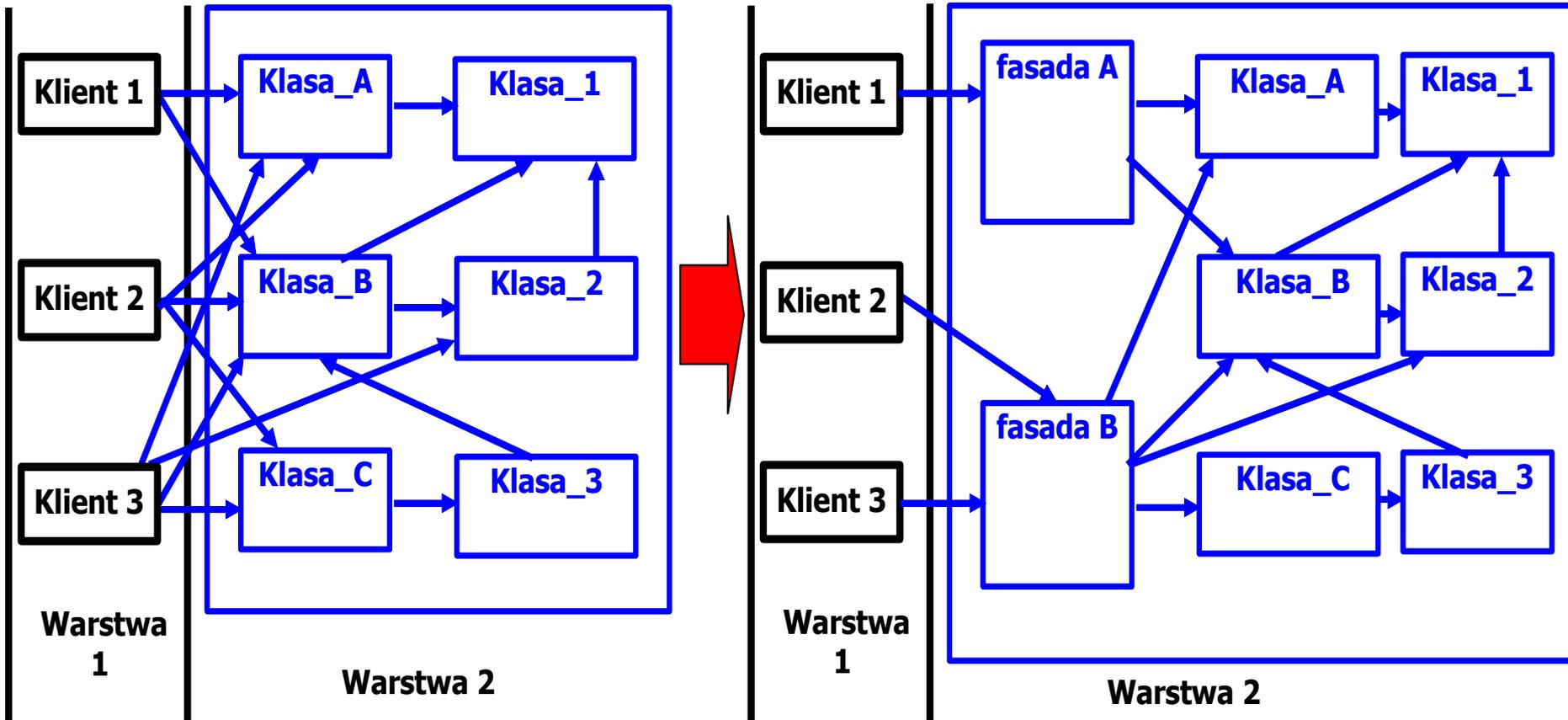
# **3. Wzorce projektowe**

# Wzorce projektowe

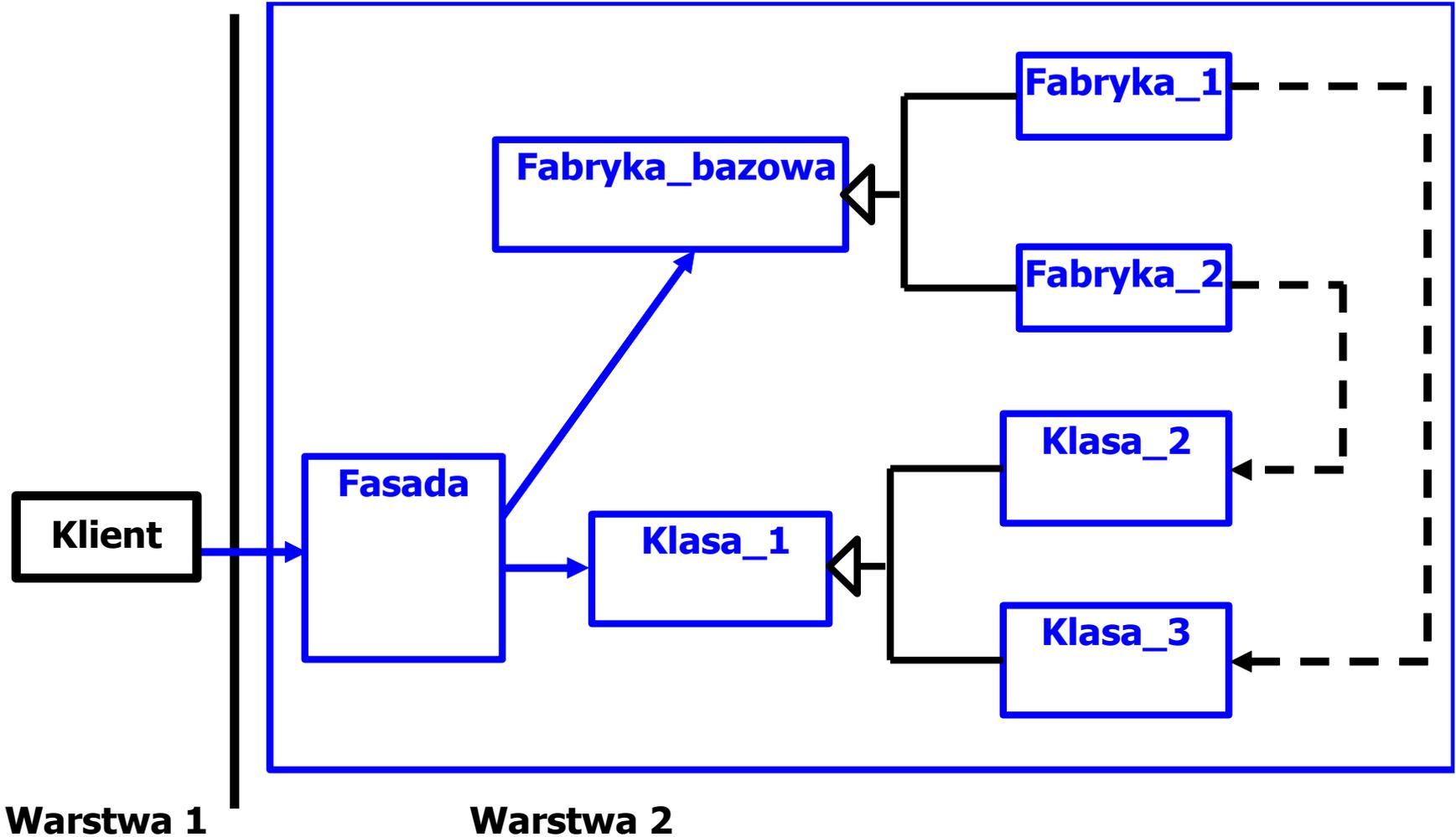
- Dobrze zbudowany system obiektowy jest pełen wzorców obiektowych
- Wzorzec to zwyczajowo przyjęte rozwiązanie typowego problemu w danym kontekście
- Strukturę wzorca przedstawia się w postaci diagramu klas
- Zachowanie się wzorca przedstawia się za pomocą diagramu sekwencji
- Wzorce projektowe: Wzorzec reprezentuje powiązanie problemu z rozwiązaniem  
**(wg Booch G., Rumbaugh J., Jacobson I., UML przewodnik użytkownika)**

- Wzorzec to pomysł, który okazał się użyteczny w jednym rzeczywistym kontekście i prawdopodobnie będzie użyteczny w innym. **(Martin Fowler)**

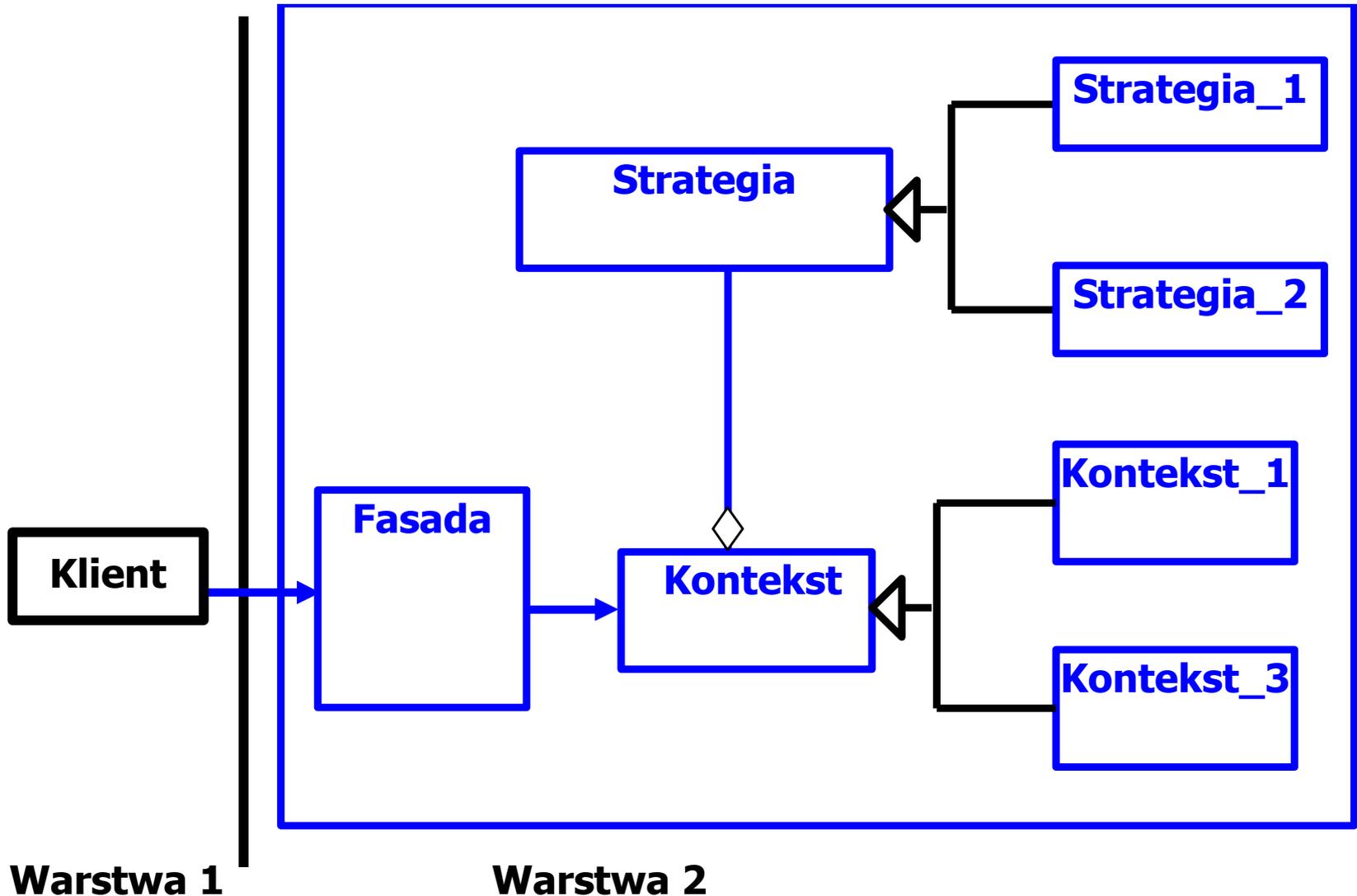
### 3.1. Wzorzec fasady (wzorzec strukturalny) – hermetyzacja logiki biznesowej



### 3.2. Wzorzec fabryki obiektów (wzorzec kreacyjny) – oddzielenie tworzenia obiektów od zarządzania nimi i używania ich



### 3.3. Wzorzec strategii (wzorzec czynnościowy) – zastosowanie polimorfizmu do wyboru algorytmu



# **4. Przykład warstwy usług należącej do warstwy biznesowej. Zastosowanie UML do modelowania tej warstwy**

# **System informacyjny „Katalogu tytułów i książek”**

1. Opis biznesowy „świata rzeczywistego”
2. Sformułowanie wymagań funkcjonalnych i niefunkcjonalnych aplikacji
3. Model analizy aplikacji oparty na diagramie przypadków użycia
4. Model projektowy warstwy biznesowej oparty na diagramie klas i diagramie sekwencji tworzony metodą iteracyjno-rozwojową sterowany realizacją przypadków użycia
5. Implementacja warstwy biznesowej tworzona w cyklu iteracyjno-rozwojowym sterowana rozwojem modelu projektowego

## **4.1. Opis biznesowy „świata rzeczywistego” w języku klienta - „Katalog tytułów i książek”**

### **1. Opis zasobów ludzkich**

Pracownik wypożyczalni może dodawać do katalogu tytułów nowe tytuły. Każdy tytuł jest reprezentowany przez następujące dane: tytuł, autor, wydawnictwo, ISBN oraz informacje o liczbie egzemplarzy i miejscu ich przechowywania i występuje w bibliotece jako pojedyncza informacja dla każdego tytułu. Pewna grupa tytułów opisuje książki nagrane na kasety, dlatego dodatkowo tytuł zawiera dane nagrania np nazwisko aktora. Każdy egzemplarz, niezależnie, czy jest książką czy kasetą, jest opisany odrębną informacją zawierającą numer egzemplarza i ewentualnie (dotyczy to wyodrębnionych egzemplarzy) informację o liczbie dni, na które można wypożyczyć egzemplarz. Numery egzemplarzy mogą się powtarzać dla różnych tytułów. Pracownik biblioteki (bibliotekarz) może dodawać nowe tytuły i egzemplarze oraz je przeszukiwać, natomiast klient może jedynie przeszukiwać tytuły i sprawdzać egzemplarze wybranych tytułów.

### **2. Przepisy**

Pracownik ponosi odpowiedzialność za poprawność danych - odpowiada materialnie za niezgodność danych ze stanem wypożyczalni.

### **3. Dane techniczne**

Klient może przeglądać dane wypożyczalni za pośrednictwem strony internetowej lub bezpośrednio za pomocą specjalnego programu. Pracownik biblioteki może dodatkowo wstawiać, modyfikować i usuwać dane o tytułach oraz egzemplarzach. Zakłada się, że klientów jednocześnie przeglądających dane wypożyczalni może być ponad 1000 oraz wypożyczalnia może zawierać kilkadziesiąt tysięcy tytułów oraz przynajmniej dwukrotnie więcej egzemplarzy. Biblioteka składa się z kilku ośrodków w różnych miastach na terenie kraju (lista miast jest dołączona do umowy). Zaleca się stosowanie technologii Java.

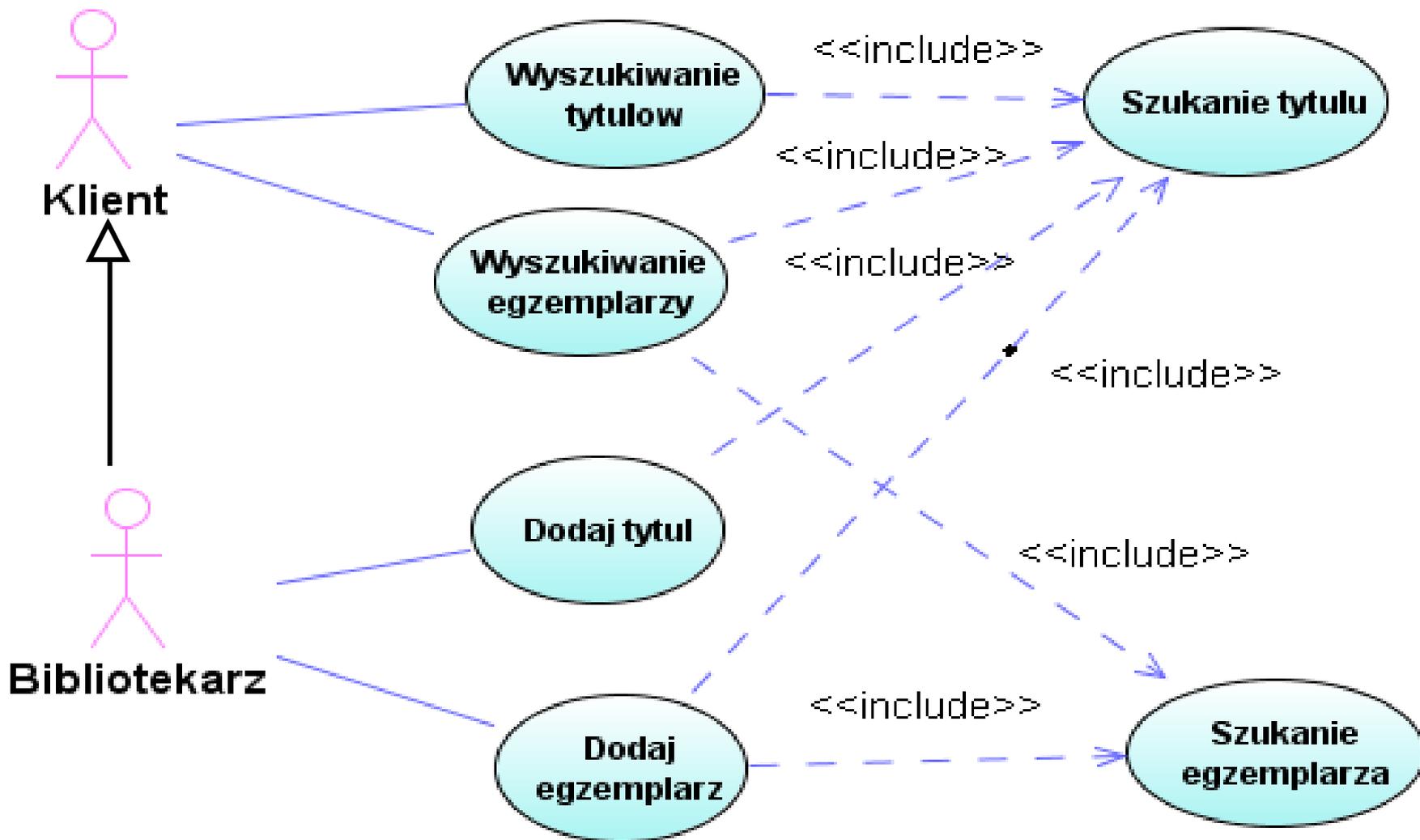
## **4.2. Lista wymagań funkcjonalnych całej aplikacji**

1. System zawiera katalog tytułów
2. System zawiera dwa typy egzemplarzy do wypożyczenia: książki i kasety z nagraniami dźwiękowymi książek.
3. Każdy egzemplarz zawiera tytuł, nazwisko autora, ISBN, wydawnictwo, jeśli jest to książka oraz dodatkowo nazwisko aktora, jeżeli jest to nagranie dźwiękowe.
  4. Może wystąpić wiele egzemplarzy książek oraz kaset z tymi samymi tytułami. Każdy egzemplarz, zarówno książka i kasecie, posiadają numer niepowtarzający się w ramach pozostałych identycznych danych (ISBN lub ISBN i nazwisko aktora).
4. W celu znalezienia tytułu należy podać ISBN lub ISBN i nazwisko aktora, jeżeli należy odszukać tytuł nagranej książki.
5. W celu wybrania właściwego egzemplarza należy podać ISBN, jeśli jest to książka oraz dodatkowo nazwisko aktora, jeśli jest to kasecie oraz numer egzemplarza.
6. Zarówno egzemplarze typu książka lub kasecie, mogą być przeznaczane do wypożyczenia na okres umowny oraz na okres ściśle określony.

## **Lista wymagań нефunkcjonalnych**

1. Wstawianie danych o tytułach i egzemplarzach może odbywać się tylko przez uprawnione osoby
2. Wyszukiwanie informacji powinno odbywać się samodzielnie przez klienta
3. Operacje zarządzania i wyszukiwania informacji mogą być dokonane przez Internet przez aplikację uruchamianą przez przeglądarkę lub bez jej pośrednictwa

### 4.3. Diagram przypadków użycia całej aplikacji „Katalog tytułów i książek”



<b>AKTOR</b>	<b>OPIS</b>	<b>PRZYPADKI UŻYCIA</b>
Bibliotekarz	<p><i>Bibliotekarz jest odpowiedzialny za utrzymywanie zasobów biblioteki (wstawianie i usuwanie: tytułów książek, egzemplarzy książek). Może on również przeszukiwać zasoby katalog tytułów i egzemplarzy książek</i></p>	<ul style="list-style-type: none"> <li>• Dodaj tytuł</li> <li>• Dodaj egzemplarz</li> <li>• Wyszukiwanie tytułów</li> <li>• Wyszukiwanie egzemplarzy</li> </ul>
Klient	<p><i>Klient może jedynie przeszukiwać zasoby katalog tytułów i egzemplarzy książek</i></p>	<ul style="list-style-type: none"> <li>• Wyszukiwanie tytułów</li> <li>• Wyszukiwanie egzemplarzy</li> </ul>

## **PU Szukanie tytułu**

### **OPIS**

#### **CEL: Poszukiwanie tytułu**

**WS (warunki wstępne):** inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)

**WK (warunki końcowe):** podanie tytułu zawierającego identyczne dane, jakie posiada tytuł wzorcowy lub podanie informacji o braku tytułu

#### **PRZEBIEG:**

1. Szukanie tytułu przebiega według atrybutów: ISBN (obowiązkowo) oraz aktor (jeśli jest to wymagane) zgodnie z danymi tytułu podanego do przypadku użycia
2. Jeśli istnieje tytuł o podanych atrybutach, zwracany jest tytuł z zasobów wypożyczalni, w przeciwnym wypadku zwracana jest informacja o braku tytułu.

## **PU Wyszukiwanie tytułów**

### **OPIS**

#### **CEL: Wyszukiwanie tytułów**

**WS (warunki wstępne):** inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)

**WK (warunki końcowe):** wyszukanie tytułu o podanych atrybutach obowiązkowych ISBN lub ISBN i aktor w przypadku nagrania dźwiękowego lub podanie informacji o braku tytułu

#### **PRZEBIEG:**

1. Należy podać atrybuty tytułu: ISBN jako obowiązkowa dana oraz dodatkowo aktor, jeśli poszukiwany jest tytuł książki jako nagranie dźwiękowe. Tworzony jest tytuł wzorcowy do wyszukiwania rzeczywistego tytułu
2. Należy wywołać **PU Szukanie tytułu**. Należy sprawdzić, czy tytuł o podanych atrybutach już istnieje. Jeśli nie, należy zakończyć PU podając informację o braku tytułu, w przeciwnym wypadku należy podać znaleziony tytuł.

## **PU Szukanie egzemplarza**

### **OPIS**

#### **CEL: Poszukiwanie egzemplarza**

**WS (warunki wstępne):** inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)

**WK (warunki końcowe):** podanie egzemplarza zawierającego identyczne dane, jakie posiada egzemplarz wzorcowy lub podanie informacji o braku egzemplarza

#### **PRZEBIEG:**

1. Szukanie egzemplarza przebiega według atrybutu: numer egzemplarza (obowiązkowo) zgodnie z danymi tytułu podanego do przypadku użycia. Przeszukiwane są egzemplarze należące do konkretnego tytułu
2. Jeśli istnieje egzemplarz o podanym numerze, zwracany jest egzemplarz z zasobów wypożyczalni, w przeciwnym wypadku zwracana jest informacja o braku egzemplarza.

## **PU Wyszukiwanie egzemplarzy**

### **OPIS**

#### **CEL: Wyszukiwanie egzemplarzy książek o podanym tytule**

**WS (warunki wstępne):** inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)

**WK (warunki końcowe):** wyszukanie egzemplarza o tytule zgodnym z podanymi atrybutami obowiązkowymi ISBN lub ISBN i aktor w przypadku nagrania dźwiękowego oraz podanym numerze lub podanie informacji o braku egzemplarza

#### **PRZEBIEG:**

1. Należy podać atrybuty tytułu: ISBN jako obowiązkowa dana oraz dodatkowo aktor, jeśli poszukiwany jest tytuł książki jako nagranie dźwiękowe. Tworzony jest tytuł wzorcowy do wyszukiwania rzeczywistego tytułu
2. Należy wywołać **PU Szukanie tytułu**. Należy sprawdzić, czy tytuł o podanych atrybutach już istnieje. Jeśli nie, należy zakończyć PU podając informację o braku tytułu.
3. Należy utworzyć wzorcowy egzemplarz zawierający numer podany do wyszukiwania egzemplarza i przekazać go do **PU Szukanie egzemplarza**. Wynik podany przez wywołany PU należy podać jako wynik końcowy.

## **PU Dodaj tytuł**

### **OPIS**

#### **CEL: Wstawienie nowego tytułu**

**WS (warunki wstępne):** inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)

**WK (warunki końcowe):** dodanie tytułu o podanych atrybutach obowiązkowych: tytuł, autor, ISBN, wydawnictwo oraz jeśli jest to nagranie dźwiękowe, to nazwisko aktora lub informacja o istnieniu takiego tytułu

#### **PRZEBIEG:**

1. Należy podać atrybuty tytułu: tytuł, autor, ISBN, wydawnictwo oraz jeśli jest to nagranie dźwiękowe, to nazwisko aktora. Należy utworzyć tytuł do wyszukiwania i ewentualnego wstawienia.
2. Należy wywołać **PU Szukanie tytułu**. Należy sprawdzić, czy tytuł o podanych atrybutach już istnieje. Jeśli tak, należy zakończyć PU, w przeciwnym wypadku należy wstawić nowy tytuł.

## **PU Dodaj egzemplarz**

### **OPIS**

#### **CEL: Wstawianie nowego egzemplarza**

**WS (warunki wstępne):** inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)

**WK (warunki końcowe):** wstawienie egzemplarza o tytule zgodnym z podanymi atrybutami obowiązkowymi ISBN lub ISBN i aktor w przypadku nagrania dźwiękowego oraz podanym numerze i ewentualnie atrybucie do określania terminu zwrotu, jeśli należy wstawić egzemplarz z wyznaczonym terminem zwrotu lub podanie informacji o istnieniu takiego egzemplarza

#### **PRZEBIEG:**

1. Należy podać atrybuty tytułu: ISBN jako obowiązkowa dana oraz dodatkowo aktor, jeśli poszukiwany jest tytuł książki jako nagranie dźwiękowe. Tworzony jest tytuł wzorcowy do wyszukiwania rzeczywistego tytułu
2. Należy wywołać **PU Szukanie tytułu**. Należy sprawdzić, czy tytuł o podanych atrybutach już istnieje. Jeśli nie, należy zakończyć PU podając informację o braku tytułu.
3. Należy utworzyć egzemplarz zawierający numer podany do wyszukiwania egzemplarza oraz atrybut terminu zwrotu, jeśli jest to wymagane i należy przekazać go do **PU Szukanie egzemplarza**. Jeśli nie istnieje egzemplarz o danym numerze, należy wstawić ten egzemplarz, w przeciwnym wypadku należy podać informację o istnieniu takiego egzemplarza.

## 4.4.1. Analiza wspólności – identyfikacja klas bazowych

Przypadki użycia	Wykryte klasy bazowe
<b>PU Szukanie tytułu</b> <b>PU Wyszukiwanie tytułów</b> <b>PU Dodaj tytuł</b>	klasa typu „Entity”: TTytul_ksiazki
<b>PU Szukanie egzemplarza</b> <b>PU Wyszukiwanie egzemplarzy</b> <b>PU Dodaj egzemplarz</b>	klasy typu „Entity”: TTytul_ksiazki (zawiera atrybuty tytułu, posiada książki – wstawia i wyszukuje je), TEgzemplarz (książka)

## 4.4.2. Analiza zmienności - identyfikacja podklas

Przypadki użycia	Wykryte podklasy
<p><b>PU Szukanie tytułu</b> <b>PU Wyszukiwanie tytułów</b> <b>PU Dodaj tytuł</b></p>	<p>klasa <b>TTytuł_książki_na_kasecie</b> typu „Entity”, która dziedziczy od klasy <b>TTytuł_książki</b></p>
<p><b>PU Szukanie egzemplarza</b> <b>PU Wyszukiwanie egzemplarzy</b> <b>PU Dodaj egzemplarz</b></p>	<p>Wyróżniono egzemplarze zwykłe typu <b>TEgzemplarz</b>, oraz egzemplarze <b>TEgzemplarz_termin</b> z dodatkowo oznaczonym terminem oddania - rozróżniane w ramach danego tytułu książki zwykłej (<b>TTytuł_książki</b>) lub nagranej w postaci dźwiękowej (<b>TTytuł_książki_na_kasecie</b>) za pomocą numeru</p>

### 4.4.3. Analiza wspólności i zmienności - identyfikacja typów relacji

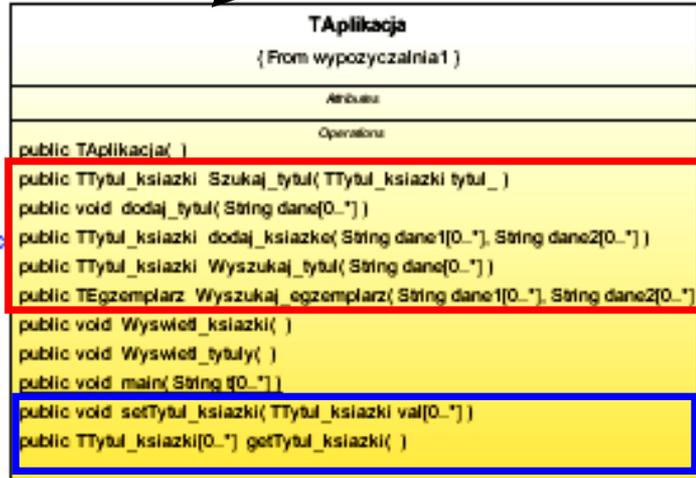
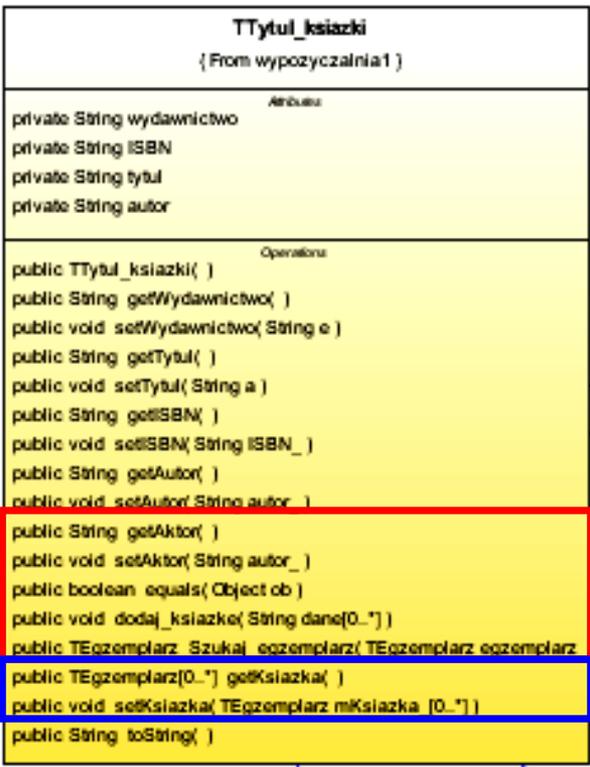
- Zależność między obiektami typu **TTytul\_książki** oraz **TEgzemplarz** są w relacji **1 do 0..\***. Związek ten dziedziczą obiekty typu **TTytul\_książki\_na\_kasecie**.
- Związek **0..\* do 1** między obiektami typu **TEgzemplarz** oraz **TTytul\_książki** są dziedziczone przez obiekty typu **TEgzemplarz\_termin**. Stąd zwykłe książki mogą być oznaczone jedynie numerami lub numerami i terminem zwrotu. Dotyczy to również książek w postaci nagrań dźwiękowych.

### 4.4.4. Analiza wspólności i zmienności - identyfikacja wzorców projektowych

- Wykryto związki silnej agregacji między tytułem i egzemplarzem – egzemplarz nie może istnieć bez tytułu. Wybrano **wzorzec strategii** do implementacji obiektów typu **TEgzemplarz**
- Zastosowano klasę **TAplikacja typu „Control”** jako **wzorzec fasady** do oddzielenia obiektów typu „Entity” od pozostałej części systemu oraz **klasę typu „Control”** jako **wzorzec fabryki obiektów (TFabryka)** do tworzenia różnych typów tytułów oraz egzemplarzy.

# Wzorzec fasady

- Implementacja powiązań
- Metody przypadków użycia
- Decyzja projektowa



Wzorzec strategii

Wzorzec fabryki obiektów

**4.5. Model projektowy warstwy biznesowej** oparty na diagramie klas i diagramie sekwencji tworzony metodą iteracyjno-rozwojową sterowany realizacją przypadków użycia

**4.6. Implementacja warstwy biznesowej** tworzona w cyklu iteracyjno-rozwojowym sterowana rozwojem modelu projektowego

Projekt przypadku użycia

„ **Szukanie tytułu** ”

za pomocą diagramu sekwencji i diagramu klas.

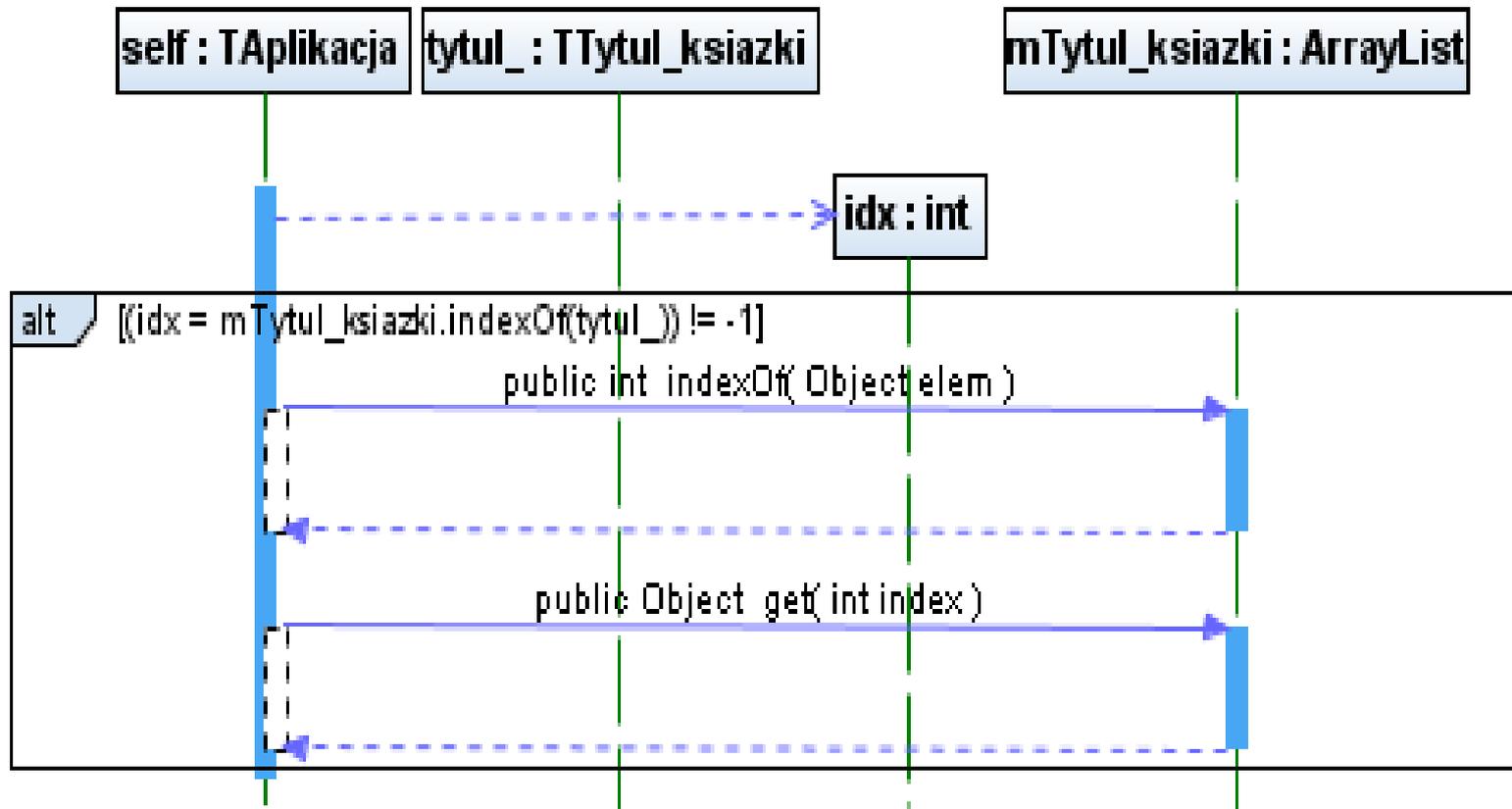
Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia

na podstawie diagramów sekwencji

# (1) Szukanie tytułu

(TTytul\_książki TAplikacja::Szukaj\_tytul(TTytul\_książki tytul\_))



# boolean TTytul\_książki::equals(Object ob)



Projekt przypadku użycia

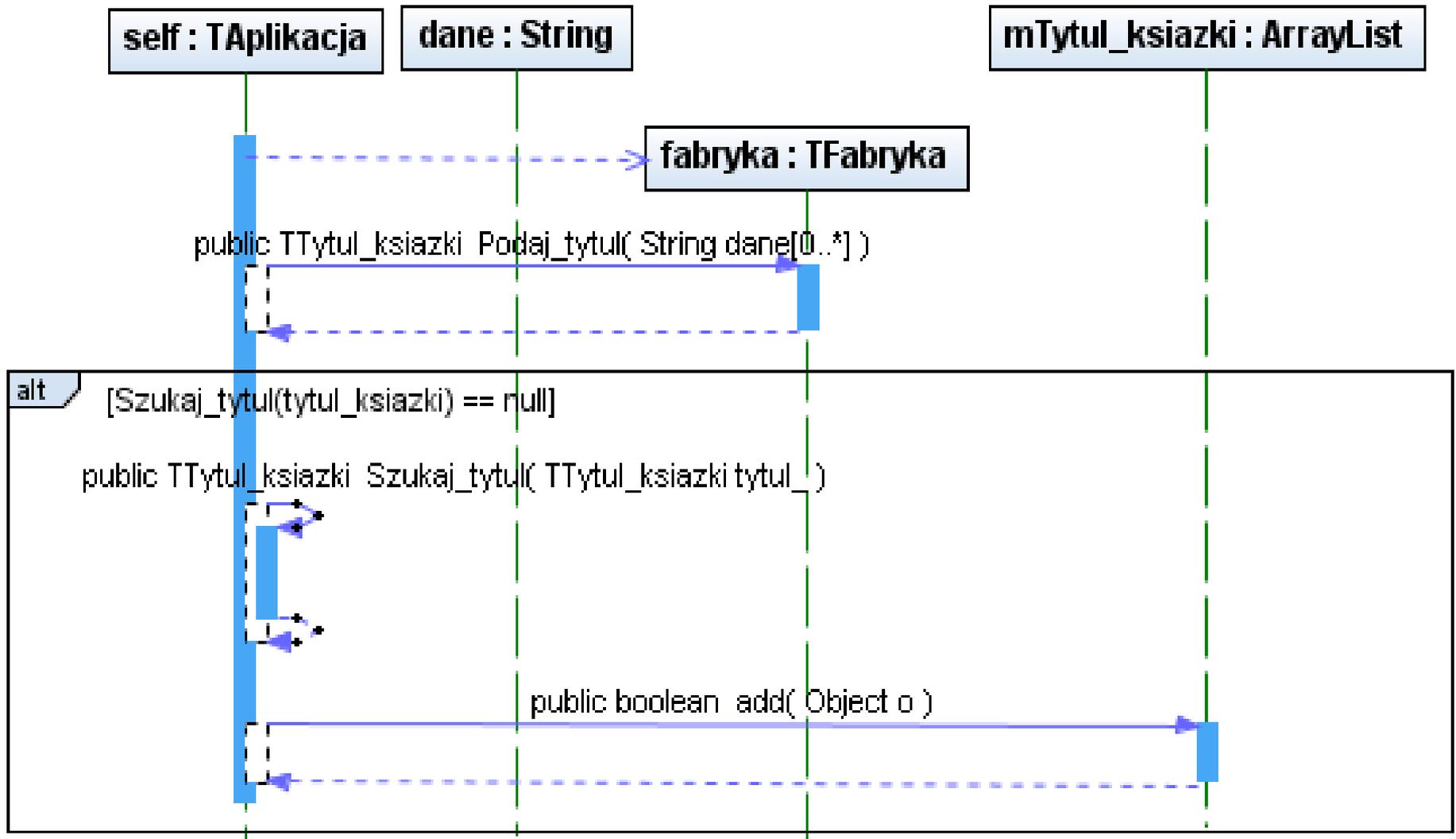
„ **Dodaj tytuł** ”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

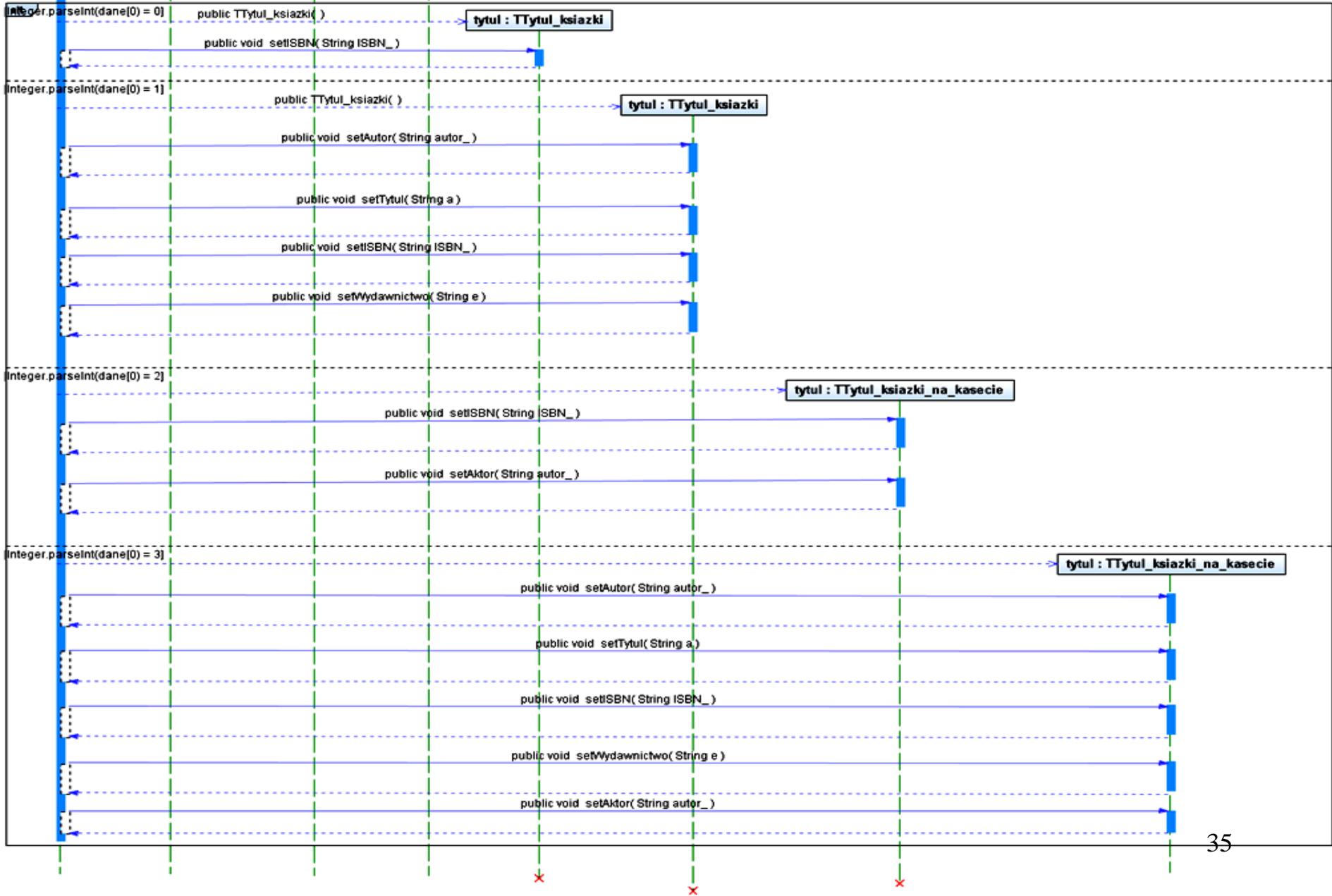
Definiowanie kodu metod realizujących przypadek użycia na podstawie diagramów sekwencji

## (2) Dodaj tytuł

**void TAplikacja::dodaj\_tytul(String dane[])**



# TTytul\_książki TFabryka::Podaj\_tytul (String[] dane)



```
C:\Users\kruczkiewicz>java -jar "E:\Dydaktyka\Wzorce\Wypożyczalnia\dist\Wypożyczalnia.jar"  
[Tytuł: 1 Autor:1 ISBN: 1 Wydawnictwo:1, Tytuł: 2 Autor:2 ISBN: 2 Wydawnictwo:2, Tytuł: 3 Autor:3 ISBN: 3 Wydawnictwo:3, Tytuł: 1 Autor:1 ISBN:  
1 Wydawnictwo:1 Aktor:1, Tytuł: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Aktor:2, Tytuł: 4 Autor:4 ISBN: 4 Wydawnictwo:4 Aktor:4]
```

```
// ten kod powinien działać po uzupełnieniu kodu dla wskazanych klas  
// biorących udział w wykonanych przypadkach użycia oraz  
// po wykonaniu metody toString() w tych klasach (TTytuł_książki oraz TTytuł_książki_na_kasecie) i getTytuł_książki() w klasie TAplikacja  
public static void main(String t[]) // your code here  
{ TAplikacja ap = new TAplikacja();  
String t1[] = {"1", "1", "1", "1", "1"}; //t1, t2, t3 – tablice łańcuchów do tworzenia tytułu książki zwykłej – pierwszy łańcuch  
String t2[] = {"1", "2", "2", "2", "2"}; // jest informacją dla fabryki, jaki obiekt wygenerować  
String t3[] = {"1", "3", "3", "3", "3"}; // "1" oznacza utworzenie obiektu klasy TTytuł_książki, a pozostałe łańcuchy to kolejno  
// autor, tytuł, ISBN, wydawnictwo dla uproszczenia w postaci cyfr - obiekty do wstawiania  
String t4[] = {"3", "1", "1", "1", "1", "1"}; // t4, t5, t6 – tablice łańcuchów do tworzenia tytułu książki jako nagranie  
//dźwiękowe  
String t5[] = {"3", "2", "2", "2", "2", "2"}; // – pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować  
String t6[] = {"3", "4", "4", "4", "4", "4"}; // "3" oznacza utworzenie obiektu klasy TTytuł_książki_na_kasecie, a pozostałe  
// łańcuchy to kolejno autor, tytuł, ISBN, wydawnictwo, aktor dla uproszczenia w postaci cyfr- obiekty do wstawiania  
ap.dodaj_tytuł(t1);  
ap.dodaj_tytuł(t2);            ap.dodaj_tytuł(t2);  
ap.dodaj_tytuł(t3);  
ap.dodaj_tytuł(t4);  
ap.dodaj_tytuł(t5);            ap.dodaj_tytuł(t5);  
ap.dodaj_tytuł(t6);  
String lan = ap.getTytuł_książki().toString();  
System.out.println(lan);  
}
```

```
Tytuł: 4 Autor:4 ISBN: 4 Wydawnictwo:4 Aktor:4 Numer: 2
```

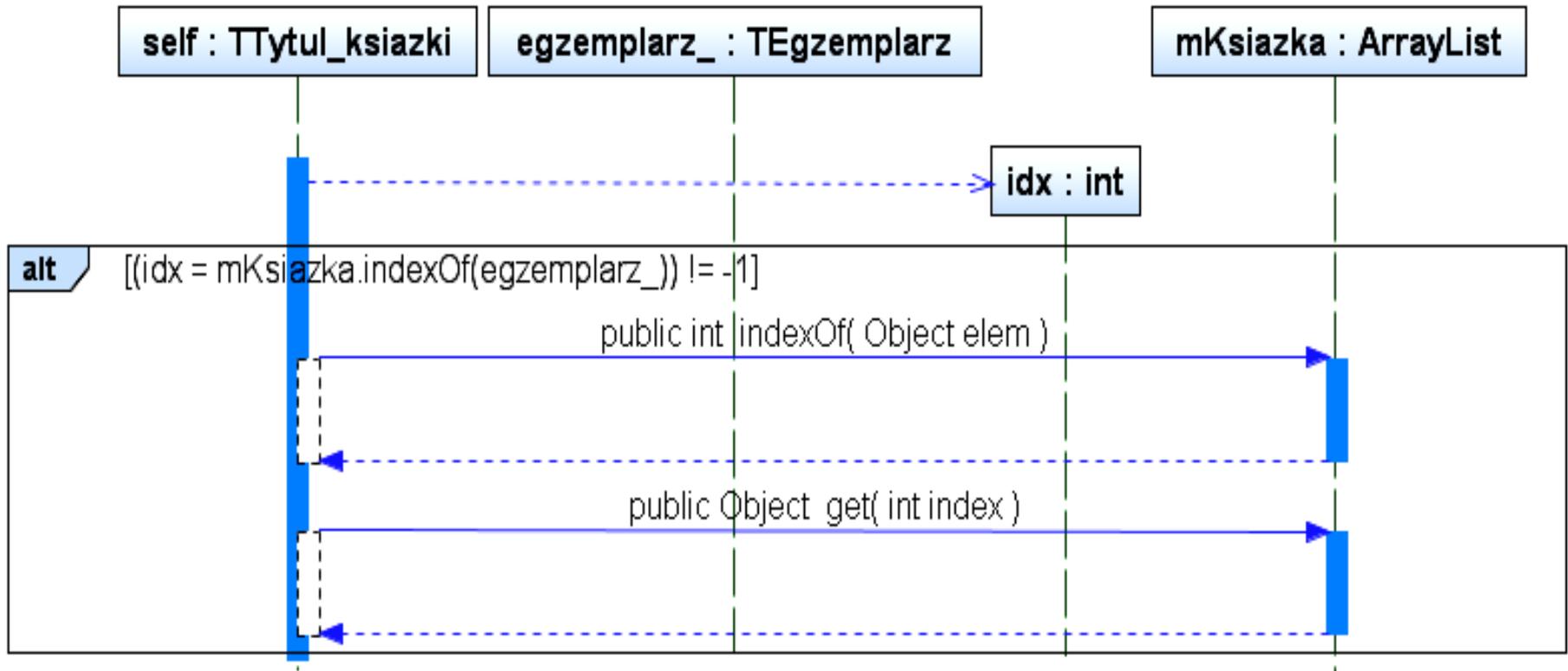
Projekt przypadku użycia  
**„Szukaj egzemplarz”**

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia na podstawie diagramów sekwencji

### (3) Szukaj egzemplarz

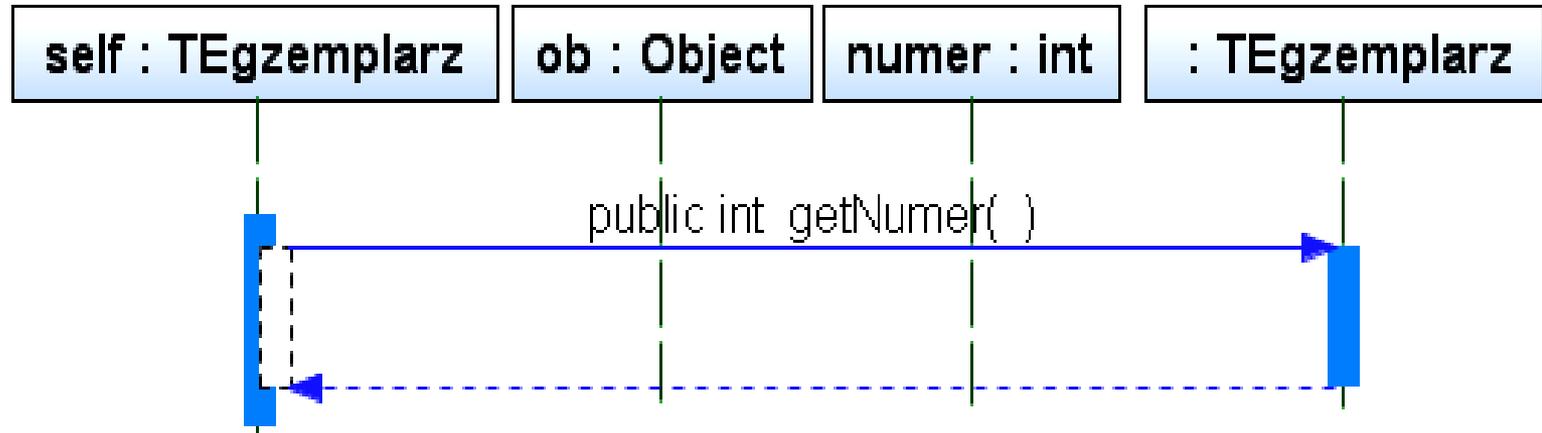
TEgzemplarz TTytul\_ksiazki::Szukaj\_egzemplarz(TEgzemplarz egzemplarz\_)



```
//TTytul_ksiazki
private ArrayList <TEgzemplarz> mKsiazka =
    new ArrayList < TEgzemplarz >();

public TEgzemplarz Szukaj_egzemplarz (TEgzemplarz egzemplarz_)
{
    int idx;
    if ((idx=mKsiazka.indexOf(egzemplarz_))!=-1 )
    {
        egzemplarz_=mKsiazka.get(idx);
        return egzemplarz;
    }
    return null;
}
```

# boolean TEgzemplarz::equals(Object ob)



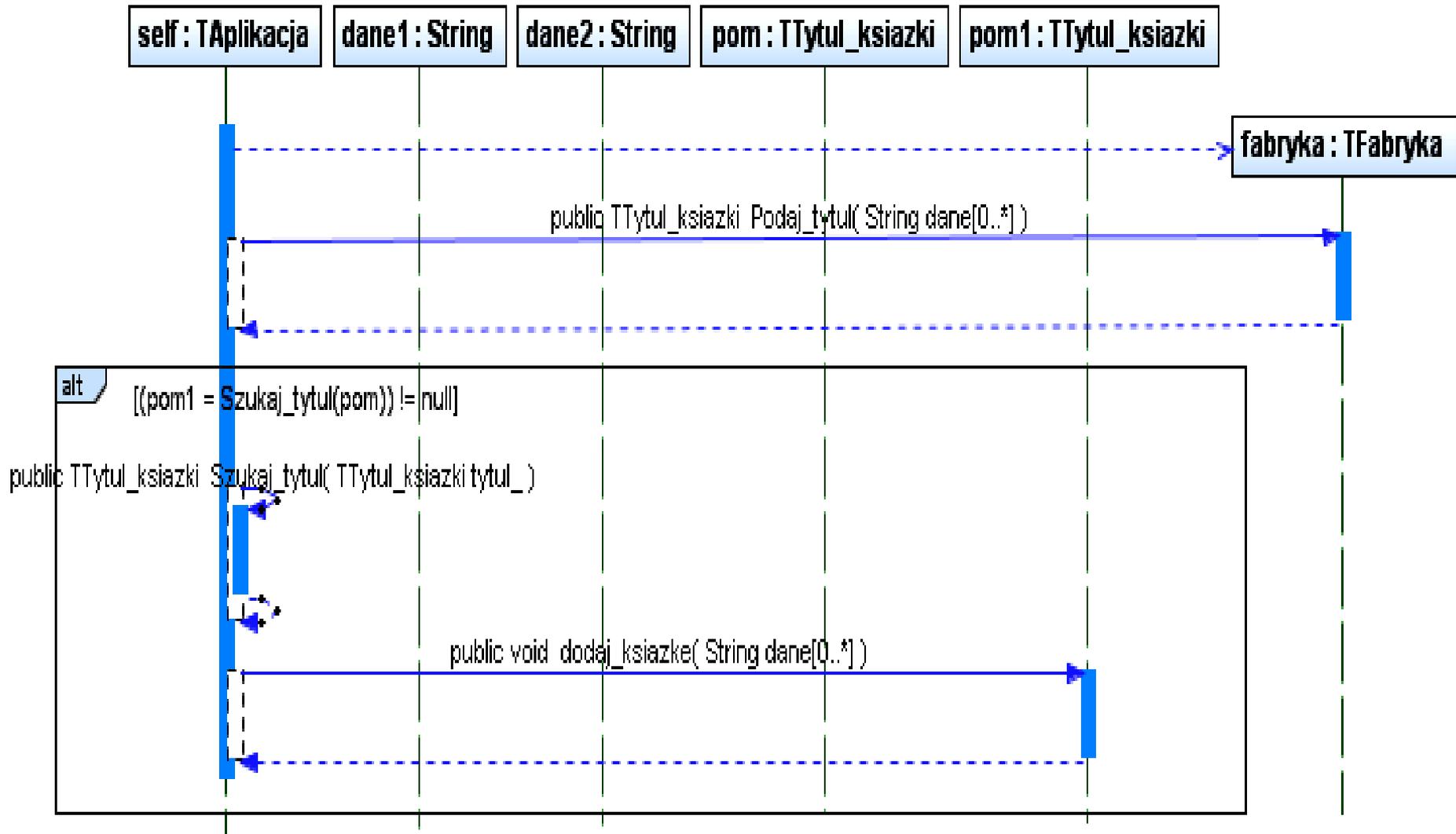
Projekt przypadku użycia  
„ **Dodaj egzemplarz** ”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia na podstawie diagramów sekwencji

## (4) Dodaj egzemplarz

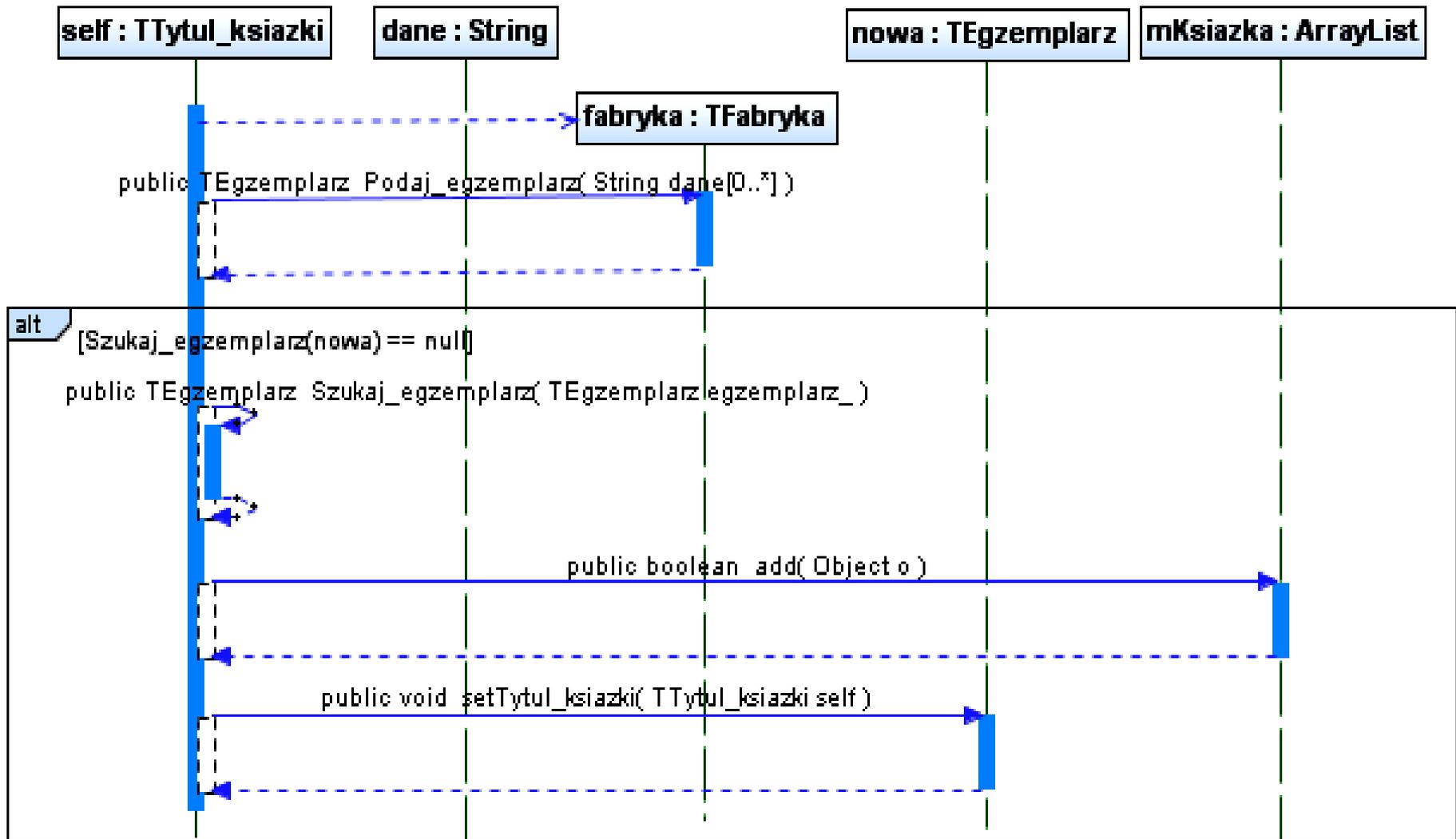
**TTytul\_książki** **TAplikacja::dodaj\_książke(String dane1[], String dane2[])**



```
//TAplikacja
```

```
public TTytul_książki dodaj_książke(String dane1[],  
    String dane2[])  
{  
    TTytul_książki pom, pom1 = null;  
    TFabryka fabryka = new TFabryka();  
    pom = fabryka.Podaj_tytul(dane1);  
    if ((pom1 = Szukaj_tytul(pom)) != null) {  
        pom1.dodaj_książke(dane2);  
    }  
    return pom1;  
}
```

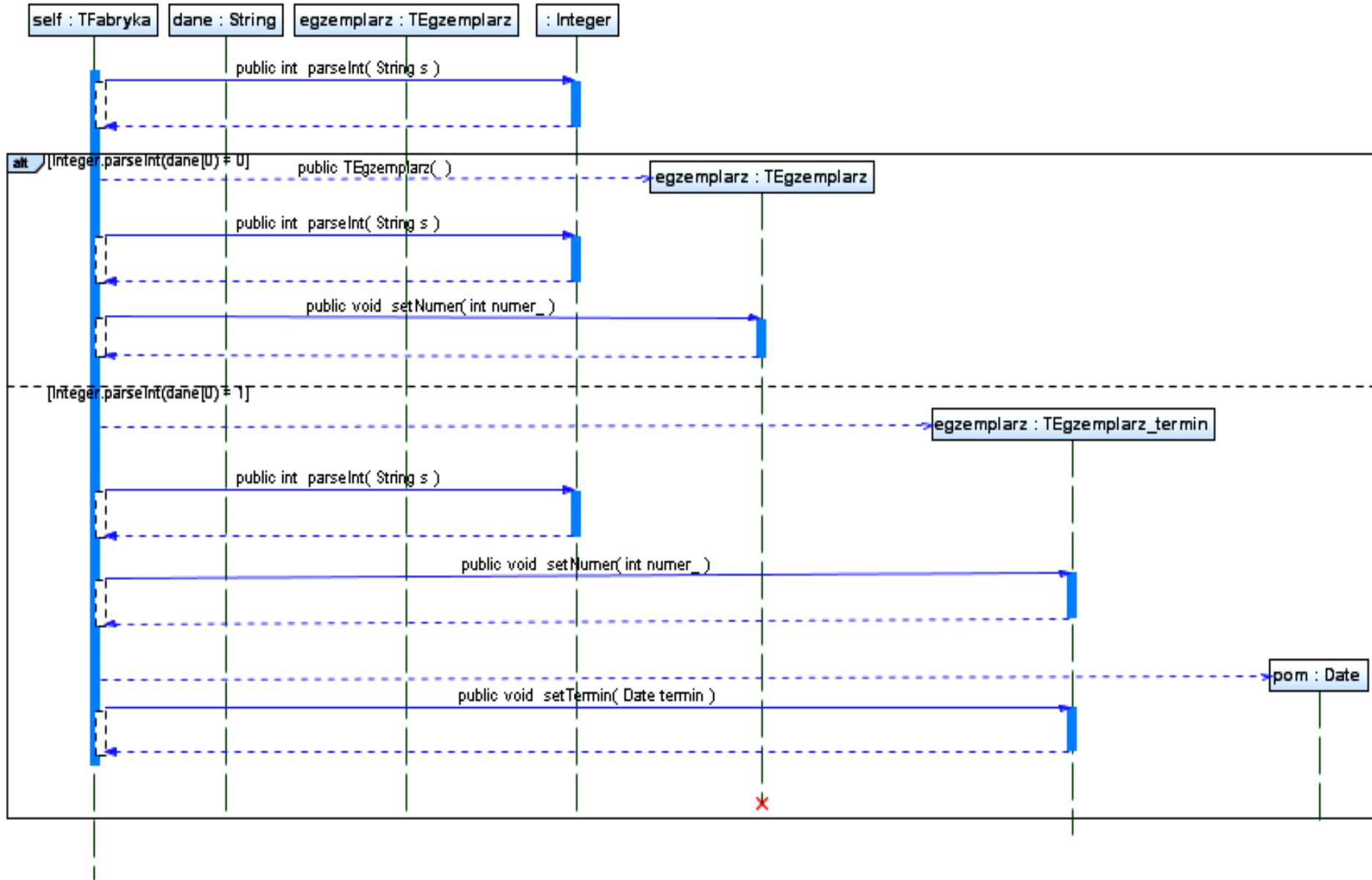
# void TTytul\_książki::dodaj\_książke(String dane[])



```
//TTytul_ksiazki
```

```
public void dodaj_ksiazke(String dane[])  
{  
    TFabryka fabryka = new TFabryka();  
    TEgzemplarz nowa;  
    nowa = fabryka.Podaj_egzemplarz(dane);  
    if (Szukaj_egzemplarz(nowa) == null) {  
        mKsiazka.add(nowa);  
        nowa.setTytul_ksiazki(this);  
    }  
}
```

# TEgzemplarz TFabryka::Podaj\_egzemplarz (String[] dane)



# Proponowany kod funkcji main w klasie fasadowej TAplikacja

```
public static void main(String t[]) // your code here
{
    TAplikacja ap = new TAplikacja();
    String t1[] = {"1", "1", "1", "1", "1"}; //t1, t2, t3 – tablice łańcuchów do tworzenia tytułu książki zwykłej – pierwszy łańcuch
    String t2[] = {"1", "2", "2", "2", "2"}; // jest informacją dla fabryki, jaki obiekt wygenerować
    String t3[] = {"1", "3", "3", "3", "3"}; //”1” oznacza utworzenie obiektu klasy TTytul_książki, a pozostałe łańcuchy to kolejno
// autor, tytuł, ISBN, wydawnictwo dla uproszczenia w postaci cyfr - obiekty do wstawiania
    String t4[] = {"3", "1", "1", "1", "1", "1"}; // t4, t5,t6 – tablice łańcuchów do tworzenia tytułu książki jako nagranie dźwiękowe
    String t5[] = {"3", "2", "2", "2", "2", "2"}; //– pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować
    String t6[] = {"3", "4", "4", "4", "4", "4"}; //”3” oznacza utworzenie obiektu klasy TTytul_książki_na_kasecie, a pozostałe
//łańcuchy to kolejno autor, tytuł, ISBN, wydawnictwo, aktor dla uproszczenia w postaci cyfr- obiekty do wstawiania
    ap.dodaj_tytul(t1);
    ap.dodaj_tytul(t2);      ap.dodaj_tytul(t2);
    ap.dodaj_tytul(t3);
    ap.dodaj_tytul(t4);
    ap.dodaj_tytul(t5);      ap.dodaj_tytul(t5);
    ap.dodaj_tytul(t6);
    String lan = ap.getTytul_książki().toString();
    System.out.println(lan);
    String d1[] = {"0", "1"}; // d1, d2, d3 - – tablice łańcuchów do tworzenia wzorcowego tytułu książki zwykłej do wyszukiwania
    String d2[] = {"0", "2"}; // – pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować: „0” oznacza generowanie
    String d3[] = {"0", "5"}; // obiektu klasy TTytul_książki, drugi łańcuch jest ISBN – obiekty do wyszukiwania
    String d4[] = {"2", "1", "1"}; //d4, d5 - tablice łańcuchów do tworzenia wzorcowego tytułu książki jako nagranie dźwiękowe
    String d5[] = {"2", "4", "4"}; //pierwszy łańcuch „2” oznacza generowanie obiektu typu TTytul_książki_na_kasecie
// drugi łańcuch to ISBN, trzeci jest nazwiskiem aktora - obiekty do wyszukiwania
    String tr1[] = {"0", "1"}; //tablice tr1 i tr2 zawierają informację o tworzeniu obiektu typu TEgzemplarz: pierwszy łańcuch
    String tr2[] = {"0", "2"}; //równy „0” oznacza tworzenie obiektu typu typu TEgzemplarz, drugi jest numerem egzemplarza
    String tr3[] = {"1", "3", "April 10, 2008, 00:00:00 GMT"}; //pierwszy łańcuch równy „1” oznacza tworzenie obiektu klasy
    String tr4[] = {"1", "2", "April 10, 2008, 00:00:00 GMT"}; //TEgzemplarz_termin, drugi oznacza numer, trzeci termin
}
```

```

TTytul_książki pom = ap.dodaj_książke(d1, tr1); //dodana
if (pom != null) {          System.out.println(pom.getKsiążka().toString());    }

pom = ap.dodaj_książke(d2, tr1); //dodana
if (pom != null) {          System.out.println(pom.getKsiążka().toString());    }

pom = ap.dodaj_książke(d2, tr1); //nie wstawi – powtórzenie numeru książki
if (pom != null) {          System.out.println(pom.getKsiążka().toString());    }

pom = ap.dodaj_książke(d2, tr2); //dodana
if (pom != null) {          System.out.println(pom.getKsiążka().toString());    }

pom = ap.dodaj_książke(d3, tr2); //nie wstawi – brak tytułu
if (pom != null) {          System.out.println(pom.getKsiążka().toString());    }

pom = ap.dodaj_książke(d4, tr3); //dodana
if (pom != null) {          System.out.println(pom.getKsiążka().toString());    }

pom = ap.dodaj_książke(d4, tr3); //nie wstawi – powtórzenie numeru książki
if (pom != null) {          System.out.println(pom.getKsiążka().toString());    }

pom = ap.dodaj_książke(d4, tr4);
if (pom != null) {          System.out.println(pom.getKsiążka().toString());    }

pom = ap.dodaj_książke(d5, tr2);
if (pom != null) {          System.out.println(pom.getKsiążka().toString());    }

}

```

# Tak może działać aplikacja po wykonaniu poszczególnych przypadków użycia

```
Wiersz polecenia
C:\Users\kruczkiewicz>java -jar "E:\Dydaktyka\Wzorce\Wypożyczalnia\dist\Wypożyczalnia.jar"
[Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1, Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2, Tytul: 3 Autor:3 ISBN: 3 Wydawnictwo:3, Tytul: 1 Autor:1 ISBN:
1 Wydawnictwo:1 Aktor:1, Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Aktor:2, Tytul: 4 Autor:4 ISBN: 4 Wydawnictwo:4 Aktor:4]
[Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Numer: 1]
[Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Numer: 1]
[Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Numer: 1]
[Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Numer: 1, Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Numer: 2]
[Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Aktor:1 Numer: 3 termin: Thu Apr 10 02:00:00 CEST 2008]
[Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Aktor:1 Numer: 3 termin: Thu Apr 10 02:00:00 CEST 2008]
[Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Aktor:1 Numer: 3 termin: Thu Apr 10 02:00:00 CEST 2008, Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Aktor:1
Numer: 2 termin: Thu Apr 10 02:00:00 CEST 2008]
[Tytul: 4 Autor:4 ISBN: 4 Wydawnictwo:4 Aktor:4 Numer: 2]
```

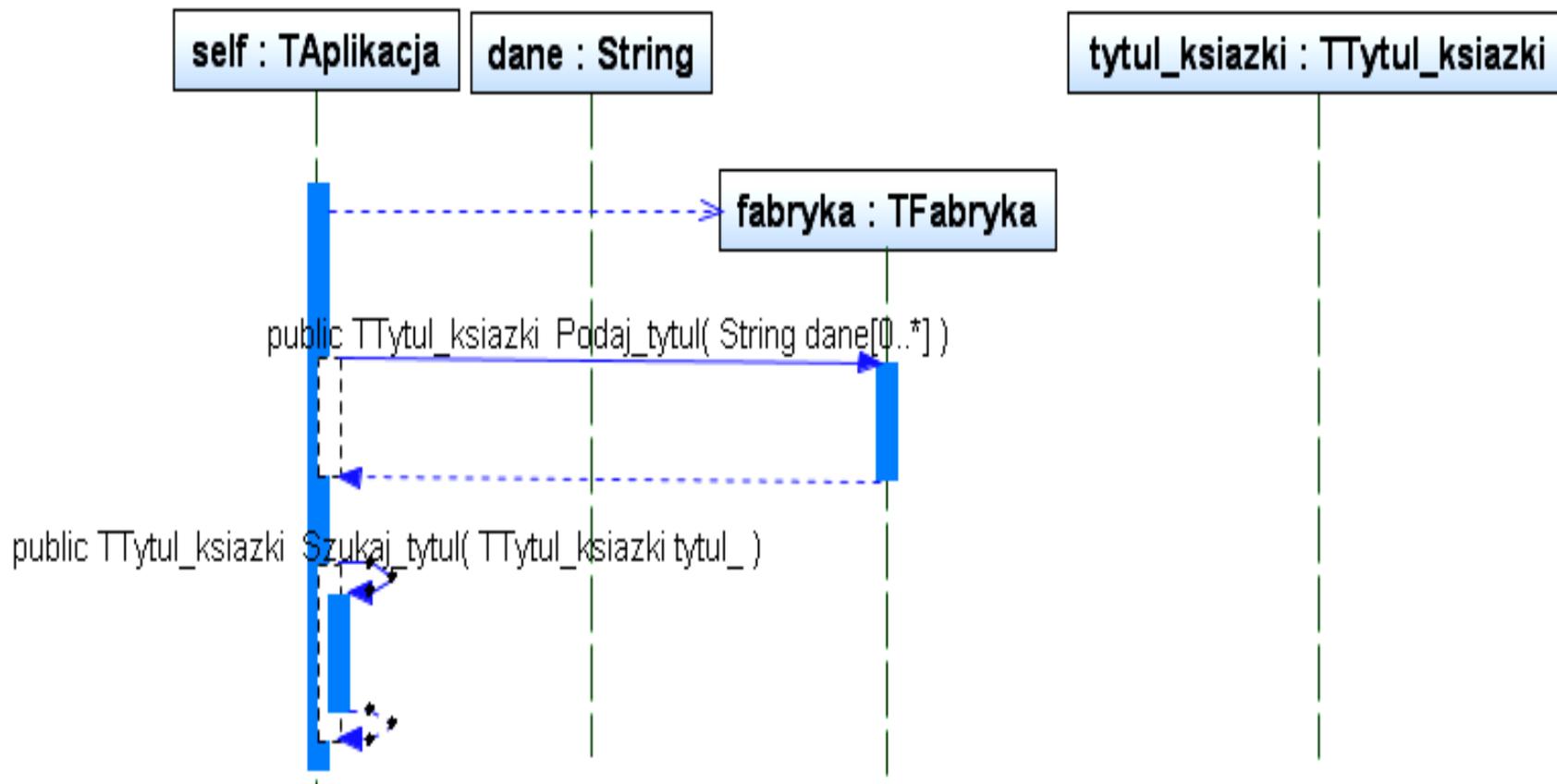
Projekt przypadku użycia  
„ **Wyszukiwanie tytułów** ”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia na podstawie diagramów sekwencji

## (5) Wyszukiwanie tytułow

**TTytul\_książki** **TAplikacja::Wyszukaj\_tytul** (String[] dane)



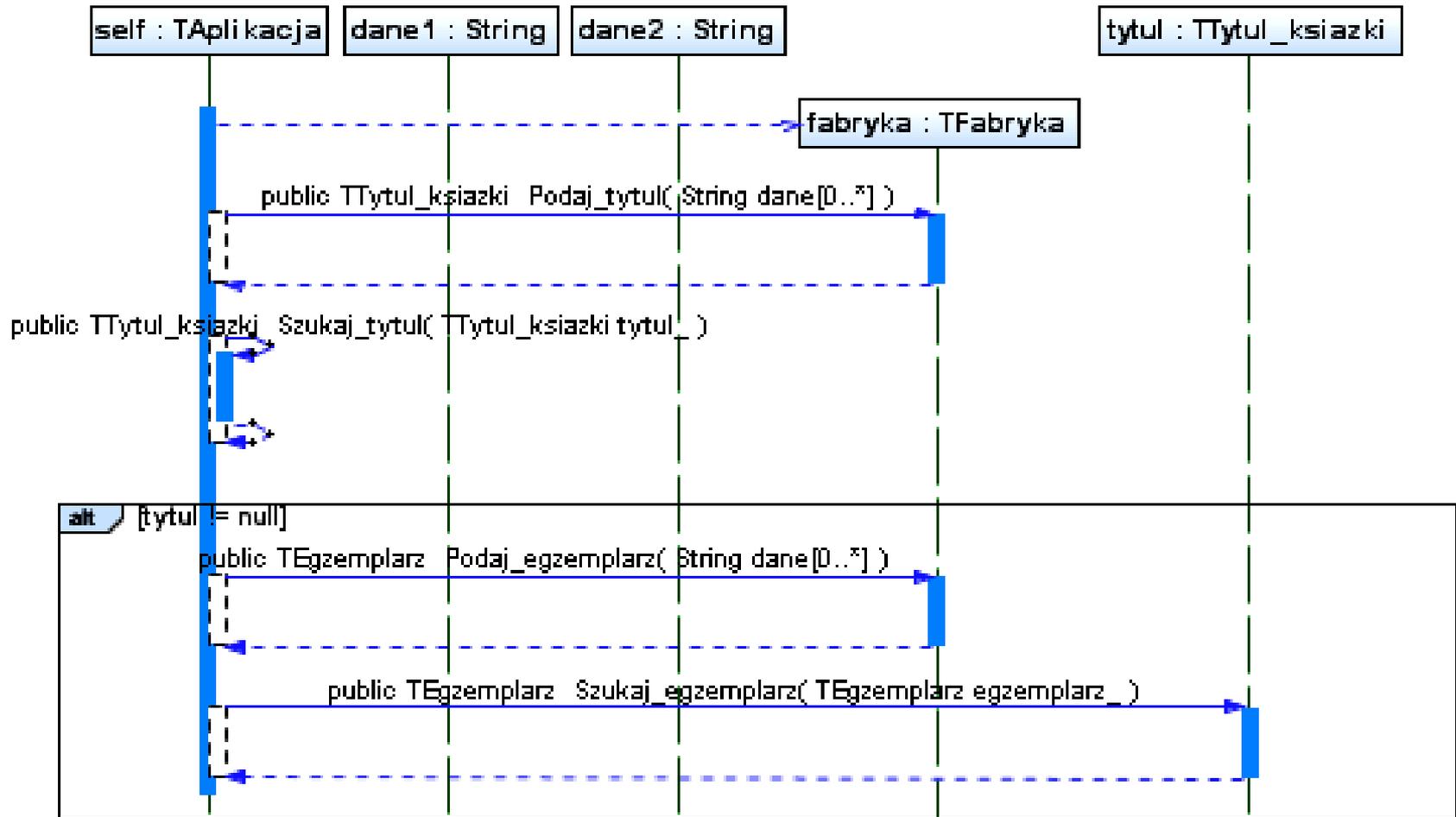
Projekt przypadku użycia  
„**Wyszukiwanie egzemplarzy**”

za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia na podstawie diagramów sekwencji

## (6) Wyszukiwanie egzemplarza

### TEgzemplarz TAplikacja::Wyszukaj\_egzemplarz (String[] dane1, String[] dane2)



```

package wypożyczalnia1;
import java.util.ArrayList;

public class TAplikacja {
    private ArrayList<TTytul_książki> mTytul_książki = new ArrayList<TTytul_książki>();

    public TAplikacja ()                                {/** */ }
    public static void main (String[] t)                {/** */ }
    private ArrayList<TTytul_książki> getTytul_książki () {/** */ }
    private void setTytul_książki (ArrayList<TTytul_książki> val) {/** */ }

    public TTytul_książki Szukaj_tytul (TTytul_książki tytul_) {/** */ }
    public void dodaj_tytul (String[] dane)              {/** */ }
    public TTytul_książki dodaj_książke (String[] dane1, String[] dane2) {/** */ }
    public TTytul_książki Wyszukaj_tytul (String[] dane) {/** */ }
    public TEgzemplarz Wyszukaj_egzemplarz (String[] dane1, String[] dane2) {/** */ }
}

```

```

package wypożyczalnia1;

public class TEgzemplarz {
    private int numer;
    private TTytul_książki mTytul_książki;

    public TEgzemplarz ()                                {/** */ }
    public int getNumer ()                                {/** */ }
    public void setNumer (int numer_)                    {/** */ }
    private TTytul_książki getTytul_książki ()          {/** */ }
    private void setTytul_książki (TTytul_książki val) {/** */ }
    public boolean equals (Object ob)                    {/** */ }
    public String toString ()                            {/** */ }
}

```

```
package wypożyczalnia1;
import java.util.Date;

public class TEgzemplarz_termin extends TEgzemplarz {
    private Date termin;

    public Date getTermin ()                {/** */      }
    public void setTermin (Date termin)     {/** */      }
    public boolean termin_minal (Date termin_) {/** */    }
    public String toString ()              {/** */      }
}
```

```
package wypożyczalnia1;

public class TFabryka {

    public TTytul_ksiazki Podaj_tytul (String[] dane)        {/** */      }
    public TEgzemplarz Podaj_egzemplarz (String[] dane)     {/** */      }
}
```

```

package wypożyczalnia1;
import java.util.ArrayList;

public class Ttytul_książki {
    private String wydawnictwo;
    private String ISBN;
    private String tytuł;
    private String autor;
    private ArrayList<TEgzemplarz> mKsiążka = new java.util.ArrayList<TEgzemplarz>();
    public Ttytul_książki ()                { /* */ }
    public String getWydawnictwo ()         { /* */ }
    public void setWydawnictwo (String e)   { /* */ }
    public String getTytuł ()               { /* */ }
    public void setTytuł (String a)         { /* */ }
    public String getISBN ()               { /* */ }
    public void setISBN (String ISBN_)     { /* */ }
    public String getAutor ()              { /* */ }
    public void setAutor (String autor_)   { /* */ }
    public String getAktor ()               { /* */ }
    public void setAktor (String aktor_)   { /* */ }
    public ArrayList<TEgzemplarz> getKsiążka () { /* */ }
    public void setKsiążka (ArrayList<TEgzemplarz> mKsiążka_) { /* */ }
    public String toString ()              { /* */ }
    public boolean equals (Object ob)      { /* */ }
    public void dodaj_książke (String[] dane) { /* */ }
    public TEgzemplarz Szukaj_egzemplarz (TEgzemplarz egzemplarz_) { /* */ }
}

```

```

package wypożyczalnia1;
public class Ttytul_książki_na_kasie extends Ttytul_książki {
    private String aktor;
    public String getAktor ()                { /* */ }
    public void setAktor (String aktor)     { /* */ }
    public String toString ()               { /* */ }
}

```

# Proponowany kod funkcji main w klasie fasadowej TAplikacja

```
public static void main(String t[]) // your code here
{TAplikacja ap = new TAplikacja();
    String t1[] = {"1", "1", "1", "1", "1"}; //t1, t2, t3 – tablice łańcuchów do tworzenia tytułu książki zwykłej – pierwszy łańcuch
    String t2[] = {"1", "2", "2", "2", "2"}; // jest informacją dla fabryki, jaki obiekt wygenerować
    String t3[] = {"1", "3", "3", "3", "3"}; //”1” oznacza utworzenie obiektu klasy TTytul_ksiazki, a pozostałe łańcuchy to kolejno
// autor, tytul, ISBN, wydawnictwo dla uproszczenia w postaci cyfr - obiekty do wstawiania
    String t4[] = {"3", "1", "1", "1", "1", "1"}; // t4, t5,t6 – tablice łańcuchów do tworzenia tytułu książki jako nagranie dźwiękowe
    String t5[] = {"3", "2", "2", "2", "2", "2"}; //– pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować
    String t6[] = {"3", "4", "4", "4", "4", "4"}; //”3” oznacza utworzenie obiektu klasy TTytul_ksiazki_na_kasecie, a pozostałe
//łańcuchy to kolejno autor, tytul, ISBN, wydawnictwo, aktor dla uproszczenia w postaci cyfr- obiekty do wstawiania
    ap.dodaj_tytul(t1);
    ap.dodaj_tytul(t2);      ap.dodaj_tytul(t2);
    ap.dodaj_tytul(t3);
    ap.dodaj_tytul(t4);
    ap.dodaj_tytul(t5);      ap.dodaj_tytul(t5);
    ap.dodaj_tytul(t6);
    String lan = ap.getTytul_ksiazki().toString();      System.out.println(lan);
    String d1[] = {"0", "1"}; // d1, d2, d3 - – tablice łańcuchów do tworzenia wzorcowego tytułu książki zwykłej do wyszukiwania
    String d2[] = {"0", "2"}; // – pierwszy łańcuch jest informacją dla fabryki, jaki obiekt wygenerować: „0” oznacza generowanie
    String d3[] = {"0", "5"}; // obiektu klasy TTytul_ksiazki, drugi łańcuch jest ISBN – obiekty do wyszukiwania
    String d4[] = {"2", "1", "1"}; //d4, d5 - tablice łańcuchów do tworzenia wzorcowego tytułu książki jako nagranie dźwiękowe
    String d5[] = {"2", "4", "4"}; //pierwszy łańcuch „2” oznacza generowanie obiektu typu TTytul_ksiazki_na_kasecie
// drugi łańcuch to ISBN, trzeci jest nazwiskiem aktora-objekty do wyszukiwania
    String tr1[] = {"0", "1"}; //tablice tr1 i tr2 zawierają informację o tworzeniu obiektu typu TEgzemplarz: pierwszy łańcuch
    String tr2[] = {"0", "2"}; //równy „0” oznacza tworzenie obiektu typu typu TEgzemplarz, drugi jest numerem egzemplarza
    String tr3[] = {"1", "3", "April 10, 2008, 00:00:00 GMT"}; //pierwszy łańcuch równy „1” oznacza tworzenie obiektu klasy
    String tr4[] = {"1", "2", "April 10, 2008, 00:00:00 GMT"}; //TEgzemplarz_termin, drugi oznacza numer, trzeci termin
```

**// W trakcie tworzenia kodu aplikacji można odsłaniać kod z  
// komentarza w celu przetestowania kolejnych przypadków użycia**

```
TTytul_książki pom = ap.dodaj_książke(d1, tr1);  
if (pom != null) {      System.out.println(pom.getKsiążka().toString());      }  
    pom = ap.dodaj_książke(d2, tr1);  
if (pom != null) {      System.out.println(pom.getKsiążka().toString());      }  
    pom = ap.dodaj_książke(d2, tr1);  
if (pom != null) {      System.out.println(pom.getKsiążka().toString());      }  
    pom = ap.dodaj_książke(d2, tr2);  
if (pom != null) {      System.out.println(pom.getKsiążka().toString());      }  
    pom = ap.dodaj_książke(d3, tr2);  
if (pom != null) {      System.out.println(pom.getKsiążka().toString());      }  
    pom = ap.dodaj_książke(d4, tr3);  
if (pom != null) {      System.out.println(pom.getKsiążka().toString());      }  
    pom = ap.dodaj_książke(d4, tr3);  
if (pom != null) {      System.out.println(pom.getKsiążka().toString());      }  
    pom = ap.dodaj_książke(d4, tr4);  
if (pom != null) {      System.out.println(pom.getKsiążka().toString());      }  
    pom = ap.dodaj_książke(d5, tr2);  
if (pom != null) {      System.out.println(pom.getKsiążka().toString());      }  
  
System.out.println("Wyszukiwanie");  
System.out.println(ap.Wyszukaj_tytul(t5).toString());  
System.out.println(ap.Wyszukaj_egzemplarz(d4, tr4).toString()); */  
}
```

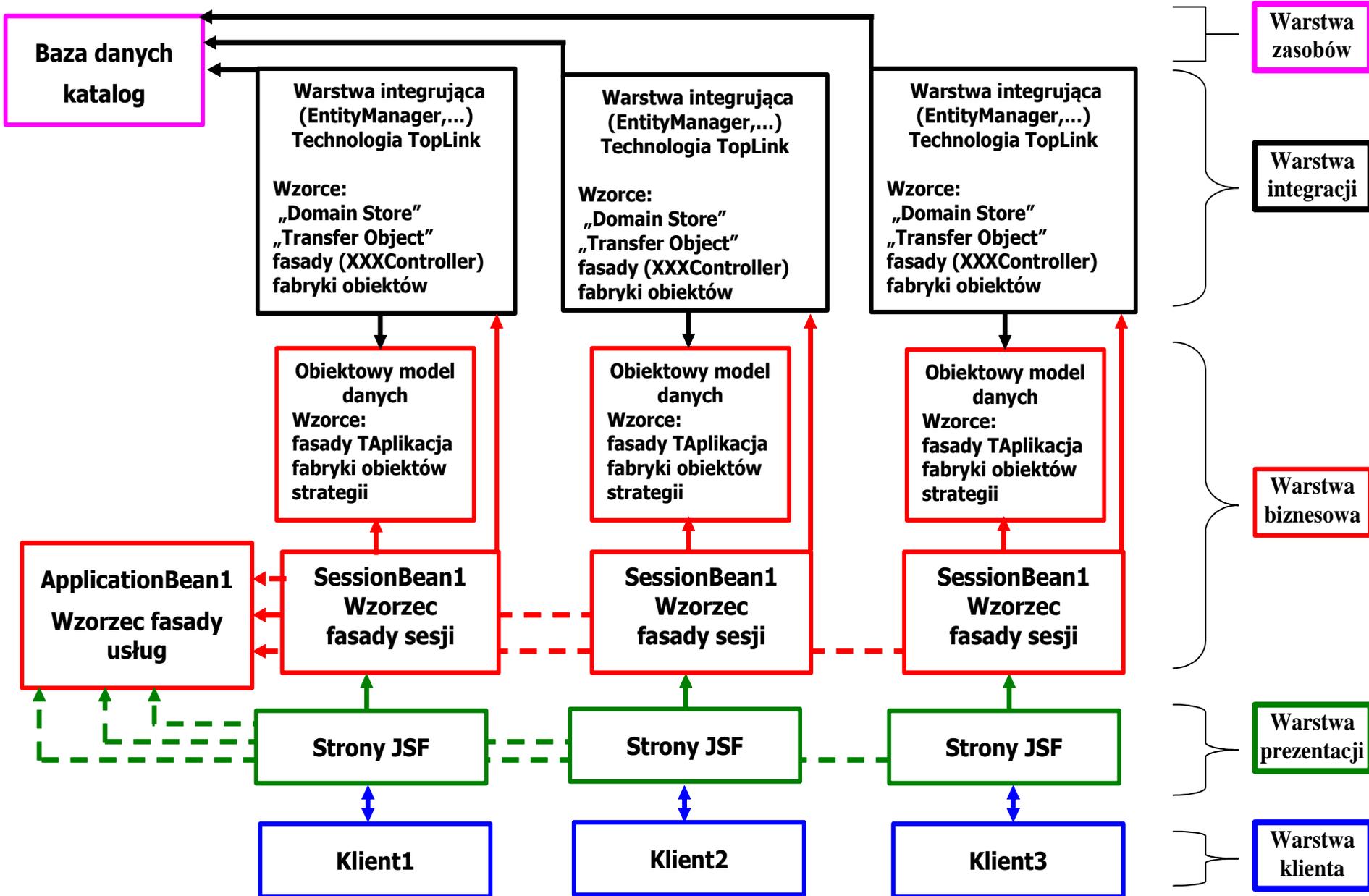
# Tak może działać aplikacja po wykonaniu poszczególnych przypadków użycia

```
Wiersz polecenia
C:\Users\kruczkiewicz>java -jar "E:\Dydaktyka\Wzorce\Wypozyczalnia\dist\Wypozyczalnia1.jar"
[Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1, Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2, Tytul: 3 Autor:3 ISBN: 3 Wydawnictwo:3, Tytul: 1 Autor:1 ISBN:
1 Wydawnictwo:1 Aktor:1, Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Aktor:2, Tytul: 4 Autor:4 ISBN: 4 Wydawnictwo:4 Aktor:4]
[Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Numer: 1]
[Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Numer: 1]
[Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Numer: 1]
[Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Numer: 1, Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Numer: 2]
[Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Aktor:1 Numer: 3 termin: Thu Apr 10 02:00:00 CEST 2008]
[Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Aktor:1 Numer: 3 termin: Thu Apr 10 02:00:00 CEST 2008]
[Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Aktor:1 Numer: 3 termin: Thu Apr 10 02:00:00 CEST 2008, Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Aktor:1
Numer: 2 termin: Thu Apr 10 02:00:00 CEST 2008]
[Tytul: 4 Autor:4 ISBN: 4 Wydawnictwo:4 Aktor:4 Numer: 2]
Wyszukiwanie
Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Aktor:2
Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Aktor:1 Numer: 2 termin: Thu Apr 10 02:00:00 CEST 2008
```

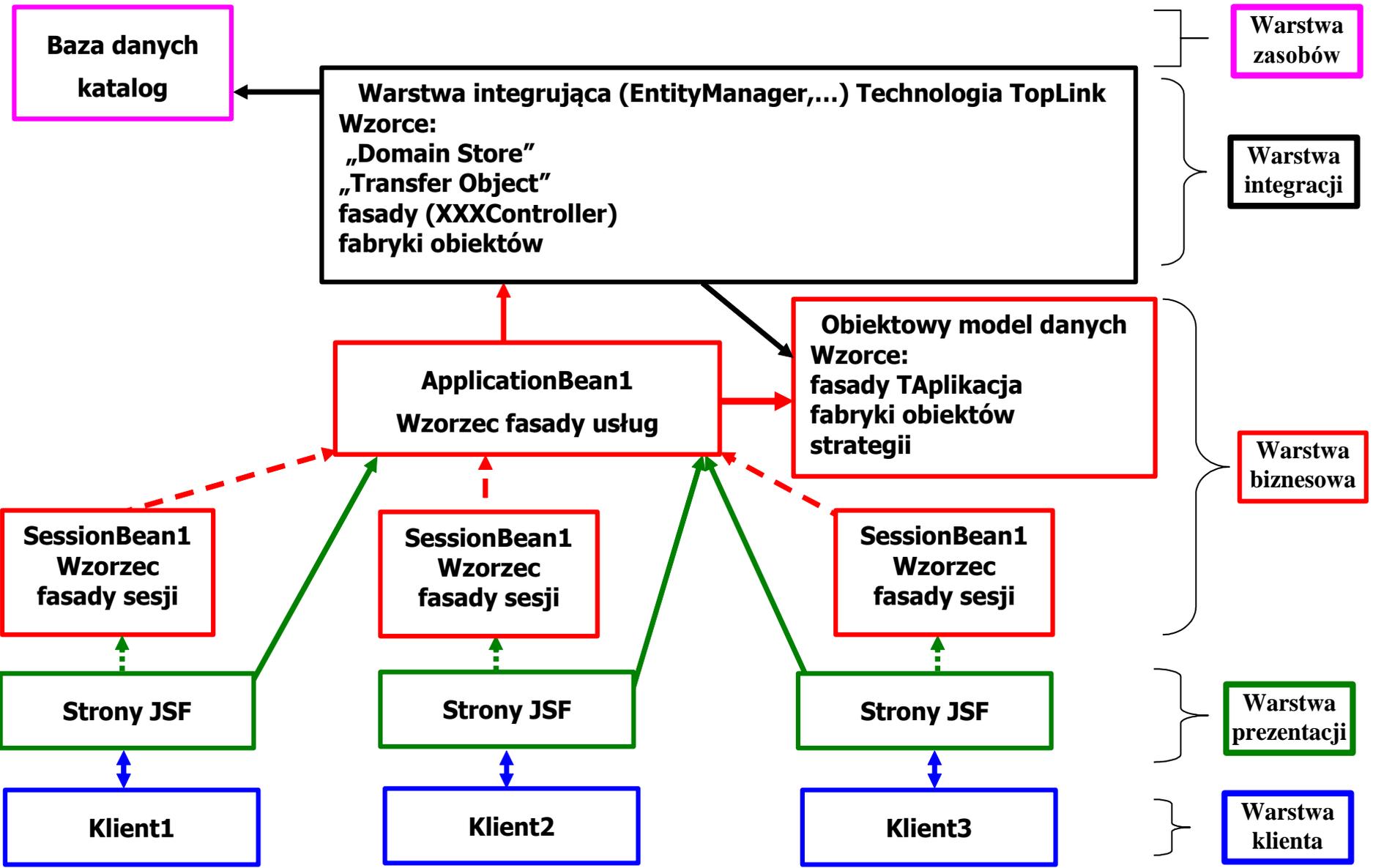
# **5. Implementacja wielowarstwowego systemu informatycznego**

Przykłady architektury wielowarstwowej w środowisku Visual Web Java Server Faces

# Architektura aplikacji pięciowarstwowej – Java EE 5.0 Visual Web Java Server Faces (linie przerywane oznaczają powiązania niewykorzystane w aplikacji)



# Architektura aplikacji pięciowarstwowej Java EE 5.0 Visual Web Java Server Faces - linie przerywane oznaczają powiązania niewykorzystane w aplikacji



## **6. Tworzenie bazy danych w systemie baz danych Derby**

- przykład **warstwy zasobów** (EIS)

# Zakładanie pustej bazy danych dla systemu baz danych Derby (1)

NetBeans IDE 6.5

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

<default config>

Services

Databases

- Java DB
- Driver
- jdbc:...
- jdbc:...
- jdbc:...
- jdbc:...
- jdbc:derby://localhost:1527/travel [
- jdbc:derby://localhost:1527/vir [vir

Web Services

Enterprise Beans (2.x)

Servers

Start Page Main.java Klient3.java serwer3.java

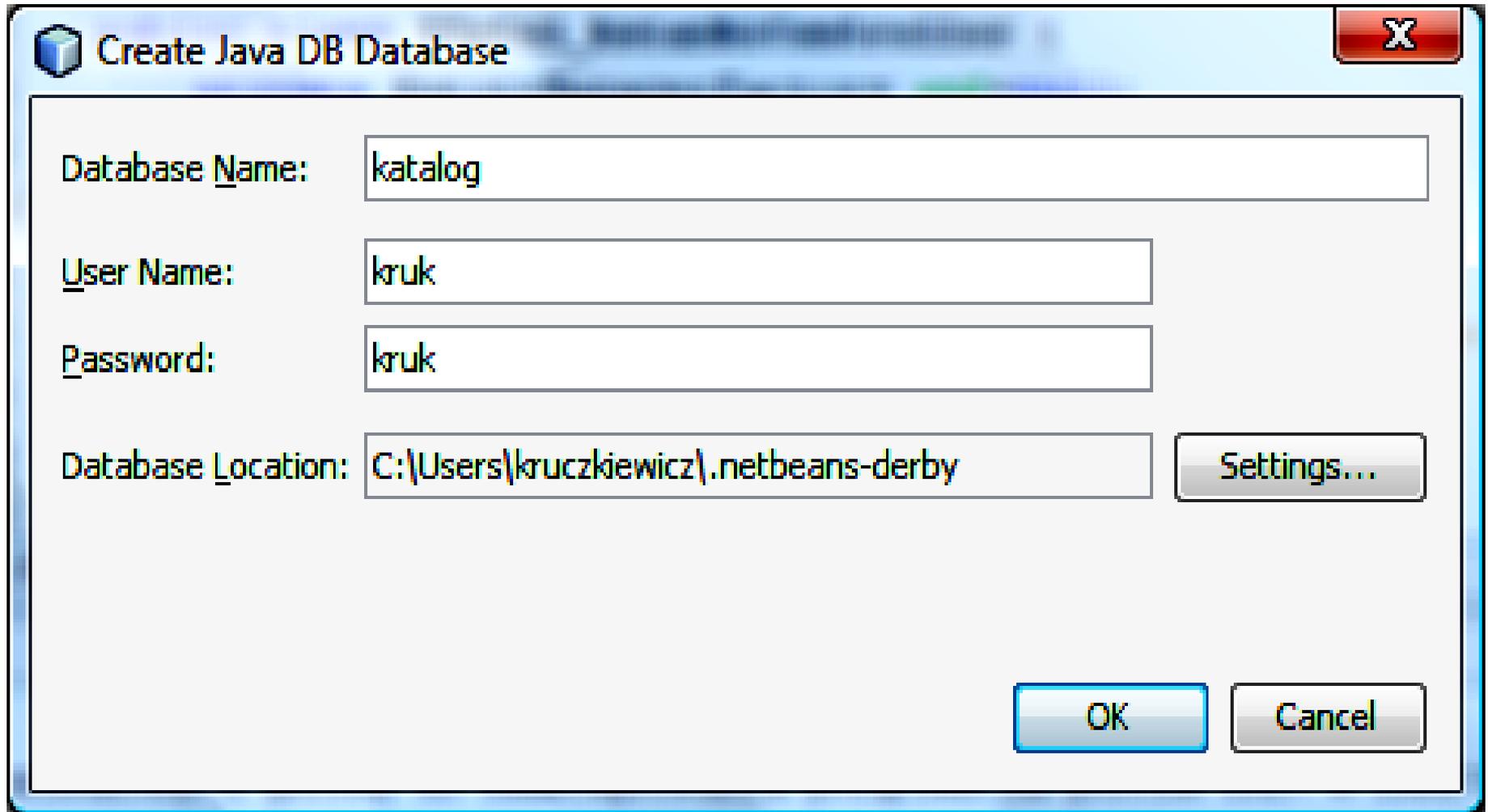
```
/*
 * To change this template, choose Tools
 * and open the template in the editor.
 */

package protsym;

import java.util.Random;

/**
 *
 * @author kruczkiewicz
 */
public class Main (
    /**
     * @param args the command line argu
```

Zakładanie pustej bazy danych **katalog** w systemie baz danych Derby (2)



**Create Java DB Database**

Database Name: katalog

User Name: kruk

Password: kruk

Database Location: C:\Users\kruczkiewicz\.netbeans-derby

Settings...

OK Cancel

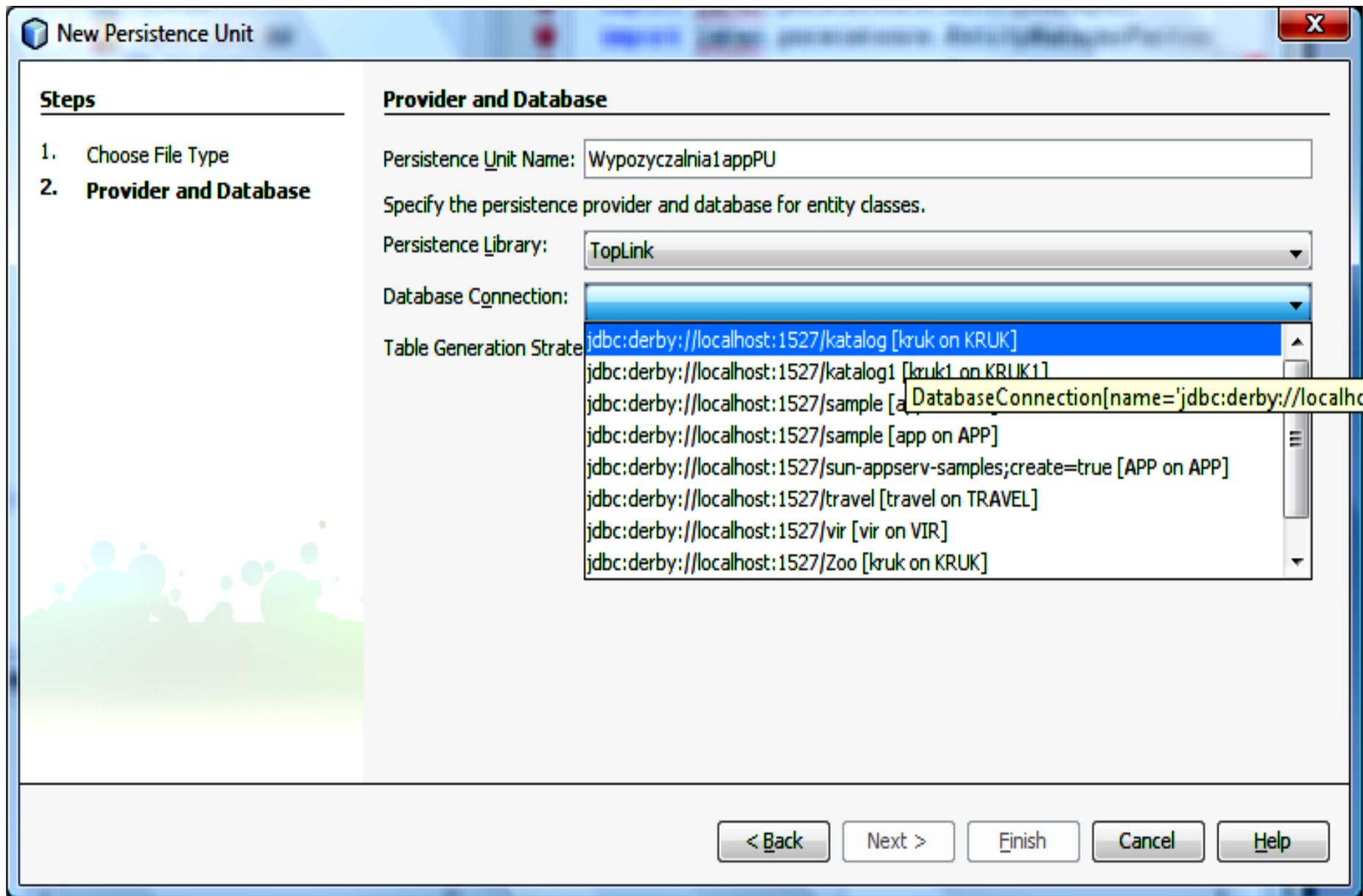
**7. Tworzenie **warstwy integracji** w  
projekcie Java Application.  
Zastosowanie wzorców projektowych  
typu **Domain Store** i **Transfer Object**.**

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

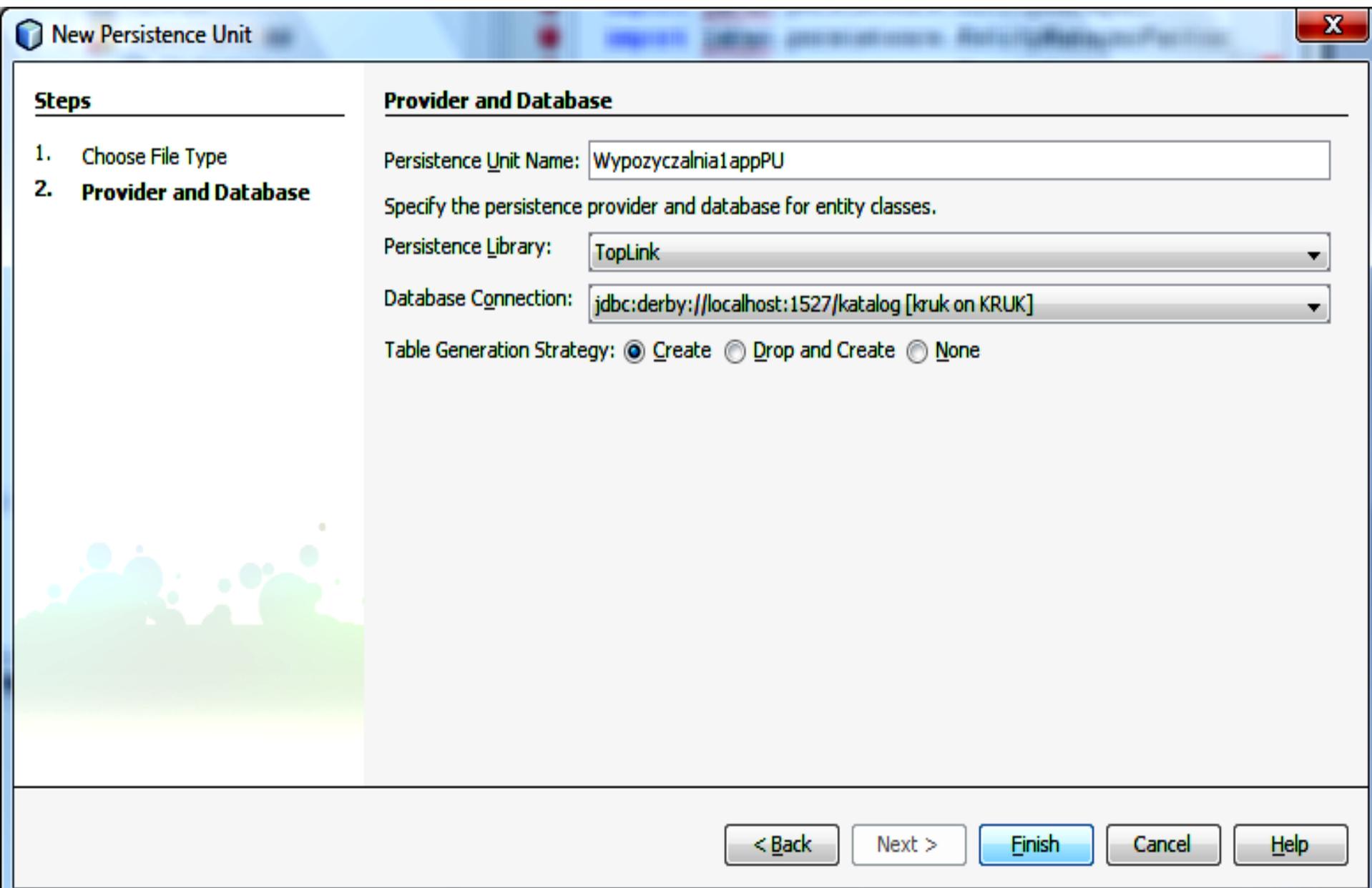
The screenshot shows an IDE window with the 'New' menu open. The 'Persistence Unit...' option is highlighted. The code editor displays the following Java code:

```
java.util.List;  
javax.persistence.EntityManager;  
javax.persistence.EntityManagerFactory;  
javax.persistence.Persistence;  
  
hor kruczkiewicz  
  
class Ttytul_książkiController {  
    private EntityManagerFactory emf=null;  
  
    private EntityManager getEntityManager() {  
        if (emf == null) {  
            emf = Persistence.createEntityManagerFactory("WypożyczalniaIappPU");  
        }  
        return emf.createEntityManager();  
    }  
  
    public Ttytul_książki[] getTtytul_książkis() {  
        return (Ttytul_książki[])getTtytul_książki().toArray(new Ttytul_książki[0]);  
    }  
  
    public List<Ttytul_książki> getTtytul_książki() {  
        EntityManager em = getEntityManager();  
        try {  
            javax.persistence.Query q =
```

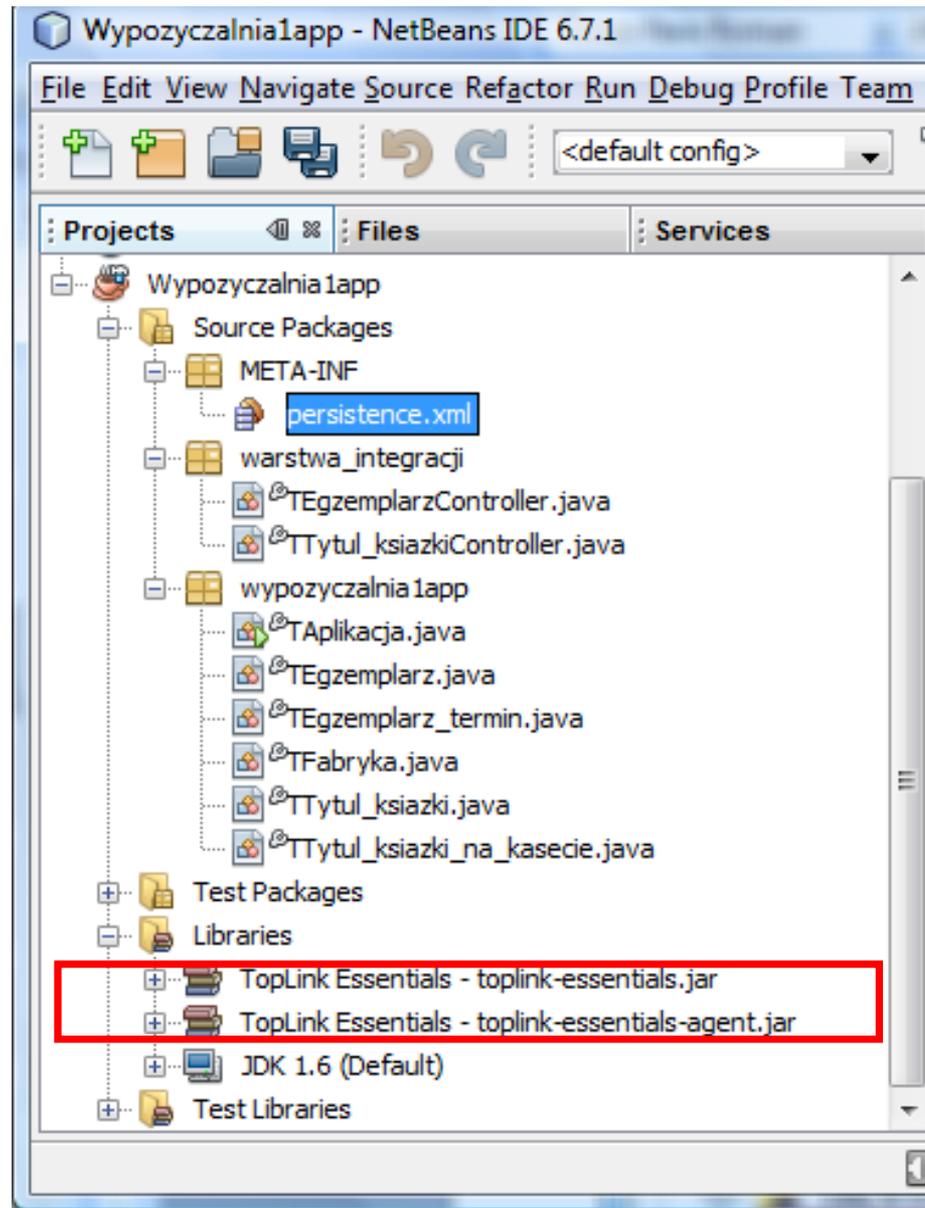
# Wybór bazy danych, w której będą utrwalane obiekty – pustej (2)



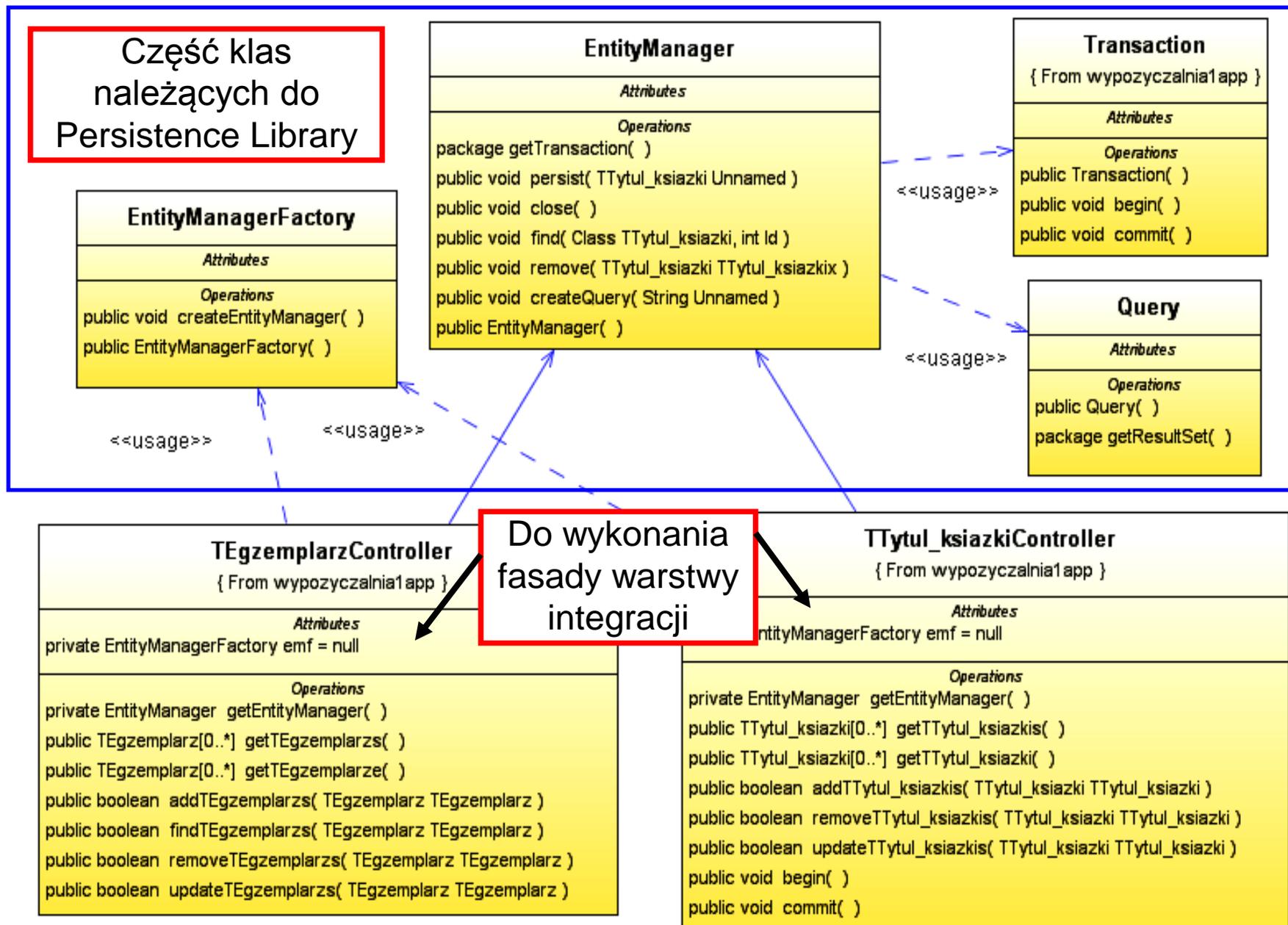
# Tworzenie modułu utrwalania danych dla technologii TopLink (3)



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)



# Diagram klas – uproszczony schemat warstwy integracji (5)



# Przykład klasy warstwy integracji – dla każdej klasy typu **Entity** (tutaj dla **TTytul\_książki**) (6)

The screenshot displays the NetBeans IDE interface for a project named "Wypożyczalnia1app". The left sidebar shows the project structure with the "warstwa\_integracji" package selected, containing files like "TEgzemplarzController.java" and "TTYtul\_książkiController.java". The main editor window shows the source code for "TTYtul\_książkiController.java".

```
package warstwa_integracji;

import ...

/**...*/
public class TTYtul_książkiController {

    private EntityManagerFactory emf = null;

    private EntityManager getEntityManager() {...}

    public TTYtul_książki[] getTTYtul_książkis() {...}

    public List<TTYtul_książki> getTTYtul_książki() {...}

    public boolean addTTYtul_książkis(TTYtul_książki TTYtul_książki) {...}

    public boolean addTTYtul_książki(ArrayList<TTYtul_książki> tytuly) {...}

    public boolean removeTTYtul_książkis(TTYtul_książki TTYtul_książki) {...}

    public boolean updateTTYtul_książkis(TTYtul_książki TTYtul_książki) {...}
}
```

The bottom status bar shows the page number "105 | 1" and the text "INS".

**Klasa typu Controller dla każdej z klas utrwalanych obiektów – (7)**  
**Realizacja wzorców typu Domain Store i Transfer Object**  
**Podstawowe metody klasy, często generowane automatycznie przez środowisko projektowe**

```
public class TTytul_ksiazkiController  
{
```

```
private EntityManagerFactory emf=null;
```

```
private EntityManager getEntityManager() {  
  if (emf == null) {  
    emf = Persistence.createEntityManagerFactory("Wypożyczalnia1appPU");  
  }  
  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; }
}
```

Itd....

**Dodanie adnotacji do klas typu dane („Entity”)  
wspierających mapowanie obiektowo-relacyjne  
(8)**

Zmiana typu klas danych na typ @Entity (10) – dodano adnotacje, nowe atrybuty (Id, Ksiazka) z metodami. Należy zestandaryzować nazwy metod dostępu do składowych mKsiazka oraz Ksiazka !

```
13  @Entity
14  public class TTytul_ksiazki implements Serializable {
15      private static final long serialVersionUID = 1L;
16      private String wydawnictwo;
17      private String ISBN;
18      private String tytul;
19      private String autor;
20
21      @Id
22      @GeneratedValue(strategy = GenerationType.AUTO)
23      private Long id;
24      public Long getId()
25      { return id; }
26      public void setId(Long id)
27      { this.id = id; }
28      @OneToMany(mappedBy = "mTytul_ksiazki")
29      private Collection<TEgzemplarz> Ksiazka;
30      public Collection<TEgzemplarz> getKsiazka()
31      { return Ksiazka; }
32      public void setKsiazka(Collection<TEgzemplarz> Ksiazka)
33      { this.Ksiazka = Ksiazka; }
34      public TTytul_ksiazki()
35      { id = null; }
36
37      @Transient
38      private ArrayList<TEgzemplarz> mKsiazka =
39      new java.util.ArrayList<TEgzemplarz>();
40      public ArrayList<TEgzemplarz> getMKsiazka()
41      { return mKsiazka; }
42      public void setMKsiazka(ArrayList<TEgzemplarz> mKsiazka)
43      { this.mKsiazka = mKsiazka; }
```

```
11  @Entity
12  public class TEgzemplarz implements Serializable {
13      private static final long serialVersionUID = 1L;
14      private int numer;
15
16      @Id
17      @GeneratedValue(strategy = GenerationType.AUTO)
18      private Long id;
19      public void setId(Long id) {
20          this.id = id;
21      }
22      public Long getId() {
23          return id;
24      }
25      @ManyToOne
26      private Ttytul_książki mTytul_książki;
27      public Ttytul_książki getMTytul_książki() {
28          return mTytul_książki;
29      }
30      public void setMTytul_książki(Ttytul_książki mTytul_książki) {
31          this.mTytul_książki = mTytul_książki;
32      }
33      public TEgzemplarz() {
34          id = null;
35      }
36  }
```

# Wstawienie do modułu typu **Persistence Unit** wybrane klasy typu **Entity** (9)

The screenshot shows the NetBeans IDE interface for configuring a Persistence Unit. The left sidebar displays the project structure for 'Wypożyczalnia1app', with 'persistence.xml' selected under 'META-INF'. The main window shows the 'Wypożyczalnia1appPU' configuration dialog with the following settings:

- Persistence Unit Name: Wypożyczalnia1appPU
- Persistence Library: TopLink
- JDBC Connection: jdbc:derby://localhost:1527/katalog [krak on KRUK]
- Table Generation Strategy:  Create  Drop and Create  None
- Include All Entity Classes in "Wypożyczalnia1app" Module

The 'Include Entity Classes' section is highlighted with a red box and contains the following list of classes:

- wypożyczalnia1app.TTytul\_książki
- wypożyczalnia1app.TEgzemplarz
- wypożyczalnia1app.TEgzemplarz\_termin
- wypożyczalnia1app.TTytul\_książki\_na\_kascecie

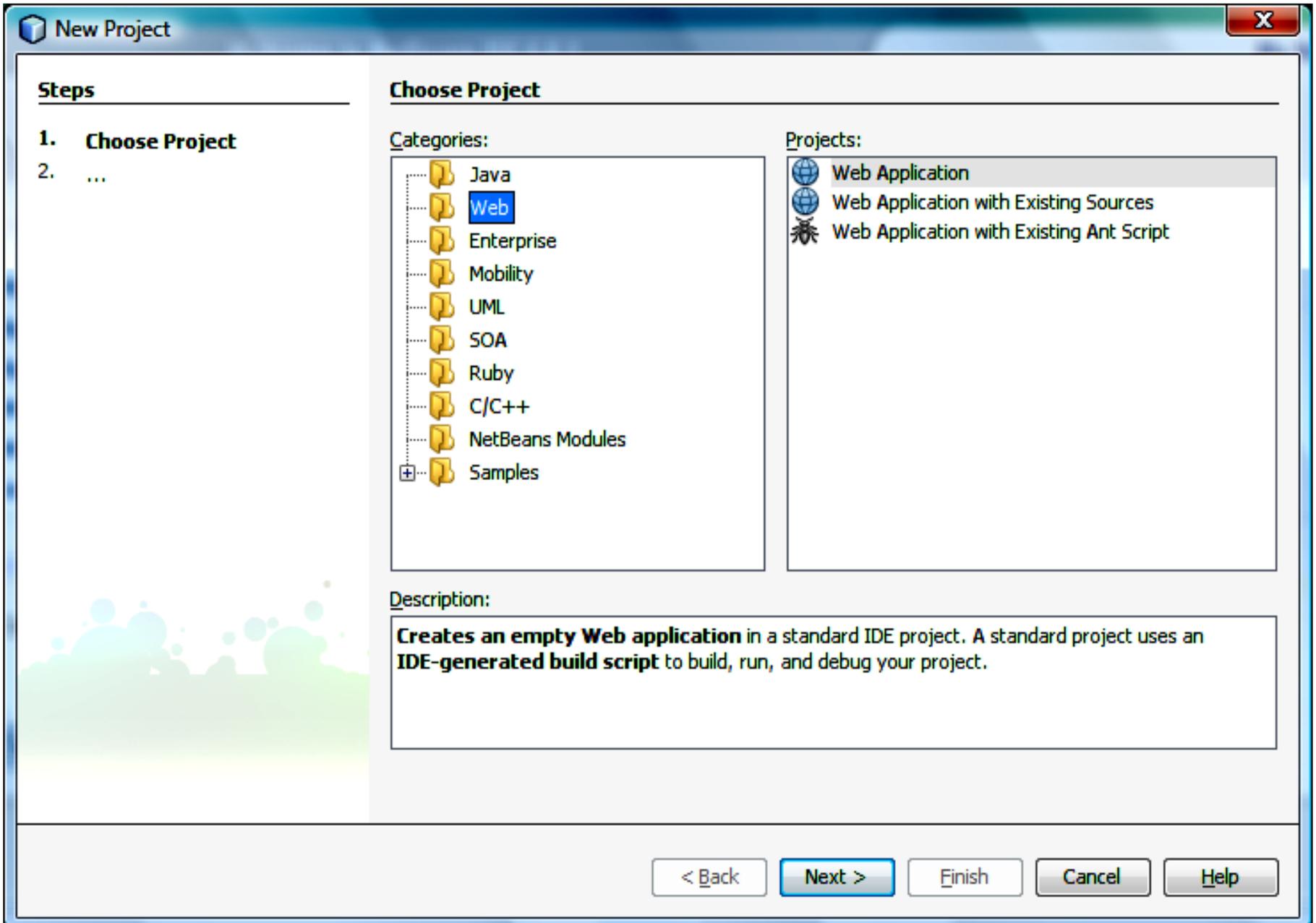
Buttons for 'Add Class...' and 'Remove' are visible to the right of the list.

# **8. Tworzenie warstwy prezentacji**

## **Pierwszy etap – tworzenie stron typu JSP**

**Wykonanie formularzy typu JSP  
zawierających  
wieloużywalne formularze typu JSPF dla  
aplikacji przeznaczonej dla wielu klientów  
ze wspólną warstwą biznesową istniejącą  
podczas działania aplikacji i wspólną  
warstwą integrującą z bazą danych.**

# 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)

**New Web Application**

**Steps**

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

**Name and Location**

Project Name:

Project Location:

Project Folder:

**Add to Enterprise Application:**

**Server:**

**Java EE Version:**

**Context Path:**

Set as Main Project

< Back   Next >   Finish   Cancel   Help

Projekt może być oparty np. na Visual Web JavaServer Faces (3) – przykład metody tworzenia warstwy prezentacji metoda „przeciągnij i upuść”

**New Web Application**

**Steps**

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

**Frameworks**

Select the frameworks you want to use in your web application.

- Visual Web JavaServer Faces
- JavaServer Faces
- Struts 1.2.9

Visual Web JavaServer Faces Configuration

Default Java Package:

JSF Servlet Name:

Servlet URL Mapping:

Validate XML     Verify Objects

< Back    Next >    **Finish**    Cancel    Help



Proj... Files Services

**WebWypożyczalnia1**

- Web Pages
  - WEB-INF
  - resources
  - Page1.jsp
- Configuration Files
- Server Resources
- Source Packages
  - webwypożyczalnia1
    - ApplicationBean1.java
    - Bundle.properties
    - Page1.java
    - RequestBean1.java
    - SessionBean1.java
- Test Packages
- Libraries
- Test Libraries
- Project Woodstock Themes
- Component Libraries
- Data Source References
- Wypożyczalnia1app
  - Source Packages
    - META-INF
      - persistence.xml
    - warstwa\_integracji
      - TEgzemplarzController.java
      - TTytul\_ksiazkiController.java
    - wypożyczalnia1app
      - TAplikacja.java
      - TEgzemplarz.java
      - TEgzemplarz\_termin.java
      - TFabryka.java
      - TTytul\_ksiazki.java
      - TTytul\_ksiazki\_na\_kasecie.java

...ava Page1

Design JSP Java Any Siz

gn your page by dragging components from the Pale

a component's behavior by adding code: Either doub

ment or right-click the component and choose its eve

ge layout is currently set to Grid Layout, which positio

ponents at specific x and y coordinates. If you wou

e flow layout, which positions components left to righ

bottom, change the Page Layout property to Flow L

Palette

**Woodstock Basic**

- Label
- Static Text
- Text Field
- Text Area
- Button
- Hyperlink
- Image Hyperlink
- Drop Down List
- Listbox
- Checkbox
- Checkbox Group
- Radio Button

No Properties

<No Properties>

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

The screenshot displays the NetBeans IDE 6.0.1 interface. The main window is titled "WebWypożyczalnia1 - NetBeans IDE 6.0.1". The menu bar includes "File", "Edit", "View", "Navigate", "Source", "Refactor", "Build", "Run", "Profile", "Versioning", "Tools", "Window", and "Help". The toolbar contains various icons for file operations and development actions.

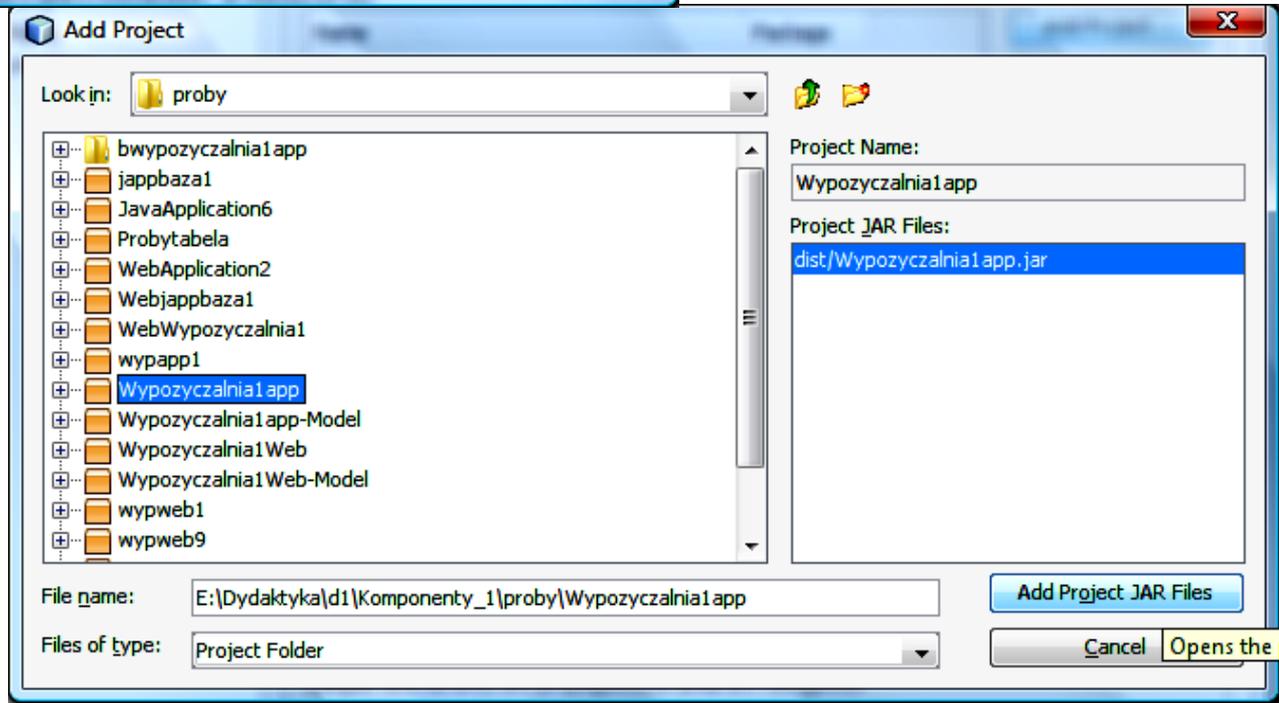
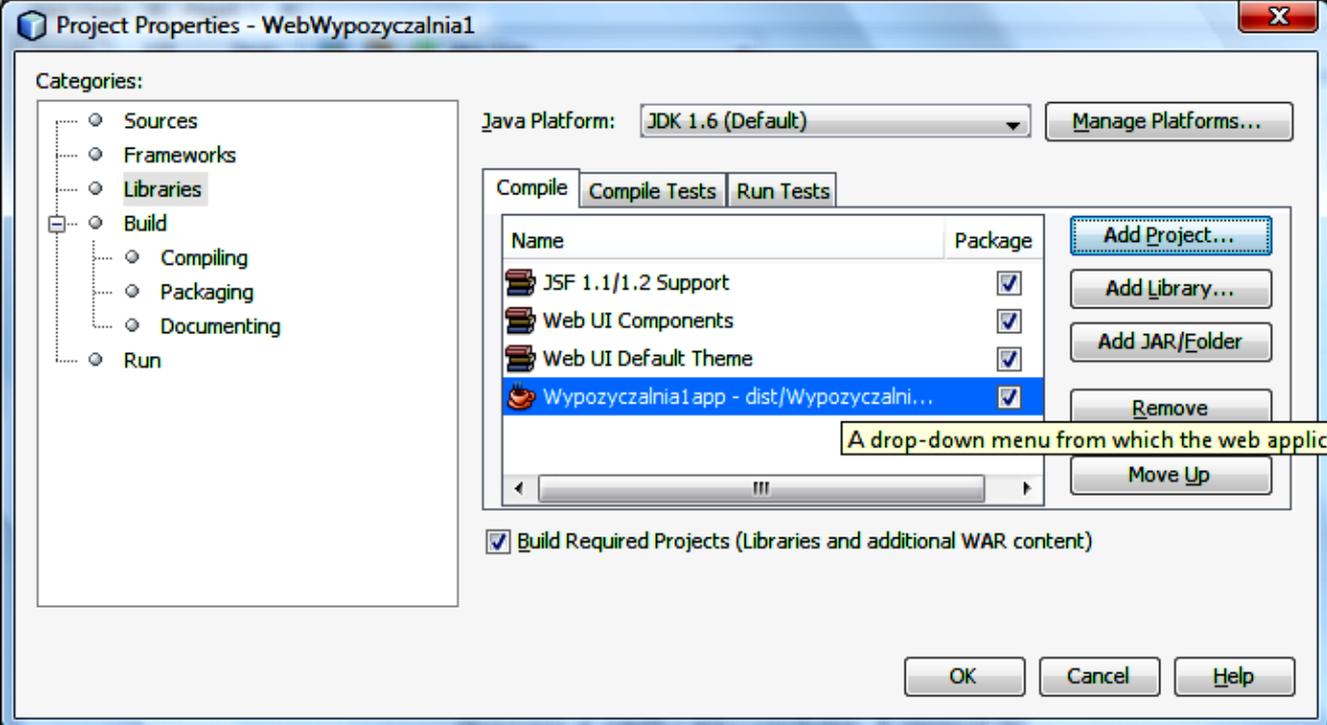
The "Project" view on the left shows a tree structure for "WebWypożyczalnia1". A context menu is open over the project, listing actions such as "New", "Build", "Clean and Build", "Clean", "Verify", "Generate Javadoc", "Run", "Undeploy and Deploy", "Debug", "Profile", "Set as Main Project", "Open Required Projects", "Close", "Rename...", "Move...", "Copy...", "Delete", "Find...", "Reverse Engineer...", "Versioning", "Local History", and "Properties". The "Properties" option is highlighted, and a tooltip below it reads "WebWypożyczalnia1 Properties".

The main editor area is titled "Page1" and is in "Design" mode. It features a grid background and contains the following text:

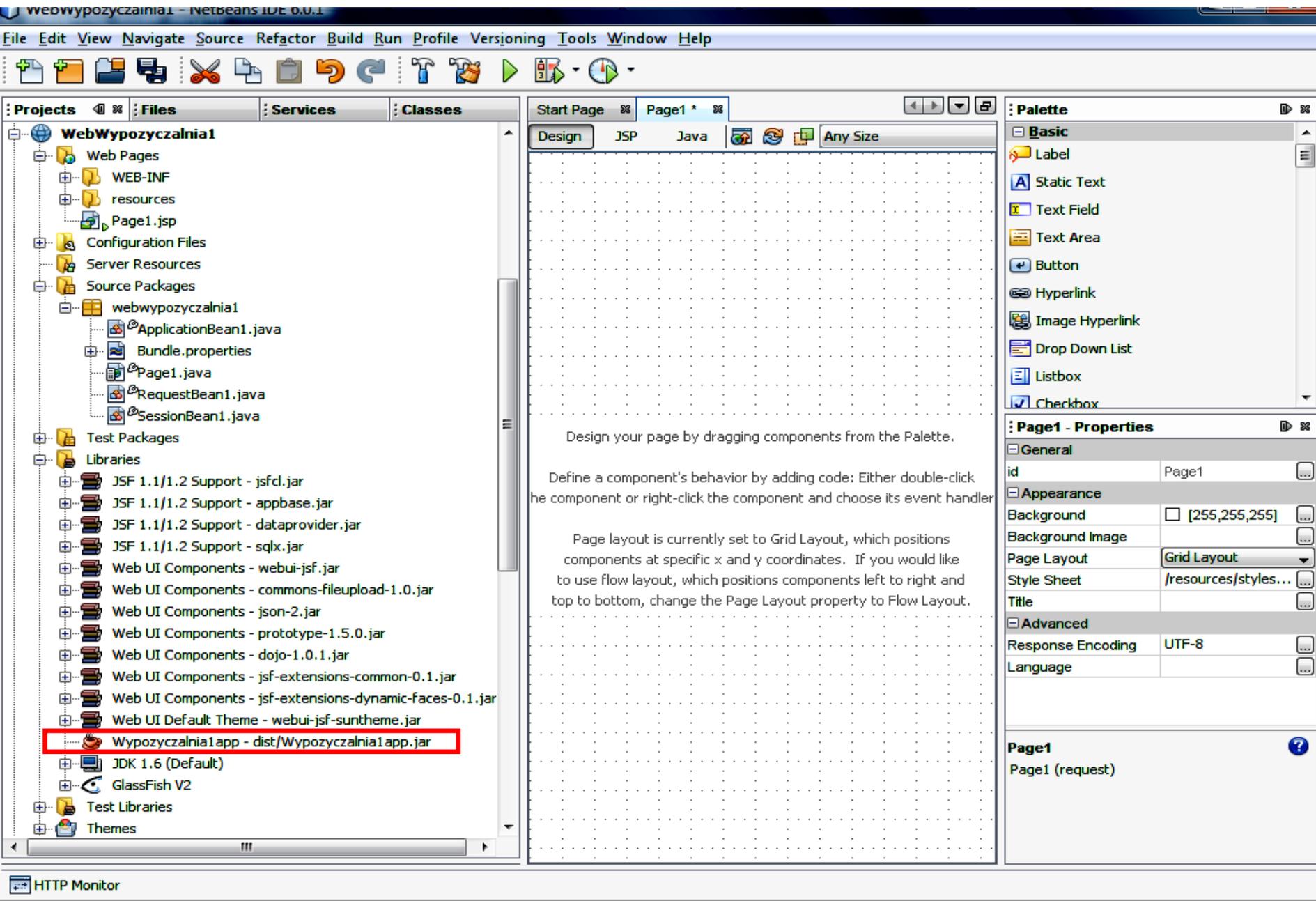
- Design your page by dragging components from the Palette.
- Define a component's behavior by adding code: Either double-click the component or right-click the component and choose its event handler.
- Page layout is currently set to Grid Layout, which positions components at specific x and y coordinates. If you would like to use flow layout, which positions components left to right and top to bottom, change the Page Layout property to Flow Layout.

The "Palette" on the right side of the editor shows a list of basic components: Label, Static Text, Text Field, Text Area, Button, Hyperlink, Image Hyperlink, Drop Down List, Listbox, and Checkbox. Below the palette, the "Properties" view for the selected component shows "<No Properties>".

At the bottom of the IDE, the "HTTP Monitor" tab is visible. The Windows taskbar at the very bottom shows the system tray with the time 16:39 and several open applications including ".netbeans-derby", "pkw3", and "Microsoft PowerP...".



# Połączone projekty–projekt Web Application korzysta z klas projektu Java Application (5)



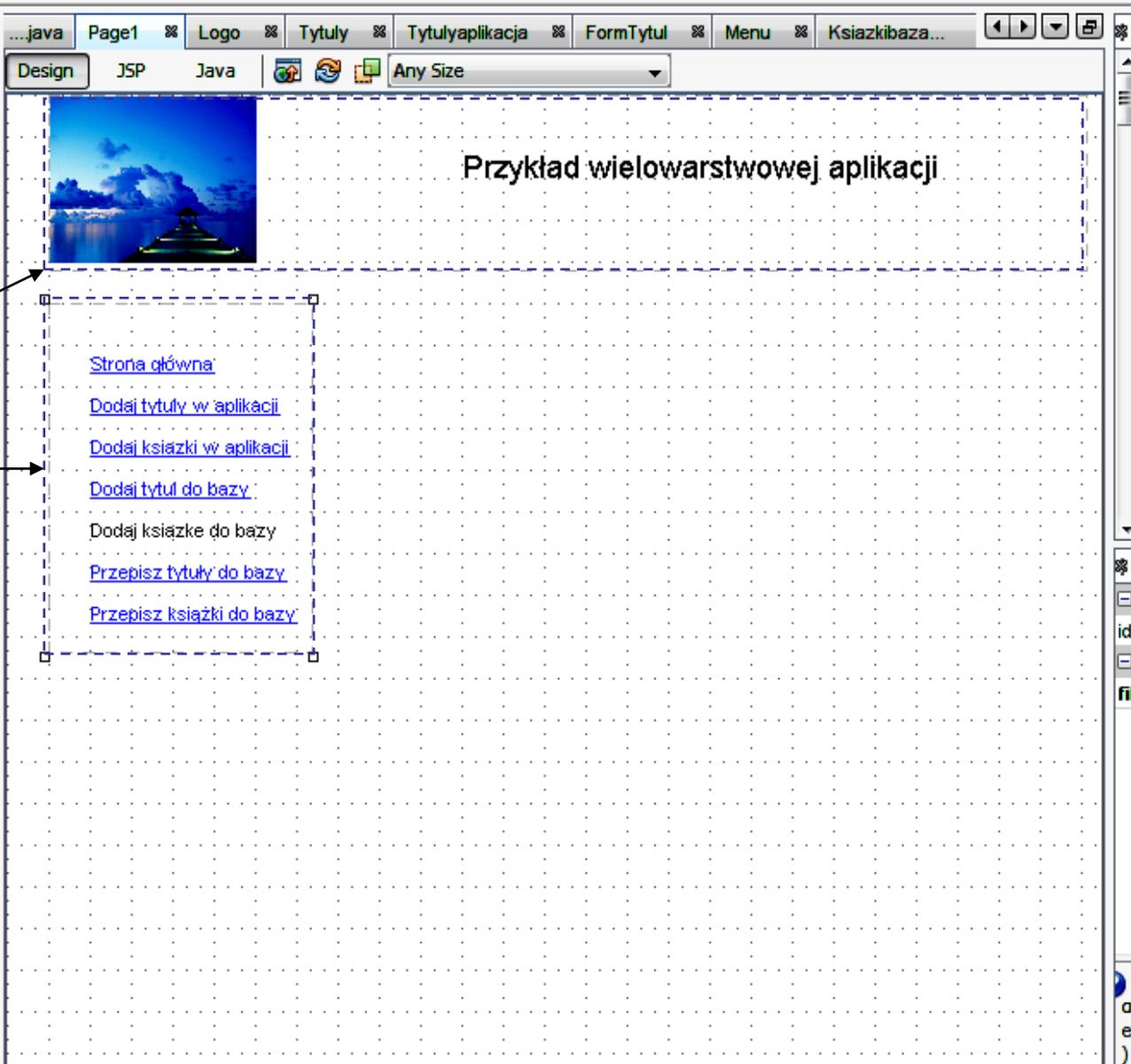
# Projekt formularza głównego „Strona główna” (Page1.jsp) – (6)

WebWypożyczalnia2

- Web Pages
  - WEB-INF
  - resources
  - Baza\_książki.jsp
  - Baza\_tytul.jsp
  - Baza\_tytuly.jsp
  - FormKsiążka.jspf
  - FormTytul.jspf
  - Książki.jsp
  - Książkiaplikacja.jspf
  - Książkibaza.jspf
  - Logo.jspf
  - Menu.jspf
  - Page1.jsp
  - Tytuly.jsp
  - Tytulyaplikacja.jspf
  - Tytulybaza.jspf

Page1

Przykład wielowarstwowej aplikacji



- [Strona główna](#)
- [Dodaj tytuły w aplikacji](#)
- [Dodaj książki w aplikacji](#)
- [Dodaj tytuł do bazy](#)
- Dodaj książkę do bazy
- [Przepisz tytuły do bazy](#)
- [Przepisz książki do bazy](#)

jsp:directive.include:Menu.jspf - Navigator

Page1

- page1
  - html1
    - head1
    - body1
      - form1
        - div
        - div
          - jsp:directive.include:Menu.jspf

RequestBean1

SessionBean1

ApplicationBean1

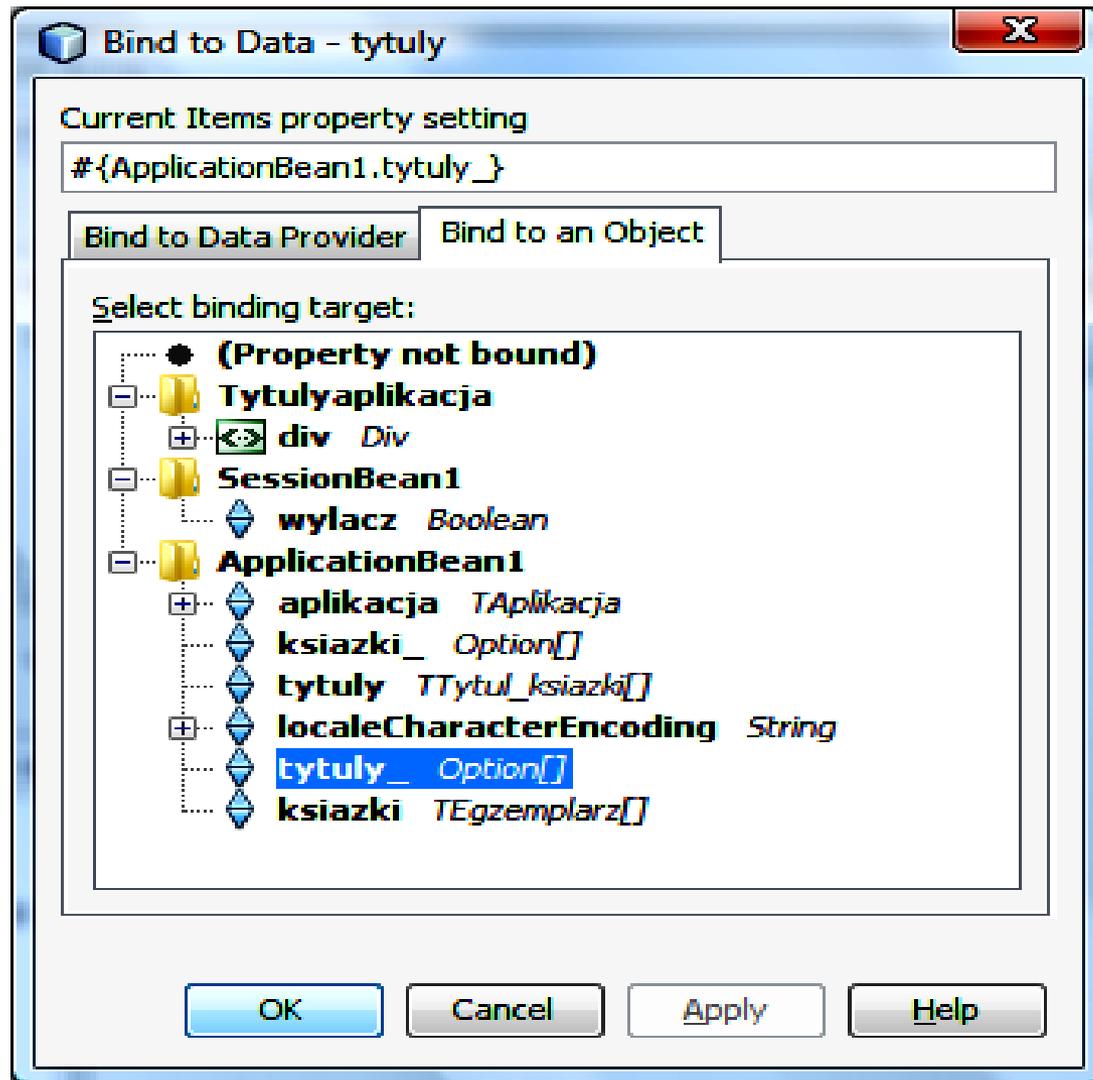


## Przykład wielowarstwowej aplikacji

- [Strona główna](#)
- [Dodaj tytuły w aplikacji](#)
- [Dodaj książki w aplikacji](#)
- [Dodaj tytuł do bazy](#)
- [Dodaj książkę do bazy](#)
- [Przepisz tytuły do bazy](#)
- [Przepisz książki do bazy](#)



Bindowanie tablicy **tytuly\_** obiektów typu **Option**, zawierających dane o tytułach, pobrane z warstwy biznesowej – z komponentem wizualnym (lista rozwijana typu Drop Down List)





## Przykład wielowarstwowej aplikacji

- [Strona główna](#)
- [Dodaj tytuły w aplikacji](#)
- [Dodaj książki w aplikacji](#)
- [Dodaj tytuł do bazy](#)
- [Dodaj książki do bazy](#)
- [Przepisz tytuły do bazy](#)
- [Przepisz książki do bazy](#)

**Tytuł**

**Autor**

**ISBN**

**Wydawnictwo**

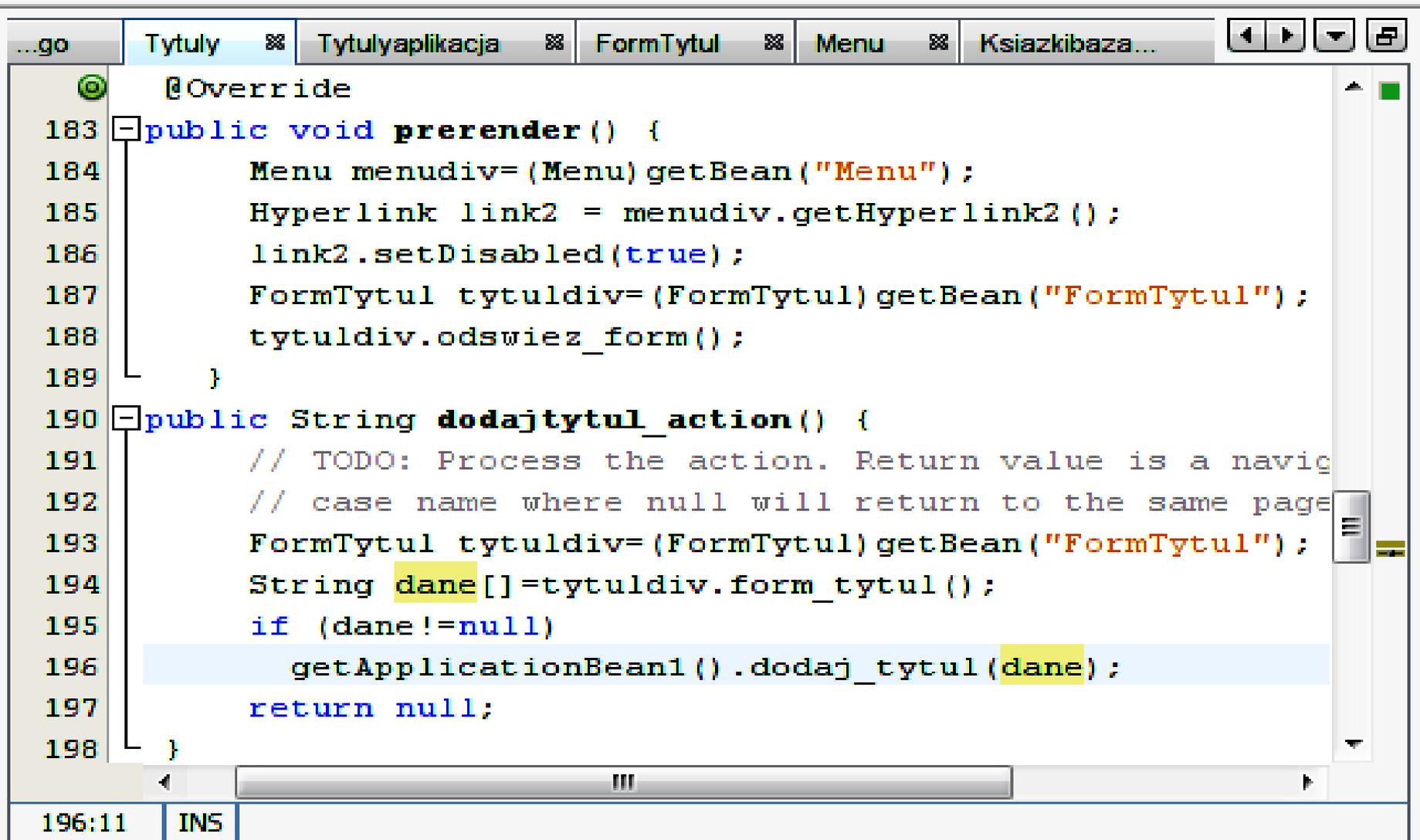
**Aktor**

**Dodaj tytuł**

- Tytuł: 1 Autor: 1 ISBN: 1 Wydawnictwo: 1
- Tytuł: 1 Autor: 1 ISBN: 1 Wydawnictwo: 1**
- Tytuł: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2
- Tytuł: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2
- Tytuł: 3 Autor: 3 ISBN: 3 Wydawnictwo: 3
- Tytuł: 3 Autor: 3 ISBN: 3 Wydawnictwo: 3 Aktor: 3
- Tytuł: 4 Autor: 4 ISBN: 4 Wydawnictwo: 4
- Tytuł: 5 Autor: 5 ISBN: 5 Wydawnictwo: 5 Aktor: 5
- Tytuł: 6 Autor: 6 ISBN: 6 Wydawnictwo: 6

# Oprogramowanie dotyczące formularza **Tytuly.jsp**

Definicje metod w klasie *Tytuly* dla strony typu JSP – do wstawiania nowego tytułu do warstwy biznesowej (obsługa zdarzenia *dodajtytul\_action*) oraz generowania widoku metodą *prerender* (wygaszanie linku do bieżącej strony w formularzu *Menu* typu JSPF i czyszczenie pól formularza *FormTytul* typu JSPF jego metodą *odswiez\_form*)



```
...go | Tytuly | Tytulyaplikacja | FormTytul | Menu | Ksiazkibaza...
@Override
183 public void prerender() {
184     Menu menudiv=(Menu) getBean("Menu");
185     Hyperlink link2 = menudiv.getHyperlink2();
186     link2.setDisabled(true);
187     FormTytul tytuldiv=(FormTytul) getBean("FormTytul");
188     tytuldiv.odswiez_form();
189 }
190 public String dodajtytul_action() {
191     // TODO: Process the action. Return value is a navig
192     // case name where null will return to the same page
193     FormTytul tytuldiv=(FormTytul) getBean("FormTytul");
194     String dane[]=tytuldiv.form_tytul();
195     if (dane!=null)
196         getApplicationBean1().dodaj_tytul(dane);
197     return null;
198 }
196:11 | INS
```

# Utworzenie warstwy biznesowej oraz obiektu typu *TAplikacja*, który jest fasadą warstwy biznesowej w postaci zwykłego obiektu Javy

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help

...java SessionBean1.java \* ApplicationBean1.java \* Page1 Logo...

```
@Override
63 public void init() {
64     // Perform initializations inherited from our sup
65     super.init();
66     // Perform application initialization that must c
67     // *before* managed components are initialized
68     // TODO - add your own initialization code here
69     Managed Component Initialization
80     // Perform application initialization that must c
81     // *after* managed components are initialized
82     // TODO - add your own initialization code here
83
84
85
86
87 }
88 private TAplikacja aplikacja = new TAplikacja();
89 public TAplikacja getAplikacja() {
90     return aplikacja;
91 }
92 public void setAplikacja(TAplikacja aplikacja) {
93     this.aplikacja = aplikacja;
94 }
```

Definicje metod w klasie *ApplicationBean1* związanych z zapisem (*dodaj\_tytul*) i odczytem (*przygotujtytuly*) danych typu kolekcja obiektów *TTytul\_książki* i *TTytul\_książki\_na\_kasecie* w warstwie biznesowej – odczytane dane wstawiane są do tablicy *tytuly\_*, która jest wyświetlana w komponencie typu DropDown List na stronie *Tytulyaplikacja* typu JSPF

```
135 public void dodaj_tytul(String dane[])
136 {getAplikacja().dodaj_tytul(dane); //przypadek użycia "dodaj tytu
137   przygotujtytuly(); //wyswietlenie kolek
138 }
139 private Option tytuly_[] = new Option[0];
140 public Option[] getTytuly_() {
141     return tytuly_;
142 }
143 public void setTytuly_(Option[] tytuly_) {
144     this.tytuly_ = tytuly_;
145 }
146 public void przygotujtytuly() {
147     ArrayList<TTytul_książki> tytuly = aplikacja.getTytul_książki();
148     int ile = tytuly.size();
149     if (ile > 0) {
150         Option pom[] = new Option[ile];
151         Iterator iterator = tytuly.iterator();
152         int i = 0;
153         while (iterator.hasNext()) {
154             pom[i++] =
155                 new Option(Integer.toString(i), iterator.next().toString());
156         }
157         tytuly_ = pom;
158     }
159 }
```

1. PU Dodaj\_tytul
2. Odświeżenie widoku

# Projekt formularza „Przepisz tytuły do bazy” (Baza\_tytuly.jsp) – (8)

The screenshot shows an IDE window for a web application project named "WebWypożyczalnia2". The project structure on the left includes a "Web Pages" folder with various JSP and JSPF files. The "Baza\_tytuly.jsp" file is selected, and its design view is displayed in the main editor. The design view shows a page with a header image, a menu of links, a button labeled "Zapisz tytuły do bazy", and a table titled "Tytuły".

**WebWypożyczalnia2 - Files**

- Web Pages
  - WEB-INF
  - resources
  - Baza\_książki.jsp
  - Baza\_tytul.jsp
  - Baza\_tytuly.jsp
  - FormKsiążka.jspf
  - FormTytul.jspf
  - Książki.jsp
  - Książkiaplikacja.jspf
  - Książkibaza.jspf
  - Logo.jspf
  - Menu.jspf
  - Page1.jsp
  - Tytuly.jsp
  - Tytulyaplikacja.jspf
  - Tytulybaza.jspf

**Baza\_tytuly - Navigator**

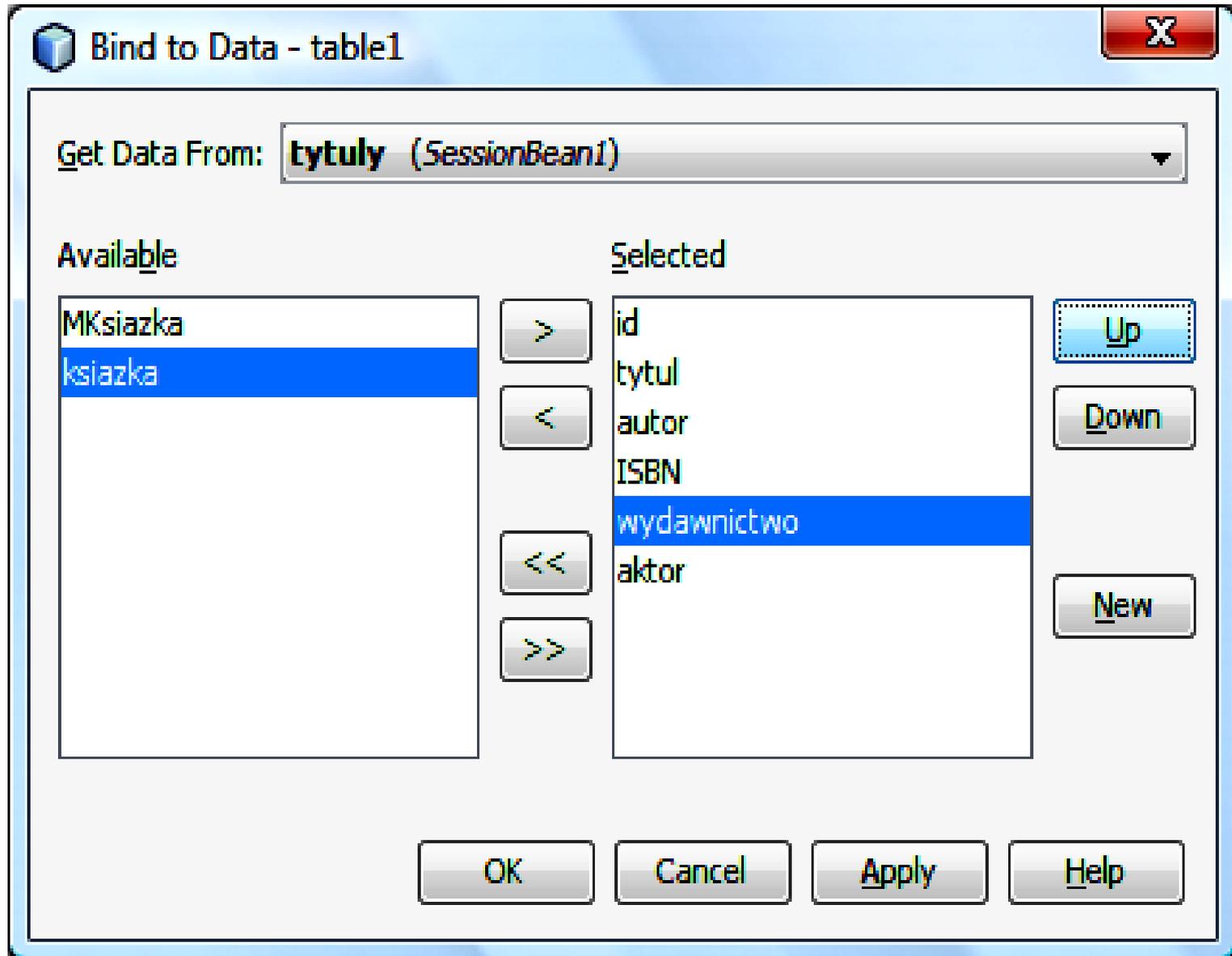
- Baza\_tytuly
  - page1
    - html1
      - head1
      - body1
        - Form1
          - div
          - div
          - div
          - jsp:directive.include:Tytulybaza
          - dodajtytulbaza:Zapisz tytuły do b

**Przykład wielowarstwowej aplikacji**

Zapisz tytuły do bazy

Tytuły						
id	tytuł	autor	ISBN	wydawnictwo	aktor	
123	abc	abc	abc	abc	abc	
123	abc	abc	abc	abc	abc	
123	abc	abc	abc	abc	abc	

Wybór kolumn (pomijanie kolekcji książka mapującej relację **OneToMany** oraz **MKsiążka** wykorzystywanej przez warstwę biznesową do gromadzenia danych o książkach dla danego tytułu)





## Przykład wielowarstwowej aplikacji

Zapisz tytuły do bazy

[Strona główna](#)

[Dodaj tytuły w aplikacji](#)

[Dodaj książki w aplikacji](#)

[Dodaj tytuł do bazy](#)

[Dodaj książkę do bazy](#)

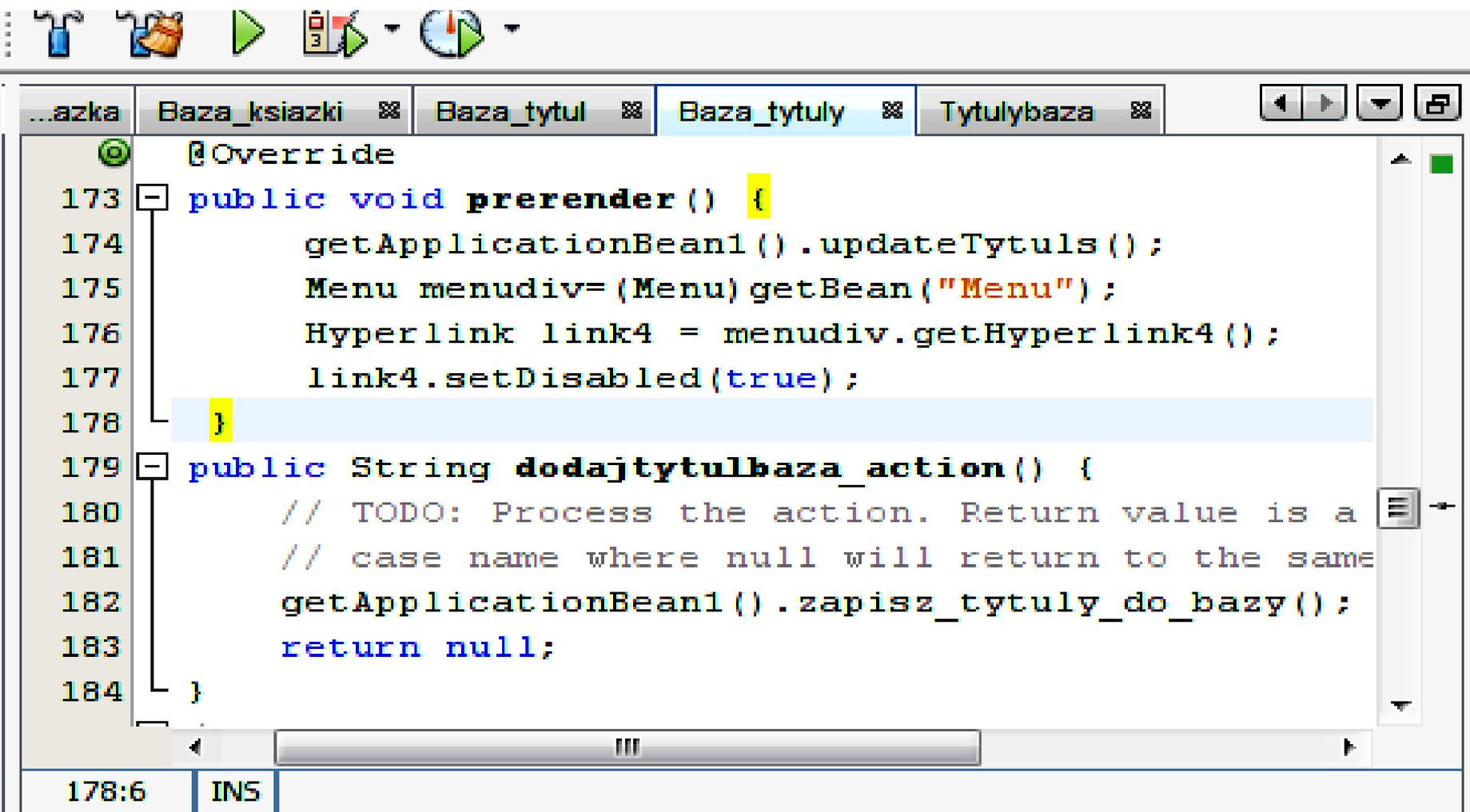
[Przepisz tytuły do bazy](#)

[Przepisz książki do bazy](#)

Tytuły					
id	tytuł	autor	ISBN	wydawnictwo	aktor
2	1	1	1	1	
4	2	2	2	2	2
152	2	2	2	2	
252	3	3	3	3	
352	3	3	3	3	3
353	4	4	4	4	
452	5	5	5	5	5
453	6	6	6	6	

# Oprogramowanie dotyczące formularza **Baza\_tytuly.jsp**

Definicje metod w klasie **Baza\_tytuly** dla strony typu JSP – do zapisu tytułów z warstwy biznesowej (obsługa zdarzenia **dodajtytulbaza\_action**) oraz generowania widoku strony metodą **prerender** (wygaszanie linku do bieżącej strony w formularzu **Menu** typu JSPF i aktualizacja tablicy tytuły metodą **updateTytuls** w klasie **ApplicationBean1** wyświetlanej w komponencie **Table** strony **Tytulybaza** typu JSPF)



```
...azka  Baza_ksiazki  Baza_tytul  Baza_tytuly  Tytulybaza
@Override
173 public void prerender() {
174     getApplicationBean1().updateTytuls();
175     Menu menudiv=(Menu) getBean("Menu");
176     Hyperlink link4 = menudiv.getHyperlink4();
177     link4.setDisabled(true);
178 }
179 public String dodajtytulbaza_action() {
180     // TODO: Process the action. Return value is a
181     // case name where null will return to the same
182     getApplicationBean1().zapisz_tytuly_do_bazy();
183     return null;
184 }
```

178:6 | INS

**Definicje metod *zapisz\_tytuly\_do\_bazy* oraz *updateTytuls* w klasie *ApplicationBean1* związanej z zapisem danych typu kolekcja obiektów *TTytul\_książki* i *TTytul\_książki\_na\_kasce* w bazie danych**

```
116     private TTytul_książki tytuly[];  
117     public TTytul_książki[] getTytuly() {  
118         return tytuly;  
119     }  
120     public void setTytuly(TTytul_książki[] tytuly) {  
121         this.tytuly = tytuly;  
122     }  
123     public void updateTytuls() {  
124         TTytul_książkiController tytulController = new TTytul_książkiController();  
125         tytuly = tytulController.getTTytul_książkis();  
126     }  
127     public void zapisz_tytuly_do_bazy() {  
128         // Add the new Entity to the database using UserController  
129         TTytul_książkiController Tytul_książkiController =  
130             new TTytul_książkiController();  
131         Tytul_książkiController.addTTytul_książki(getApplikacja().getTytul_książki());  
132     }
```

# Wygenerowane tabele bazy danych po uruchomieniu wielowarstwowej aplikacji internetowej (11)

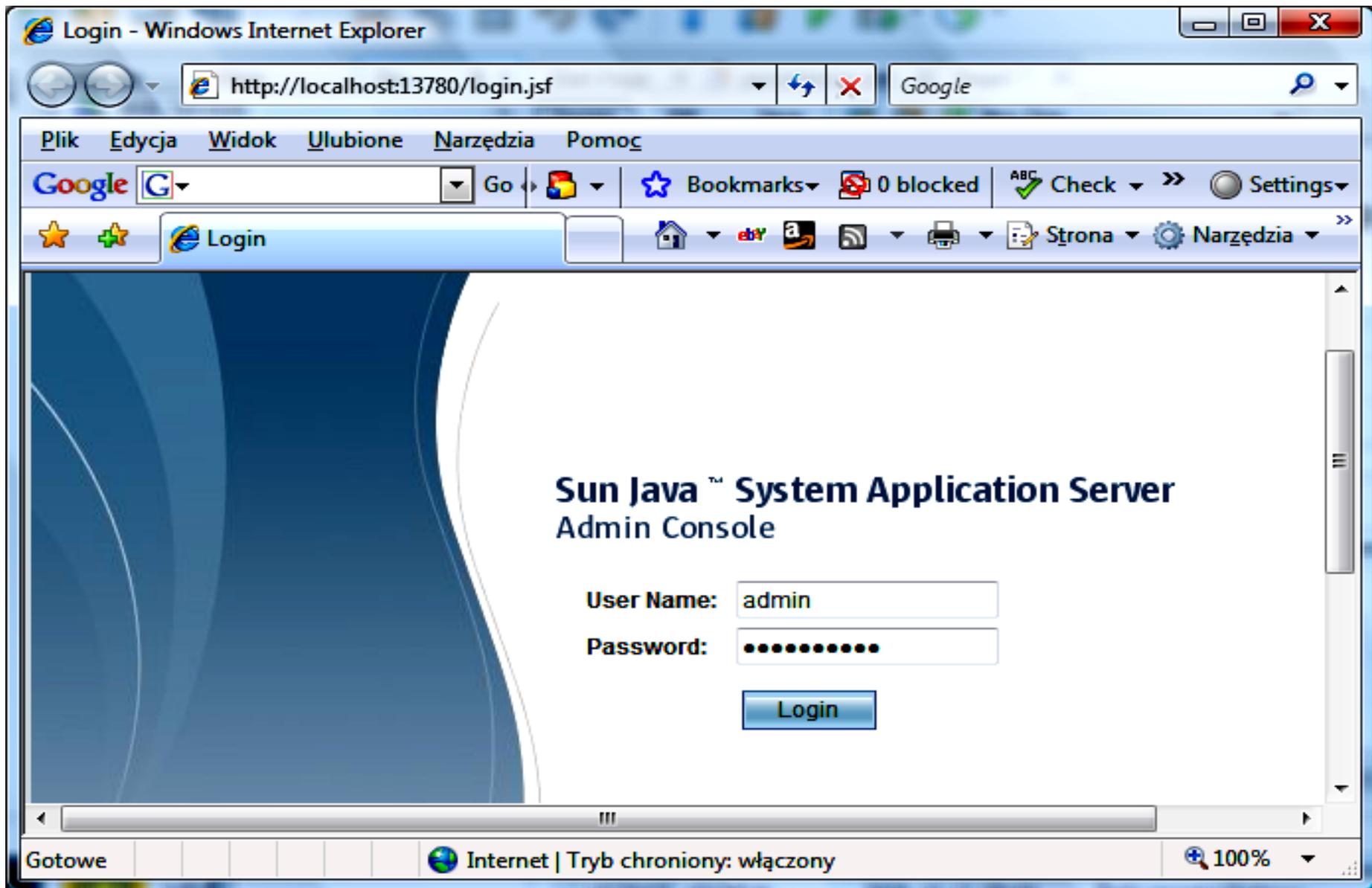
The screenshot shows a database management tool interface. On the left, a tree view displays the database schema for 'KRUUK'. The 'TEGZEMPLARZ' table is selected, showing its columns: ID, DTYPE, NUMER, MTYTUL\_KSIAZKI\_ID, and TERMIN. Below the tree, a SQL Command window contains the query: `select * from KRUUK.TEGZEMPLARZ`. The results of this query are displayed in a table below the command window.

ID	DTYPE	NUMER	MTYTUL_K...	TERMIN
3	TEgzemplarz_termin	1	2	2008-05-06
52	TEgzemplarz	2	4	NULL
102	TEgzemplarz_termin	1	4	2008-05-06
153	TEgzemplarz	1	152	NULL
202	TEgzemplarz	3	4	NULL

# **9. Uwierzytelnianie i autoryzacja oprogramowania**

## **Drugi etap tworzenia warstwy prezentacji**

# (1) Dodanie użytkowników do Serwera aplikacji JavaEE



## (2) Wstawienie zwykłego użytkownika z ograniczonymi uprawnieniami, które zostaną ustawione na poziomie aplikacji

The screenshot shows the Sun Java System Application Server Admin Console interface. The browser address bar displays the URL: `Sun Java System Application Server 9.1_02 Admi...`. The console header includes navigation links for `Home`, `Version`, `Logout`, and `Help`. The user information is `User: admin`, `Domain: personalDomain`, and `Server: localhost`. The main title is `Sun Java™ System Application Server Admin Console`.

The left sidebar shows a tree view of the configuration hierarchy. The `Security` folder is expanded, and the `Realms` sub-folder is selected. The `file` realm is highlighted in blue.

The main content area displays the `New File Realm User` configuration dialog. The breadcrumb path is `Configuration > Security > Realms > file`. The dialog contains the following fields:

- User ID \***: `klient`  
Name of a user to be granted access to this realm; name can be up to 255 characters, must contain only alphanumeric, underscore, dash, or dot characters
- Group List**: `klienci`  
Separate multiple groups with commas
- New Password \***: `.....`
- Confirm New Password \***: `.....`

Buttons for `OK` and `Cancel` are located at the top right of the dialog.

The status bar at the bottom shows `Gotowe`, `Internet | Tryb chroniony: włączony`, and `100%`.

Home Version

Logout Help

User: admin | Domain: personalDomain | Server: localhost

# Sun Java™ System Application Server Admin Console

Configuration > Security > Realms > file

## File Users

Back

Manage user accounts for the currently selected security realm.

Users (2)

	User ID	Group List
<input type="checkbox"/>	klient	klienci
<input type="checkbox"/>	administrator	klienci

- Service Assemblies
  - Components
  - Shared Libraries
- Custom MBeans
- Resources
- Configuration
  - Web Container
  - EJB Container
  - Java Message Servi
  - Security
    - Realms
      - admin-realm
      - file
      - certificate
    - JACC Providers
    - Audit Modules
    - Message Securi
  - Transaction Service

### (3) Konfigurowanie logowania – deskryptor aplikacji *web.xml* po wybraniu opcji *Security* -> *Login Configuration* (deklaratywne konfigurowanie mechanizmów bezpieczeństwa w kontenerze internetowym)

The screenshot shows the NetBeans IDE interface with the *web.xml* file open in the *Security* tab. The *Login Configuration* section is expanded, showing the *Basic* authentication mechanism selected. The *Realm Name* is set to *file*. The *Form Login Page* and *Form Error Page* fields are empty, with *Browse...* buttons next to them. The *Security Roles* section is also visible, showing a table with columns for *Role Name* and *Description*, and buttons for *Add...*, *Edit...*, and *Remove*. The *Security Constraints* section is at the bottom, with an *Add Security Constraint* button.

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help

...age persistence.xml Page1 \* web.xml \* Security

#### Login Configuration

None  
 Digest  
 Client Certificate  
 Basic

Realm Name:

Form

Form Login Page:

Form Error Page:

#### Security Roles

Role Name	Description
-----------	-------------

#### Security Constraints

web.xml - Navigator

Filters:

# (4) Wstawianie ról użytkowników do aplikacji – deskryptor aplikacji *web.xml* po wybraniu opcji *Security*-> *Security Roles* (deklaratywne konfigurowanie mechanizmów bezpieczeństwa w kontenerze internetowym )

The screenshot shows the NetBeans IDE 6.1 interface with the *web.xml* file open in the Security configuration tab. The **Security Roles** section is expanded, displaying a table with the following data:

Role Name	Description
client1	
administrator1	

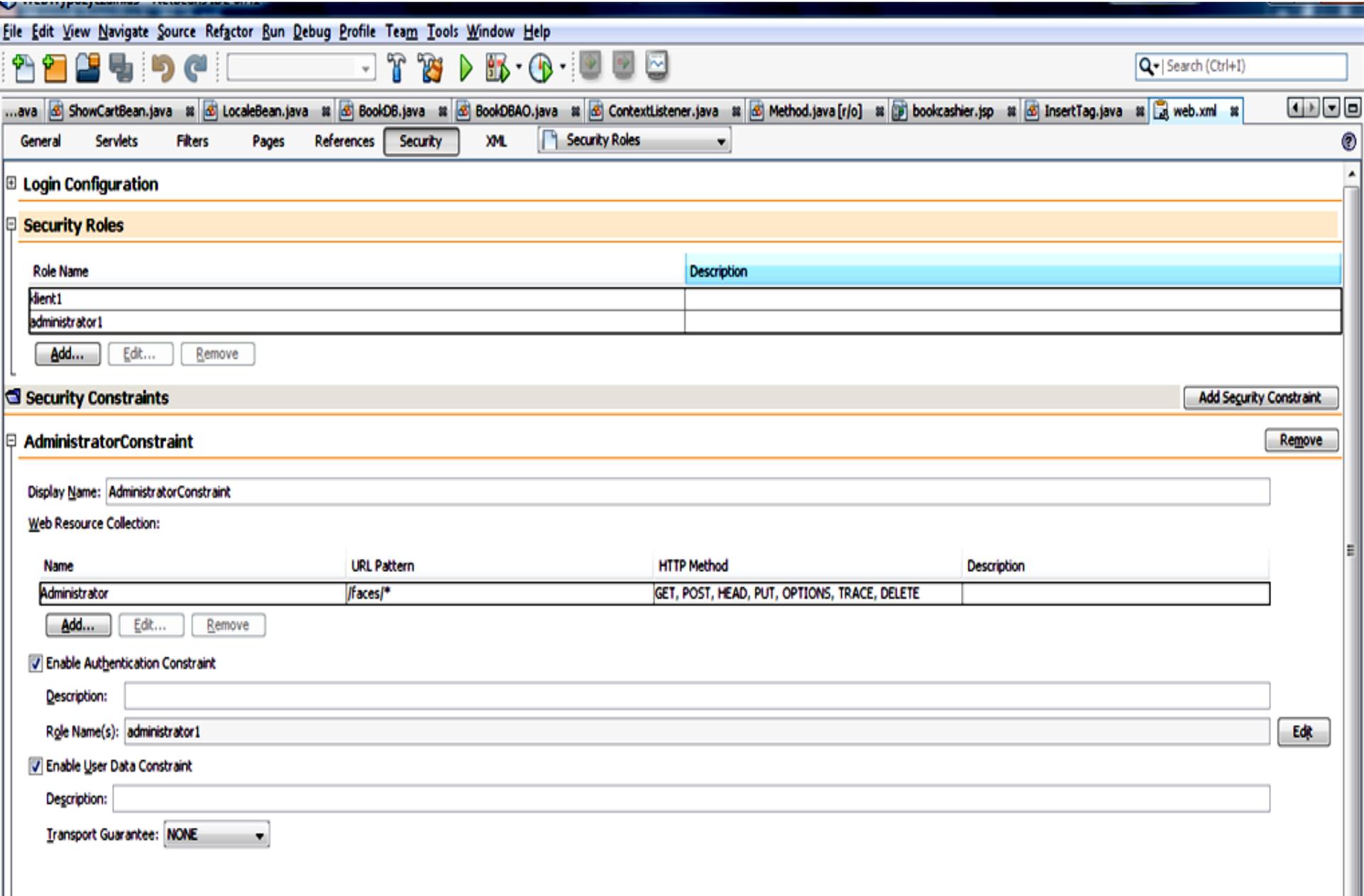
Below the table are buttons for **Add...**, **Edit...**, and **Remove**. The **Security Constraints** section is also visible at the bottom of the configuration pane, with an **Add Security Constraint** button.

The **Properties** panel on the right shows the following details for *web.xml*:

- Name: web
- All Files: E:...
- File Size: 6811
- Modification: 2008...
- Servlet Spec: 2.5

The **web.xml - Navigator** panel at the bottom left shows the file's encoding as "UTF-8" and includes filter icons.

# (5) Wstawianie ograniczeń dla ról użytkowników – deskryptor aplikacji *web.xml* po wybraniu opcji *Security-> Security Constraints*



# (6) Wstawianie ograniczeń dla ról użytkowników – deskryptor aplikacji *web.xml* po wybraniu opcji *Security*-> *Security Constraints*

General Servlets Filters Pages References Security XML Security Roles

### Login Configuration

### Security Roles

Role Name	Description
Klient1	
administrator1	

Add... Edit... Remove

### Security Constraints

Add Security Constraint

#### AdministratorConstraint

Remove

#### KlientConstraint

Remove

Display Name: KlientConstraint

Web Resource Collection:

Name	URL Pattern	HTTP Method	Description
Klient	/faces/Page1.jsp, /faces/Tytuly.jsp, /faces/Ksiazki.jsp	GET, POST, HEAD, PUT, OPTIONS, TRACE, DELETE	

Add... Edit... Remove

Enable Authentication Constraint

Description:

Role Name(s): Klient1

Enable User Data Constraint

Description:

Transport Guarantee: NONE

## (7) Ustawienie czasu sesji

The screenshot shows the NetBeans IDE 6.1 interface. The title bar reads "WebWypożyczalnia3 - NetBeans IDE 6.1". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Build, Run, Profile, Versioning, Tools, Window, and Help. The toolbar contains various icons for file operations and development tools. The left sidebar shows a project tree with folders "Files" and "Services". The "Files" folder is expanded, showing a list of files: ColorChooser, Jsflpa, JsflpaCrud, MovieAdmin, Paint, Paint Application, SinglePageCrudForm, SinglePageCrudTable, and TravelCenter. Below the project tree is the "web.xml - Navigator" panel, which shows two filters: a gear icon and a plus icon. The main editor area displays the configuration for "web.xml". The "General" tab is selected, and the "General" section is expanded. The "Display Name" field is empty. The "Description" field is empty. The "Distributable" checkbox is unchecked. The "Session Timeout" field is set to "30" minutes. The "Content Parameters" section is partially visible at the bottom.

WebWypożyczalnia3 - NetBeans IDE 6.1

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help

ColorChooser  
Jsflpa  
JsflpaCrud  
MovieAdmin  
Paint  
Paint Application  
SinglePageCrudForm  
SinglePageCrudTable  
TravelCenter

web.xml - Navigator

Filters: [gear icon] [plus icon]

General

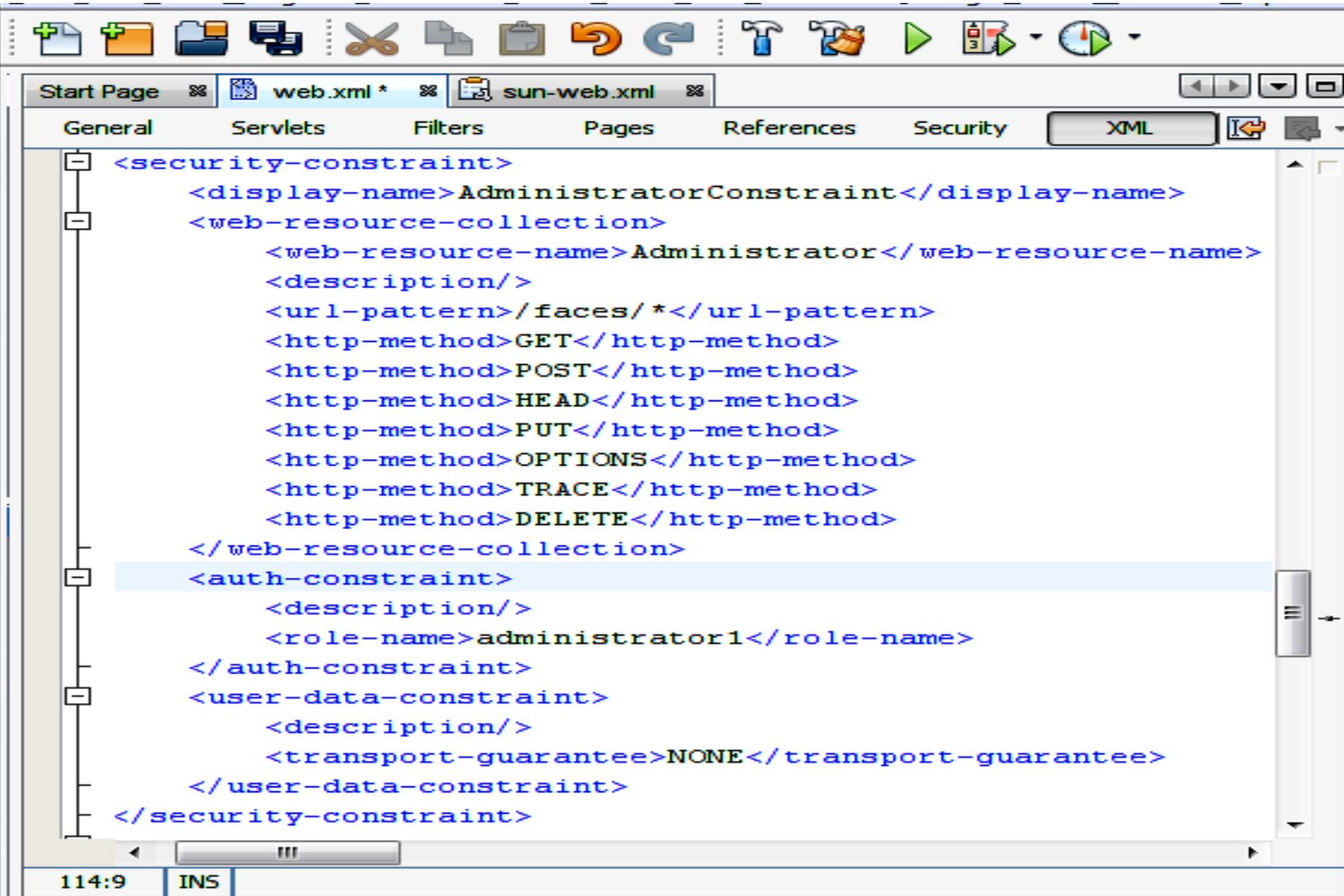
Display Name:

Description:

Distributable

Session Timeout: 30 min.

## (8) Deskryptor aplikacji *web.xml* po wybraniu opcji *XML*

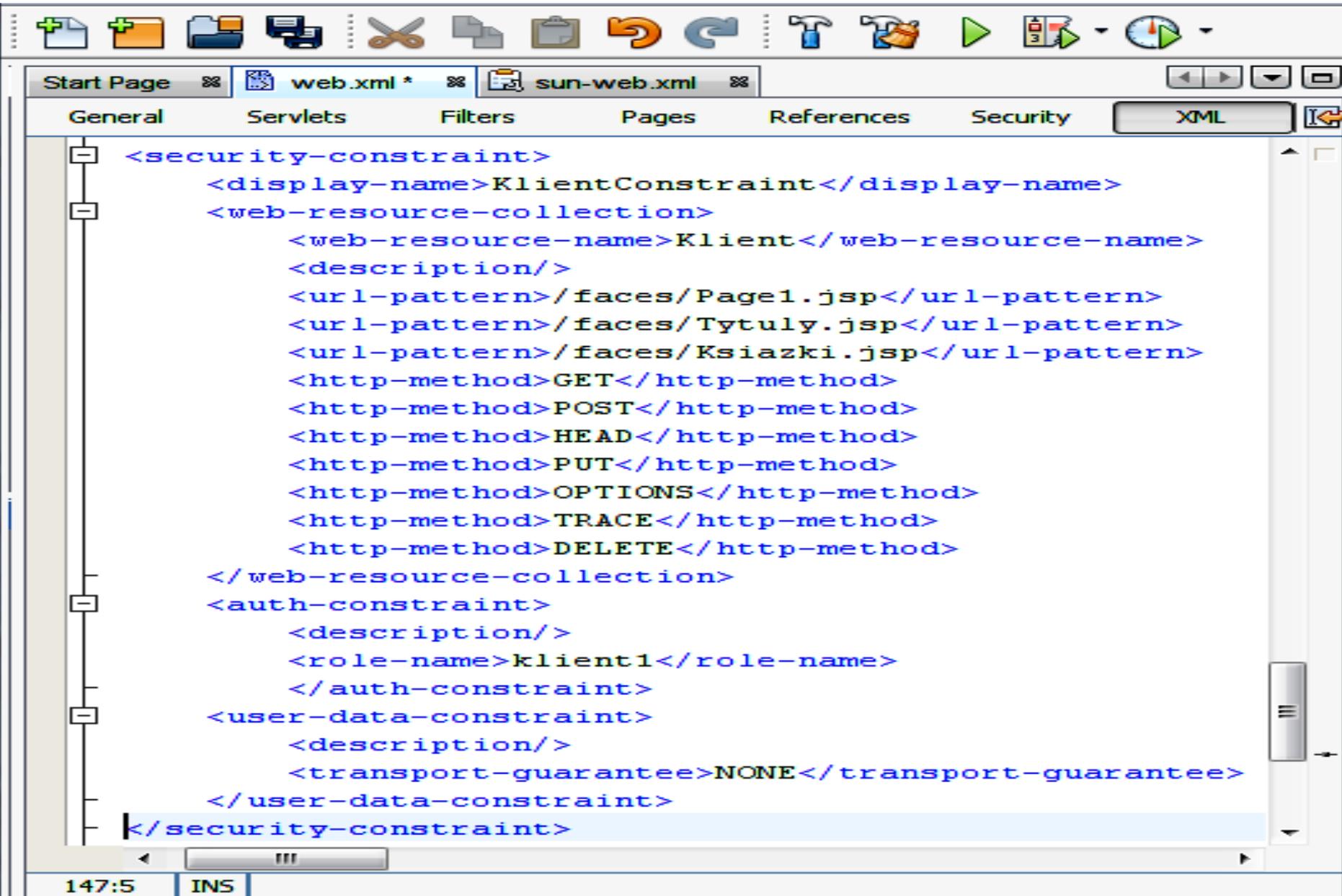


The screenshot shows an IDE window with the following elements:

- Toolbar:** Contains icons for file operations (new, open, save, print, copy, paste, undo, redo), editing (cut, delete), and navigation (back, forward, search).
- Tab Bar:** Shows three tabs: "Start Page", "web.xml \*", and "sun-web.xml".
- Navigation Tabs:** Includes "General", "Servlets", "Filters", "Pages", "References", "Security", and "XML" (which is selected).
- Code Editor:** Displays the XML content of the web.xml file. The current view is the XML tab, showing the security-constraint configuration. The code is as follows:

```
<security-constraint>
  <display-name>AdministratorConstraint</display-name>
  <web-resource-collection>
    <web-resource-name>Administrator</web-resource-name>
    <description/>
    <url-pattern>/faces/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
    <http-method>HEAD</http-method>
    <http-method>PUT</http-method>
    <http-method>OPTIONS</http-method>
    <http-method>TRACE</http-method>
    <http-method>DELETE</http-method>
  </web-resource-collection>
  <auth-constraint>
    <description/>
    <role-name>administrator1</role-name>
  </auth-constraint>
  <user-data-constraint>
    <description/>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```
- Tree View:** Located on the left side, it shows a hierarchical structure of the XML document with expandable nodes.
- Status Bar:** At the bottom left, it displays "114:9" and "INS".

## (9) Deskryptor aplikacji *web.xml* po wybraniu opcji *XML*

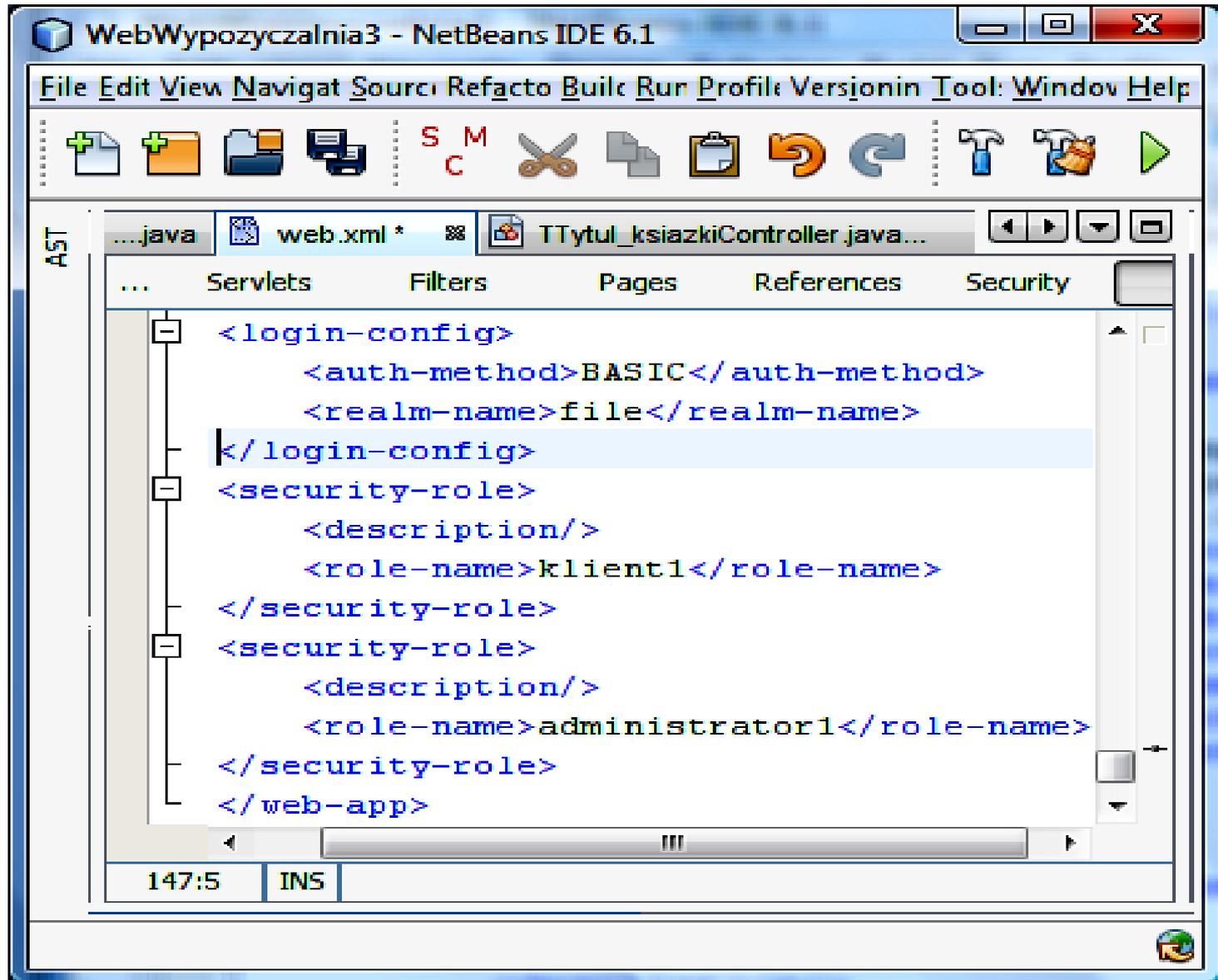


The screenshot shows an IDE window with the following elements:

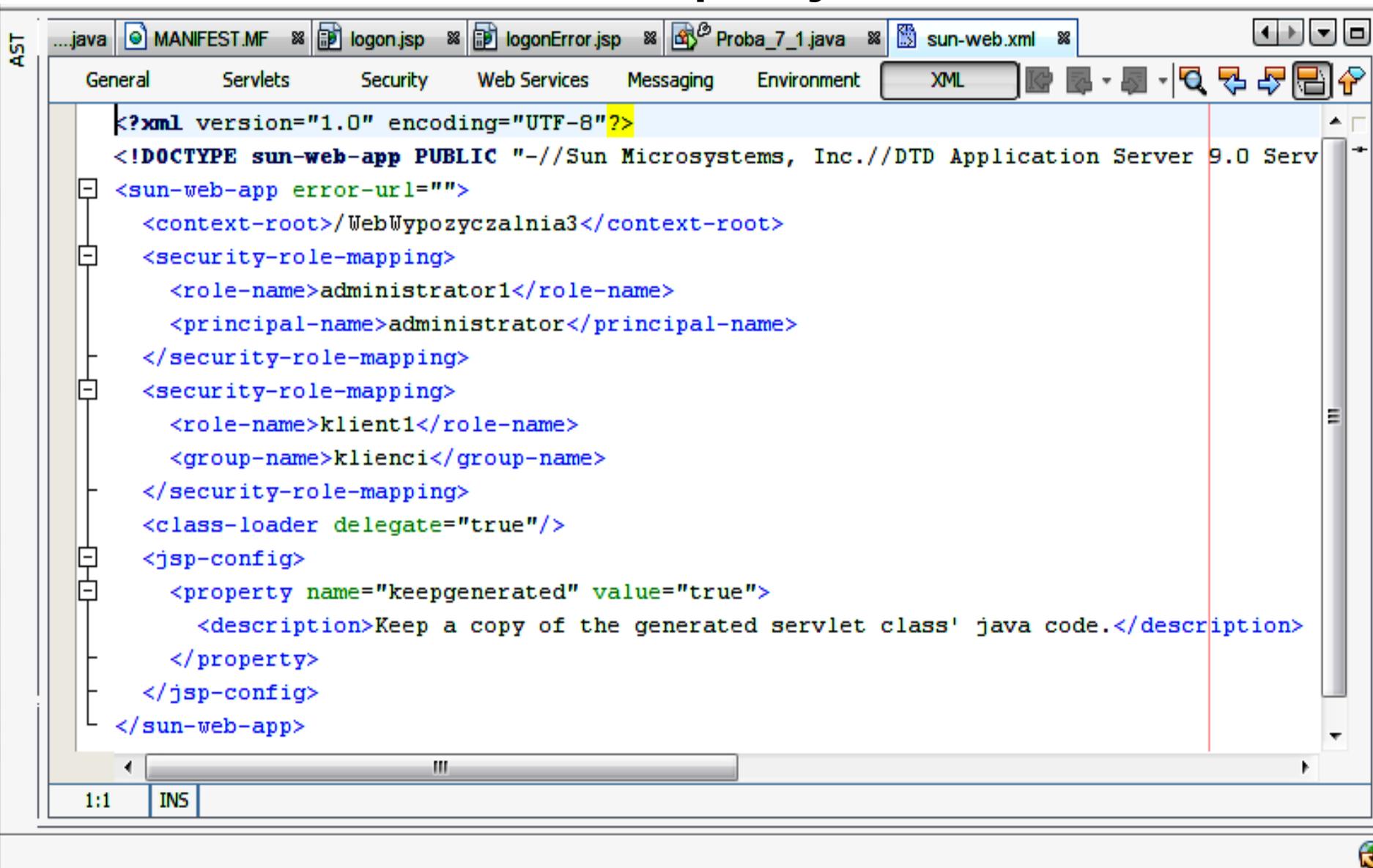
- Toolbar:** Contains icons for file operations (new, open, save, print, copy, paste, delete, undo, redo), navigation (back, forward), and execution (run, debug).
- Tab Bar:** Shows three tabs: "Start Page", "web.xml \*", and "sun-web.xml".
- Navigation Bar:** Includes tabs for "General", "Servlets", "Filters", "Pages", "References", "Security", and "XML". The "XML" tab is currently selected.
- XML View:** A tree view on the left side of the editor shows the structure of the XML document, with expandable nodes.
- Text Editor:** Displays the XML code for the selected node. The code is as follows:

```
<security-constraint>
  <display-name>KlientConstraint</display-name>
  <web-resource-collection>
    <web-resource-name>Klient</web-resource-name>
    <description/>
    <url-pattern>/faces/Page1.jsp</url-pattern>
    <url-pattern>/faces/Tytuly.jsp</url-pattern>
    <url-pattern>/faces/Ksiazki.jsp</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
    <http-method>HEAD</http-method>
    <http-method>PUT</http-method>
    <http-method>OPTIONS</http-method>
    <http-method>TRACE</http-method>
    <http-method>DELETE</http-method>
  </web-resource-collection>
  <auth-constraint>
    <description/>
    <role-name>klient1</role-name>
  </auth-constraint>
  <user-data-constraint>
    <description/>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```
- Status Bar:** Located at the bottom, it shows the cursor position "147:5" and the current mode "INS".

## (10) Deskryptor aplikacji *web.xml* po wybraniu opcji *XML*



# (11) Zawartość deskryptora serwera aplikacji – sun-web.xml po zmapowaniu nazwy użytkownika z serwera aplikacji do roli nadanej mu w aplikacji



The image shows a screenshot of an IDE window displaying the content of the sun-web.xml file. The window title bar includes several tabs: ...java, MANIFEST.MF, logon.jsp, logonError.jsp, Proba\_7\_1.java, and sun-web.xml. The IDE interface has a menu bar with options: General, Servlets, Security, Web Services, Messaging, Environment, and XML. The main editor area shows the XML code for the sun-web-app. The code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Application Server 9.0 Serv
<sun-web-app error-url="">
  <context-root>/WebWypozyczalnia3</context-root>
  <security-role-mapping>
    <role-name>administrator1</role-name>
    <principal-name>administrator</principal-name>
  </security-role-mapping>
  <security-role-mapping>
    <role-name>klient1</role-name>
    <group-name>klienci</group-name>
  </security-role-mapping>
  <class-loader delegate="true"/>
  <jsp-config>
    <property name="keepgenerated" value="true">
      <description>Keep a copy of the generated servlet class' java code.</description>
    </property>
  </jsp-config>
</sun-web-app>
```

At the bottom of the IDE window, the status bar shows "1:1" and "INS".

(12) Uruchomienie aplikacji w trybie uwierzytelniania  
**Basic-Based Authentication** HTTP, zabezpieczenia przez role

Łączenie z localhost



Serwer localhost w lokalizacji file wymaga nazwy użytkownika i hasła.

Ostrzeżenie: ten serwer żąda wysłania Twojej nazwy użytkownika i hasła w niezabezpieczony sposób (podstawowe uwierzytelnienie bez bezpiecznego połączenia).

Nazwa użytkownika:

Hasło:

Zapamiętaj moje hasło

OK Anuluj

Łączenie z localhost



Serwer localhost w lokalizacji file wymaga nazwy użytkownika i hasła.

Ostrzeżenie: ten serwer żąda wysłania Twojej nazwy użytkownika i hasła w niezabezpieczony sposób (podstawowe uwierzytelnienie bez bezpiecznego połączenia).

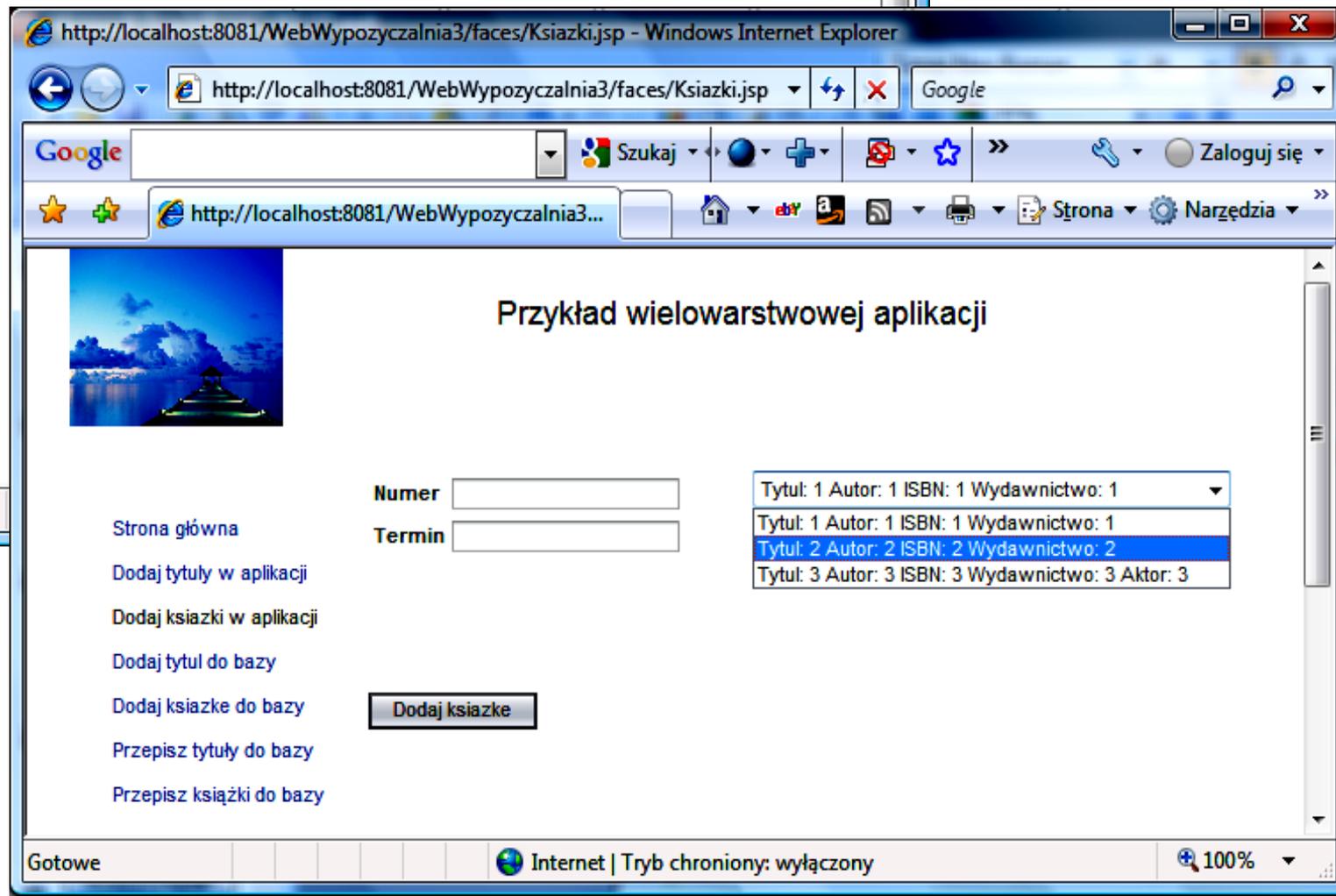
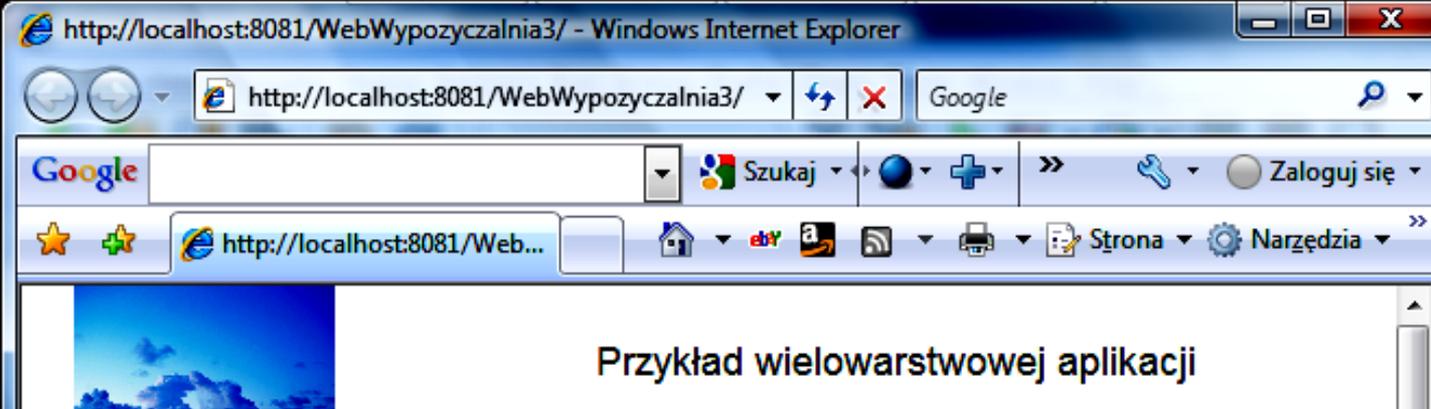
Nazwa użytkownika:

Hasło:

Zapamiętaj moje hasło

OK Anuluj

(13) Zalogowany  
użytkownik  
jako klient



(14) Niedostępne strony dla użytkownika „klient” występującego w roli „klient1” (objęte ograniczeniem **Web Resource Collection**)

The screenshot shows a Windows Internet Explorer browser window. The title bar reads "Sun GlassFish Enterprise Server v2.1 - Error report - Windows Internet Explorer". The address bar contains the URL "http://localhost:8081/WebWypożyczalnia3/faces/Baza\_tytuly.jsp". The search bar is set to "Google". The browser's toolbar includes a search box, a "Szukaj" button, and various utility icons. The main content area displays an error report with a dark blue header that reads "HTTP Status 403 - Access to the requested resource has been denied". Below the header, the report details are as follows:

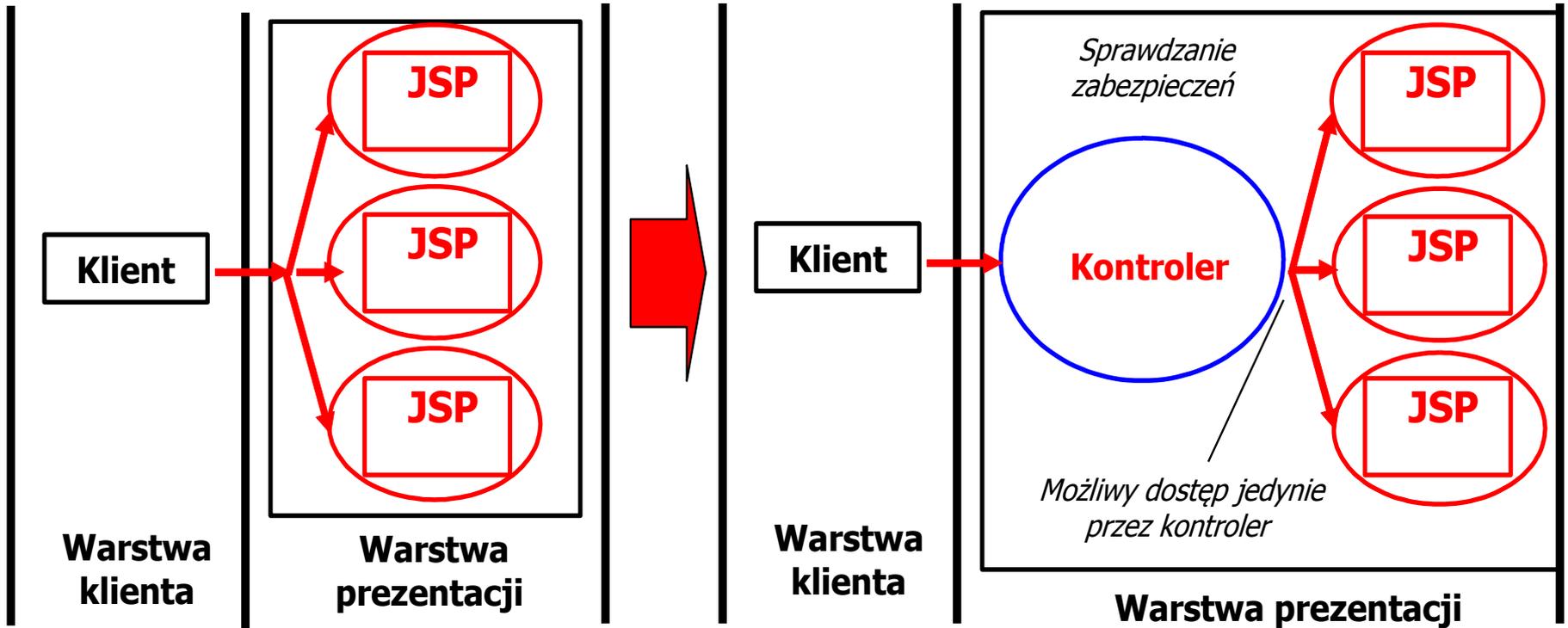
- type** Status report
- message** Access to the requested resource has been denied
- description** Access to the specified resource (Access to the requested resource has been denied) has been forbidden.

At the bottom of the error report, there is a dark blue bar with the text "Sun GlassFish Enterprise Server v2.1". The browser's status bar at the very bottom shows "Gotowe", "Internet | Tryb chroniony: wyłączony", and a zoom level of "100%".



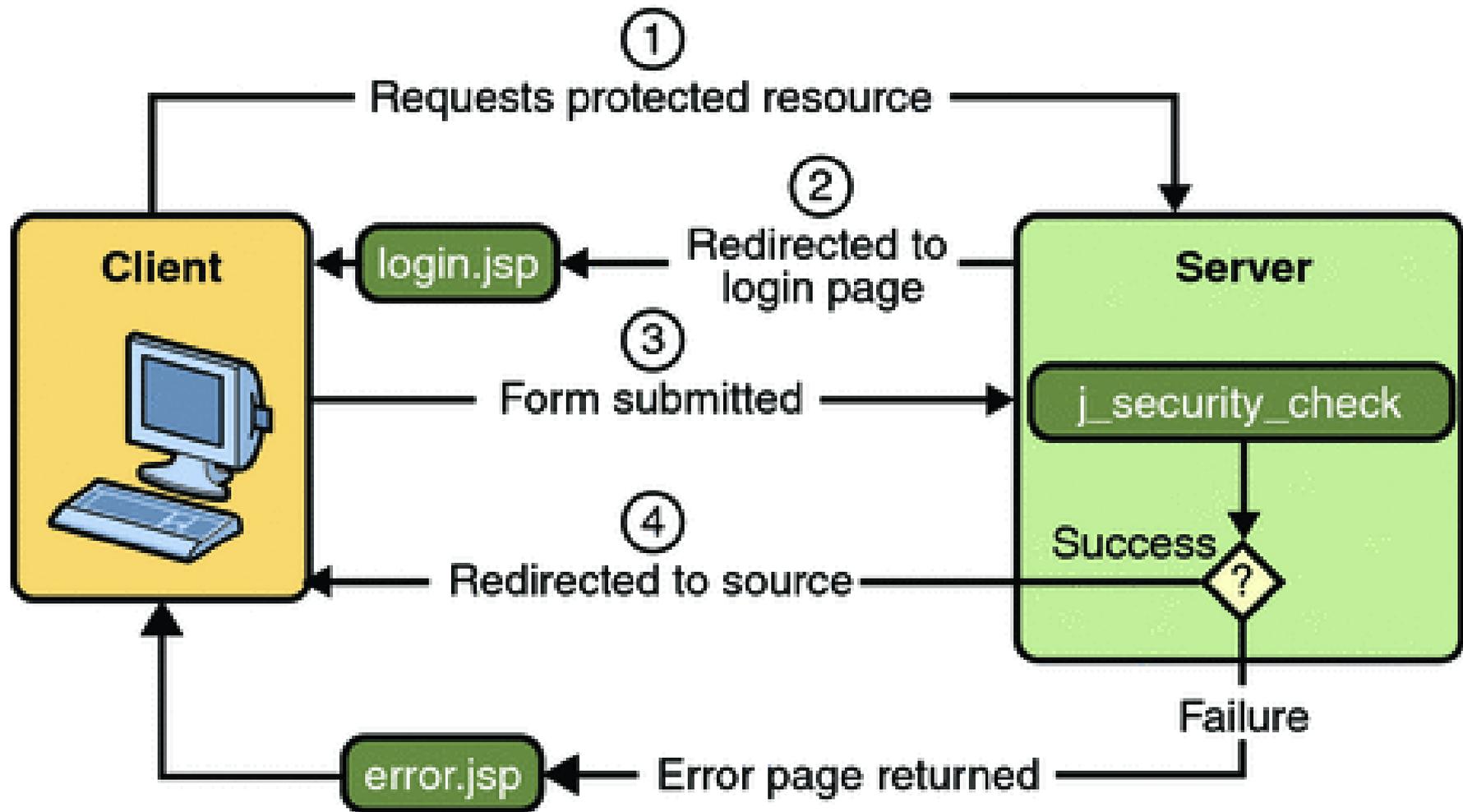
## **2.1. Refaktoryzacja warstwy prezentacji**

# Ukrywanie zasobów przed klientem za pomocą konfiguracji kontenera – **uwierzytelnianie i autoryzacja**



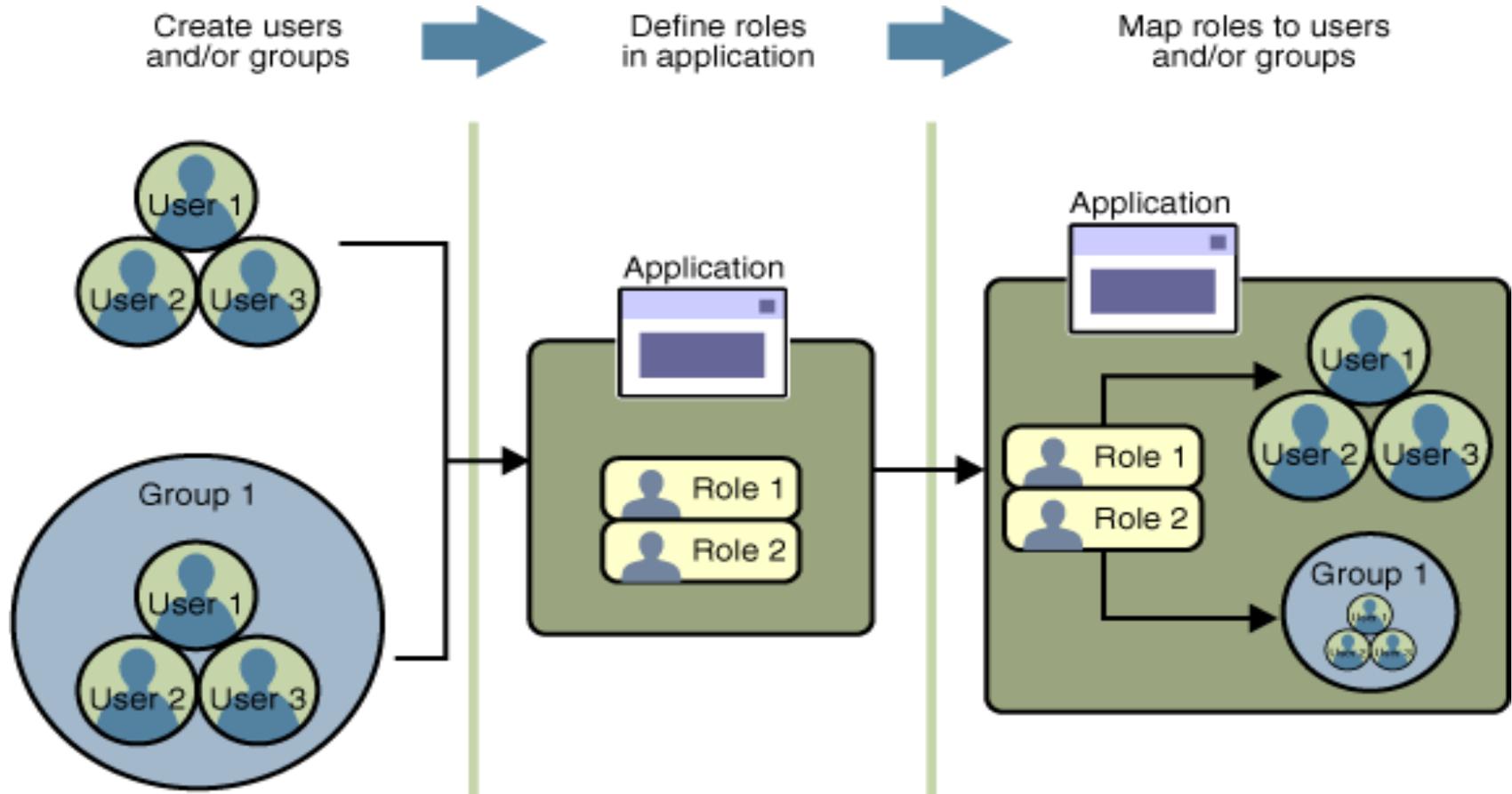
# Przebieg uwierzytelniania (logowania)

<http://download.oracle.com/javaee/5/tutorial/doc/bncbe.html>



# Bazy użytkowników i grup, Użytkownik, Grupa, Rola

<http://download.oracle.com/javaee/5/tutorial/doc/bnbxj.html>



# Rodzaje mechanizmów bezpieczeństwa w kontenerach

- **Deklaratywne mechanizmy bezpieczeństwa** – deklarowane za pomocą tzw. „*deployment descriptors*” (*deskrytory aplikacji np. **web.xml*** dla aplikacji typu **web**). Deskrytory jako zewnętrzny element aplikacji zawierają informację specyfikującą role bezpieczeństwa i wymagania dostępu są mapowane w role specyficzne dla środowiska oraz użytkowników i polisy bezpieczeństwa.
- **Programowe mechanizmy bezpieczeństwa** – są osadzone w aplikacji i służą do podejmowanie decyzji o bezpieczeństwie. Uzupełniają deklaratywne mechanizmy bezpieczeństwa – lepiej wyrażają model bezpieczeństwa aplikacji. API mechanizmów programowych:
  - metody interfejsu EJBContext
  - metody interfejsu HttpServletRequest. Metody te pozwalają na podejmowanie decyzji biznesowych opartych na rolach bezpieczeństwa nadawcy lub zdalnego odbiorcy
- **Adnotacje lub metadane** są używane do specyfikowania informacji wewnątrz pliku z kodem klasy. Kiedy aplikacja jest uruchamiana, informacja ta jest używana lub pokrywana przez deskrytor aplikacji.

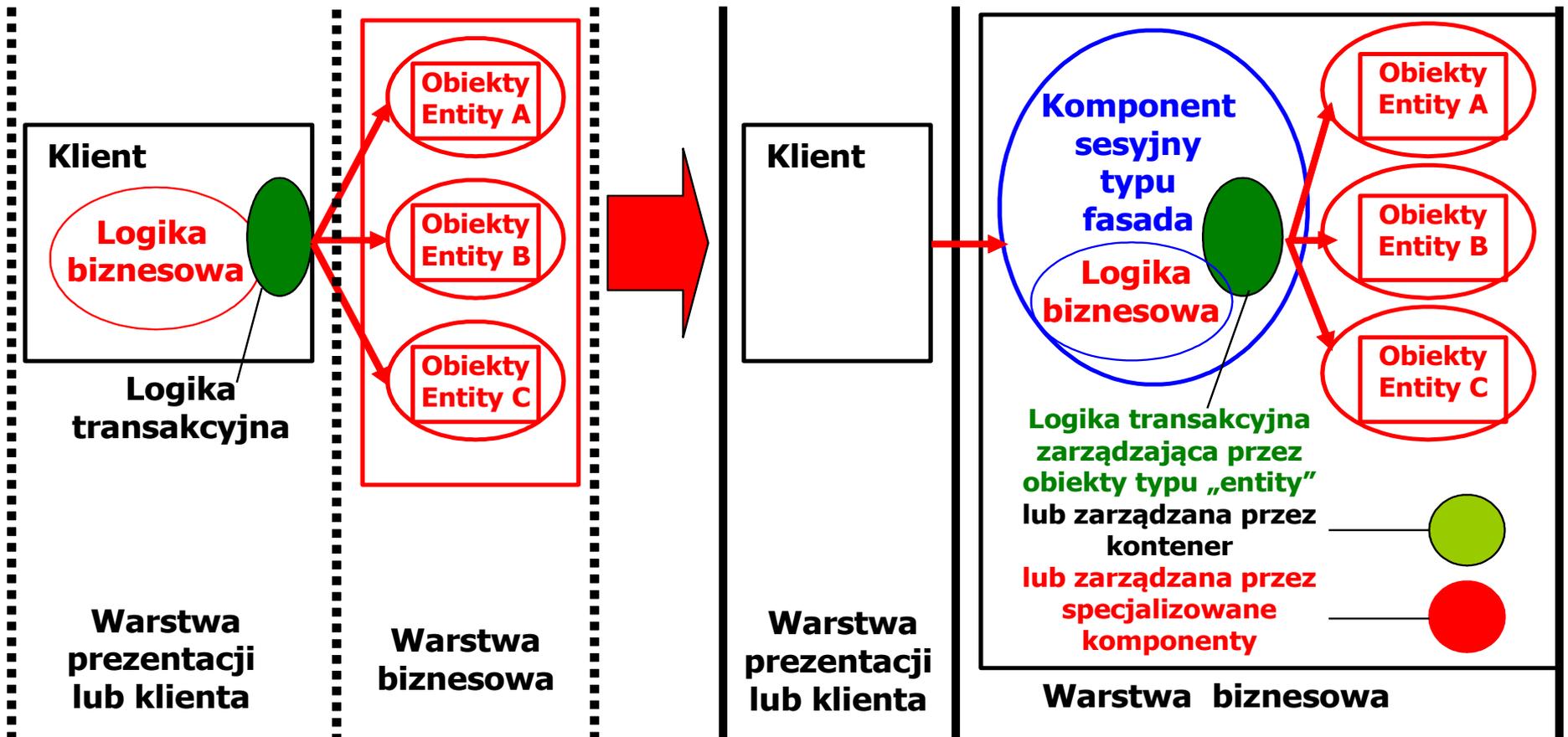
Np.

```
@DeclareRoles(„klient”) public class Page1 extends AbstractPageBean  
{ //... }
```

## **2.2. Refaktoryzacja warstwy biznesowej**

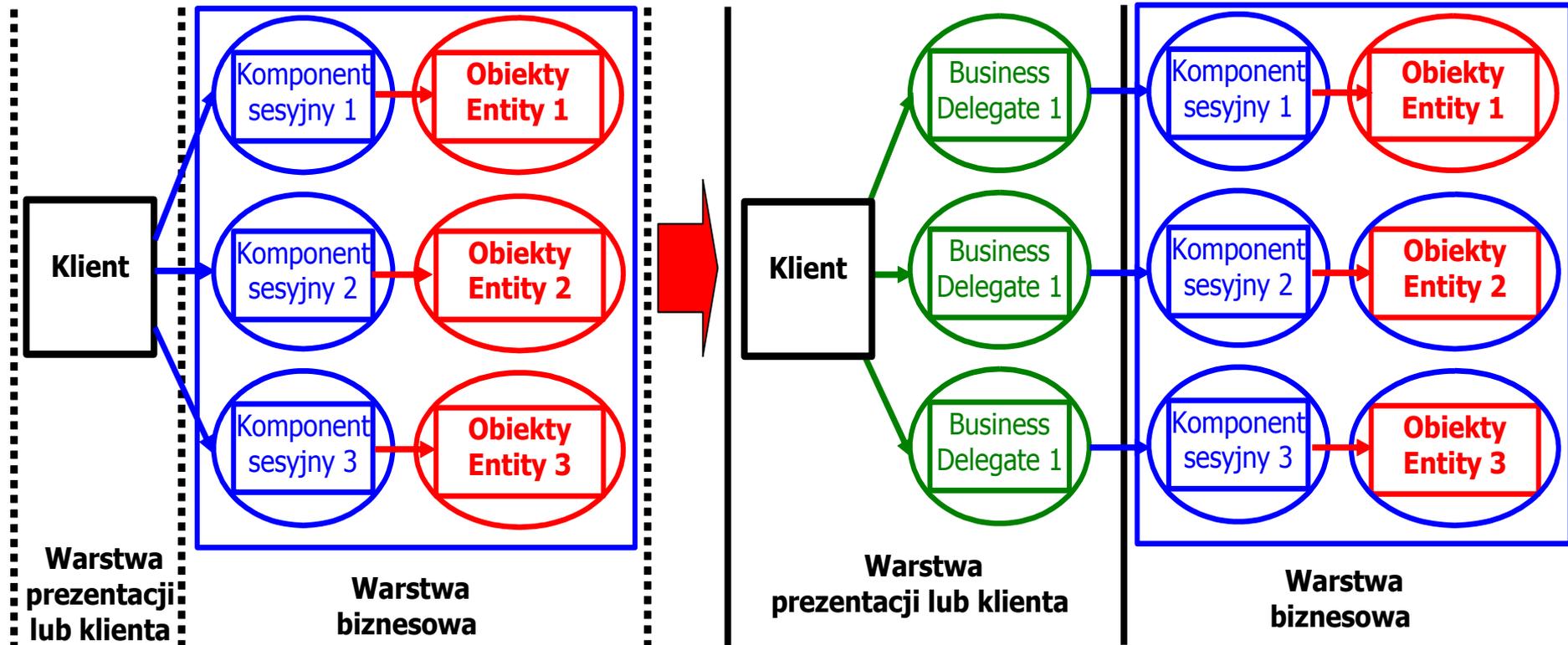
# Refaktoryzacja warstwy biznesowej 1

**Obiekty danych typu „Entity” (obiekty biznesowe)** z warstwy biznesowej są udostępniane klientom w innych warstwach za pomocą **fasadowych komponentów sesyjnych typu „Control” (komponent typu fasada - hermetyzujący dostęp do usług biznesowych)**



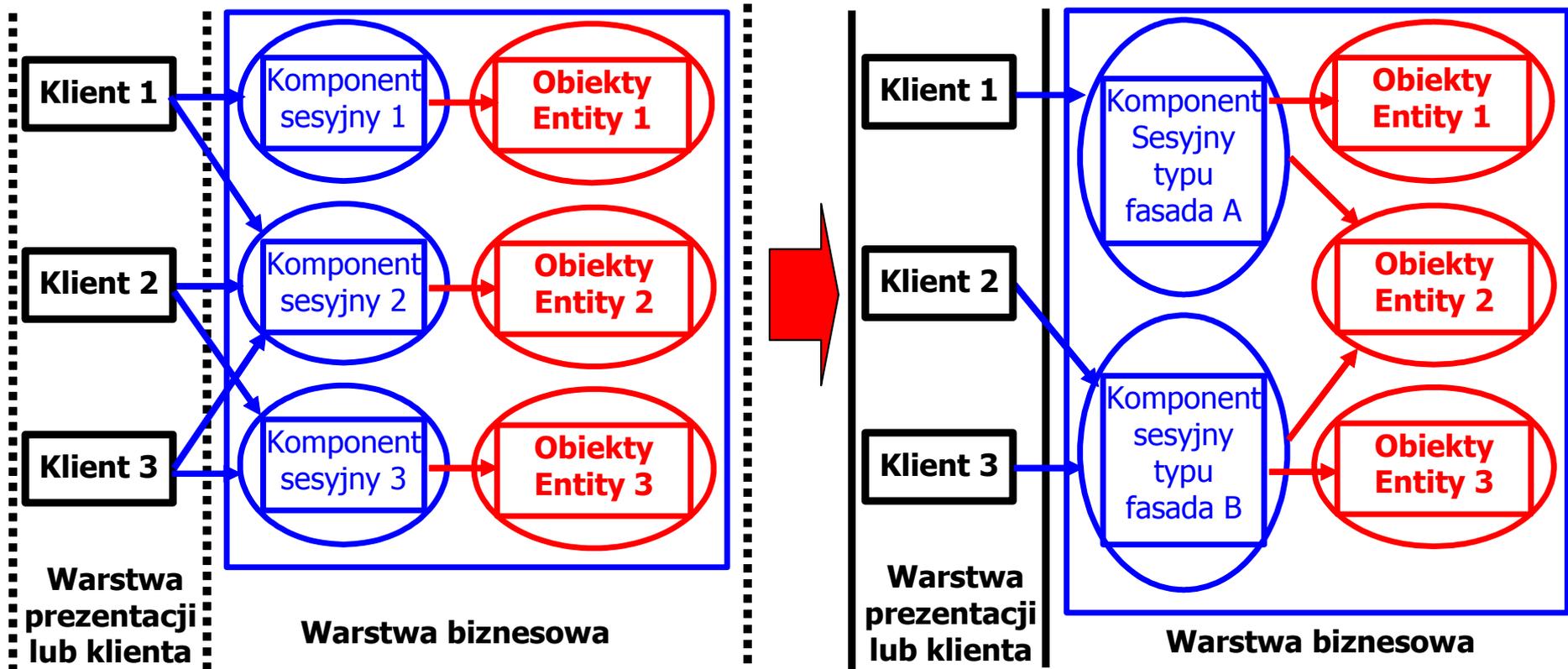
## Refaktoryzacja warstwy biznesowej 2

**Komponenty sesyjne typu „Control”** (pośredniczące w dostępie do **obiektów danych typu „Entity”**) z warstwy biznesowej są udostępniane klientom w innych warstwach za pomocą **obiektów fasadowych typu „Control”** (hermetyzujących dostęp do warstwy biznesowej- **komponentów Business Delegate**)



## Refaktoryzacja warstwy biznesowej 3

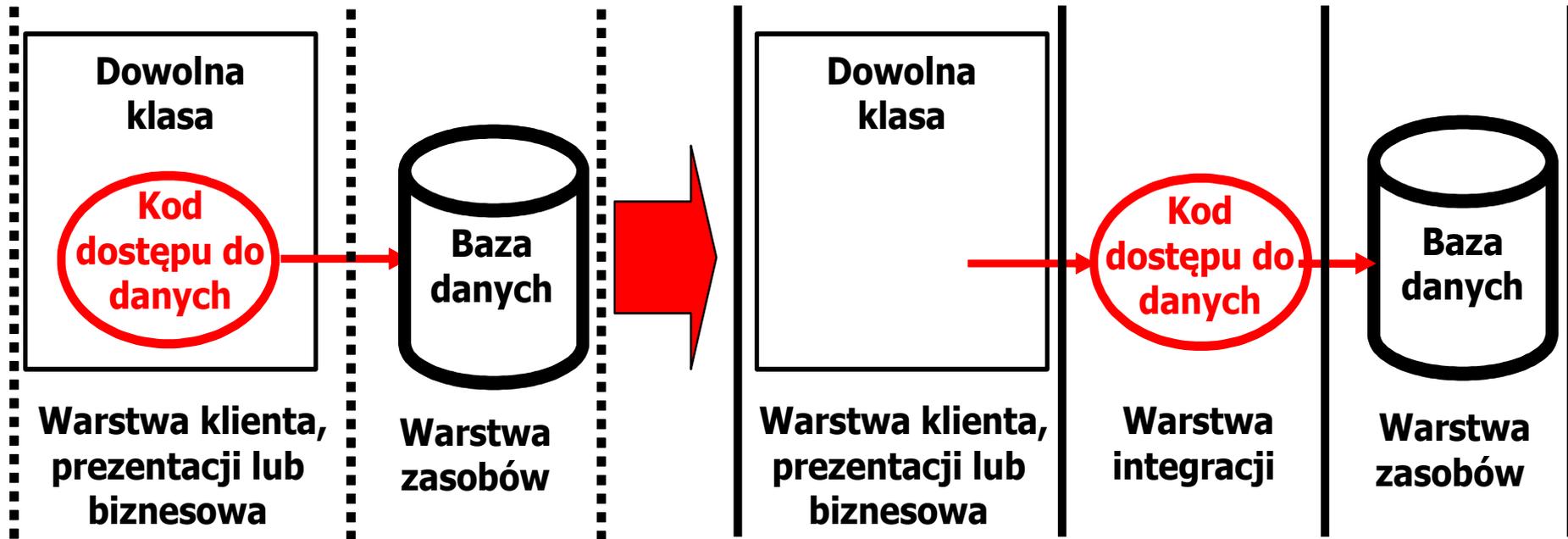
**Sesyjne komponenty fasadowe typu „Control” (każdy komponent jako odrębna usługa biznesowa)**, hermetyzujące **obiekty danych typu „Entity”** z warstwy biznesowej są udostępniane klientom w innych warstwach. Zwykle obiekty sesyjne są jedynie pośrednikami obiektów „Entity”, natomiast nie hermetyzują całych usług, które wymagają odwołania do wielu zwykłych komponentów sesyjnych.



## **2.3. Tworzenie warstwy integracji**

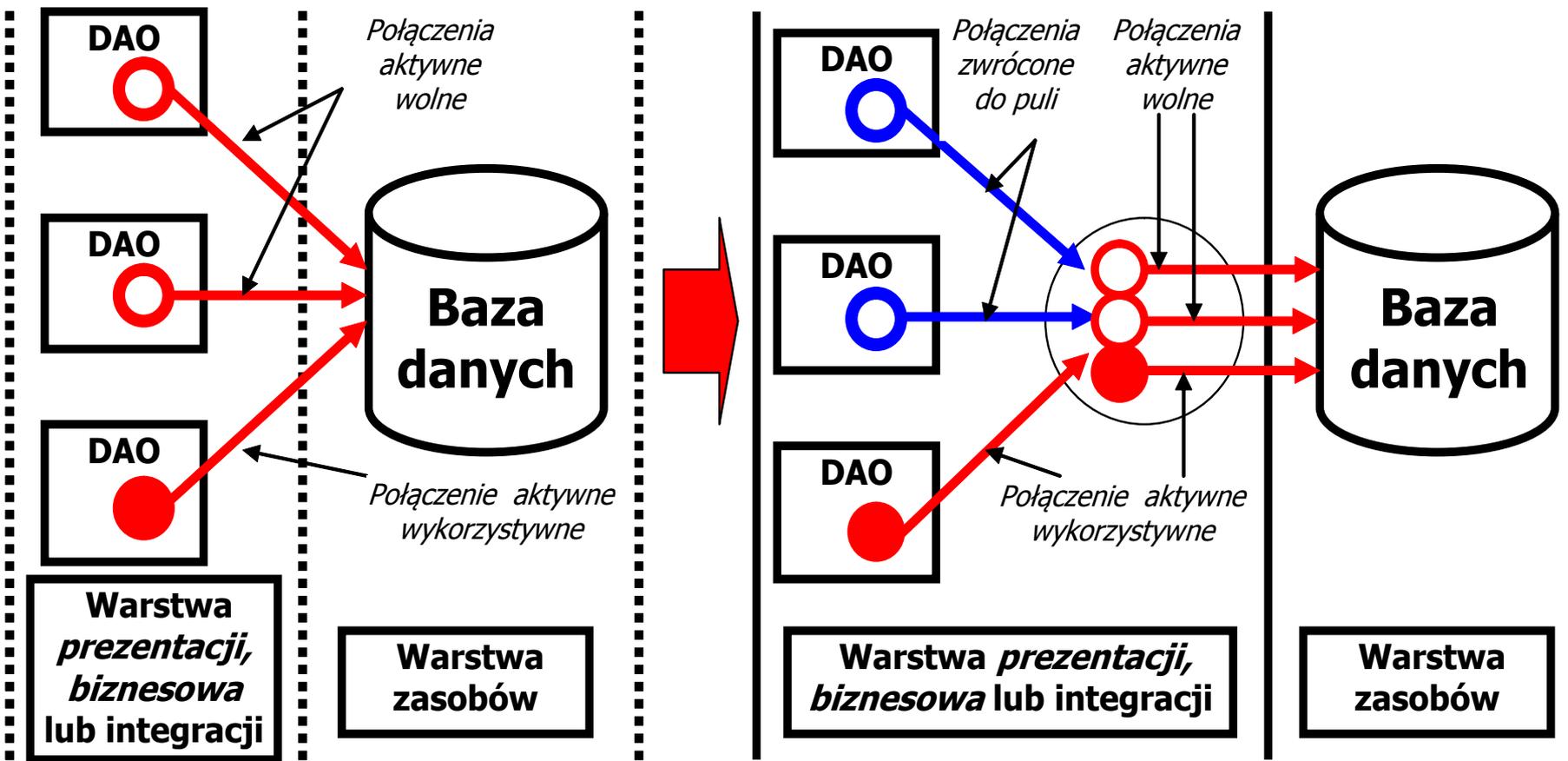
# Wydzielanie kodu dostępu do danych

- Kod dostępu do danych jest wydzielany z klas, które są używane do spełniania również innych celów
- Kod dostępu do danych powinno umieszczać się logicznie i fizycznie bliżej źródła danych



# Refaktoryzacja dostępu do danych – pula obiektów

- Liczba połączeń kodu dostępu do danych (DAO) z bazą danych jest ograniczona
- Połączenia kodu dostępu do danych (DAO) nie zawsze są wykorzystywane, lecz są utrzymywane, ponieważ otwarcie połączenia z bazą danych zabiera i zasoby
- **Pula połączeń** kodu dostępu do danych (DAO) pozwala racjonalnie zarządzać połączeniami aplikacji z bazą danych



```
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; }
}
```

```
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; }
    }
}
```

```
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();
    }
}
```

```
...java Ttytul_książki_na_kascecie.java * Page1 SQL Command 3 SQL Command
package wypożyczalniaapp;

import java.io.Serializable;
import javax.persistence.Entity;

/**
 *
 * @author kruczkiewicz
 */
@Entity
public class Ttytul_książki_na_kascecie extends Ttytul_książki
    implements Serializable {

    private static final long serialVersionUID = 1L;
    private String aktor;

    @Override
    public String getAktor() {
        return aktor;
    }

    @Override
    public void setAktor(String aktor) {
        this.aktor = aktor;
    }

    /* @Override
    public boolean equals(Object object) {
```

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5  package wypożyczalniaapp;
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
24     @Override
25     public Date getTermin() {
26         return termin;
27     }
28
29     @Override
30     public void setTermin(Date termin) {
31         this.termin = termin;
32     }
33 }
```

## Definicje metod w klasie *FormTytul* typu BackingBean dla strony typu JSPF –

- do pobierania danych o nowym tytule (*form\_tytul*): dane dla wstawianych tytułów książek
- do czyszczenia pól formularza (*odswiez\_form*)

```
...acja  FormTytul  Menu  Ksiazkibaza  Ksiazkiaplikacja  Ksiazki  FormKsiazka...
211  public void odswiez_form()
212  {  tytulyp aplikacjaPanel.setRendered(true);
213      tytul.setText("");
214      autor.setText("");
215      ISBN.setText("");
216      wydawnictwo.setText("");
217      aktor.setText("");
218  }
219  public String [] form_tytul()
220  {
221      String jaki;
222      if ((autor.getText().equals("") || tytul.getText().equals("") ||
223          ISBN.getText().equals("") || wydawnictwo.getText().equals(""))
224          return null;
225      if (aktor.getText().equals("")) {
226          jaki = "1";
227      } else {
228          jaki = "3";
229      }
230      String dane[] = {jaki, (String) autor.getText(),
231          (String) tytul.getText(), (String) ISBN.getText(),
232          (String) wydawnictwo.getText(), (String) aktor.getText()};
233      return dane;
234  }
```

212:6 INS

## Zsynchronizowanie zawartości bazy danych i warstwy biznesowej przy starcie aplikacji – wywołanie metod update... przy tworzeniu obiektu ApplicationBean1 (9)

```
@Override
public void init() { // Perform initializations inherited from
    super.init();
    // Perform application initialization that must complete
    // *before* managed components are initialized
    // TODO - add your own initialization code here
    Managed Component Initialization
    // *after* managed components are initialized
    // TODO - add your own initialization code here
    updateTytuls();
    updateKsiazkis();
    updateAplikacja();
    przygotujtytul();
}

private T Aplikacja aplikacja = new T Aplikacja();
public T Aplikacja getAplikacja() {
    return aplikacja;
}

public void setAplikacja(T Aplikacja aplikacja) {
    this.aplikacja = aplikacja;
}

public void updateAplikacja() {
    for (int i = 0; i < tytul.length; i++)
        aplikacja.getTytul_ksiazki().add(tytul[i]);
    Iterator it = aplikacja.getTytul_ksiazki().iterator();
    while (it.hasNext()) {
        TTytul_ksiazki tytul = (TTytul_ksiazki) it.next();
        for (int j = 0; j < ksiazki.length; j++) {
            TTytul_ksiazki tytul1 = ksiazki[j].getMTytul_ksiazki();
            if (tytul1 != null)
                if (tytul1.equals(tytul))
                    { tytul.getMKsiazka().add(ksiazki[j]); }
        }
    }
}
```

Nazwa metody po  
zestandaryzowaniu  
nazw dla atrybutu  
mTytul\_ksiazki w  
klasie TEgzemplarz

Stąd:

getMTytul\_ksiazki  
oraz

setMTytul\_ksiazki

# Przystosowanie do pracy z wieloma wątkami warstwy biznesowej – metody typu synchronized (10)

```
...java  TApplikacja.java *  Tegzemplarz.java  SQL Command 1  SQL Command 6  TTytul_książki.java  TE...
9  package wypożyczalniaapp;
10
11  import java.io.Serializable;
12  import java.util.ArrayList;
13  import java.util.Iterator;
14
15  public class TApplikacja implements Serializable{
16  private ArrayList<TTytul_książki> mTytul_książki = new ArrayList<TTytul_książki>();
17  public TApplikacja() {...}
19  public synchronized TTytul_książki Szukaj_tytul(TTytul_książki tytul_) {...}
27
28  public synchronized TTytul_książki dodaj_tytul(String dane[]) {
29  TFabryka fabryka = new TFabryka();
30  TTytul_książki tytul_książki = fabryka.Podaj_tytul(dane);
31  if (Szukaj_tytul(tytul_książki) == null) {
32  mTytul_książki.add(tytul_książki);
33  return tytul_książki;
34  }
35  return null;
36  }
37
38  public synchronized ArrayList<TTytul_książki> getTytul_książki() {...}
41  public synchronized void setTytul_książki(ArrayList<TTytul_książki> val) {...}
44  public synchronized TTytul_książki dodaj_książke(String dane1[], String dane2[]) {...}
53  public synchronized TTytul_książki Wyszukaj_tytul(String dane[]) {...}
58  public synchronized TEgzemplarz Wyszukaj_egzemplarz(String dane1[], String dane2[]) {...}
68  public synchronized void Wyswietl_książki() {...}
82  public synchronized void Wyswietl_tytuly() {...}
91  public static void main(String t[]) {...}
160 }
```



Projects Files Services Classes

jdbc:derby://localhost:1527/katalog [kruk on KRUK]

- Tables
  - SEQUENCE
    - SEQ\_NAME
    - SEQ\_COUNT
  - Indexes
    - SQL080505160827070
  - Foreign keys
- TEGZEMPLARZ
  - ID
  - DTYPE
  - NUMER
  - MTYTUL\_KSIAZKI\_ID
  - TERMIN
  - Indexes
    - SQL080505160826740
    - SQL080505162122680
  - Foreign keys
    - TGZMPLRZMTYTLKSZKD
      - MTYTUL\_KSIAZKI\_ID -> TTYTUL\_KSIAZKI.ID
- TTYTUL\_KSIAZKI
  - ID
  - DTYPE
  - ISBN
  - TYTUL
  - WYDAWNICTWO
  - AUTOR
  - AKTOR
  - Indexes
    - SQL080505160826800
  - Foreign keys
- Views
- Procedures

...d 13 SQL Command 14 SQL Command 12 SQL Command 15

```
1 select * from KRUK.TTYTUL_KSIAZKI
```

1:1 INS

ID	DTYPE	ISBN	TYTUL	WYDAWNI...	AUTOR	AKTOR
2	TTytul_ksiazki	1	1	1	1	NULL
4	TTytul_ksiazki_na_kasecie	2	2	2	2	2
152	TTytul_ksiazki	2	2	2	2	NULL

# Pomocnicza tabela **Sequence** do generowania kluczy głównych

The screenshot shows a database management tool interface. On the left, a tree view displays the database structure for 'jdbc:derby://localhost:1527/katalog [kruk on KRUK]'. The tree includes tables 'SEQUENCE', 'TEGZEMPLARZ', and 'TTYTUL\_KSIAZKI', each with its columns and indexes listed. The 'SEQUENCE' table has columns 'SEQ\_NAME' and 'SEQ\_COUNT'. The 'TEGZEMPLARZ' table has columns 'ID', 'DTYPE', 'NUMER', and 'MTYTUL\_KSIAZKI\_ID'. The 'TTYTUL\_KSIAZKI' table has columns 'ID', 'DTYPE', 'ISBN', 'TYTUL', 'WYDAWNICTWO', 'AUTOR', and 'AKTOR'. A foreign key relationship is shown between 'TEGZEMPLARZ.MTYTUL\_KSIAZKI\_ID' and 'TTYTUL\_KSIAZKI.ID'. On the right, a SQL query window shows the query 'select \* from' and a table with two columns: 'SEQ\_NAME' and 'SEQ\_COUNT', with a value of '251' under 'SEQ\_COUNT'. The table is labeled '1:1' and 'INS'.

HTTP Monitor

SQL statement(s) executed successfully.

# Właściwości kluczy głównych przy zastosowanej strategii AUTO

NetBeans IDE 6.0.1

File Edit View Naviga Sourc Refact Buil Rui Profi Versjoni Too Windc Hel

Projects Files Serv... Classes

jdbc:derby://localhost:1527/katalog [kruk on KRUK]

Tables

- SEQUENCE
  - SEQ\_NAME
  - SEQ\_COUNT
- Indexes
- Foreign keys
- TEGZEMPLARZ
  - ID**
  - DTYPE
  - NUMER
  - MTYTUL\_KSIAZKI\_ID
  - TERMIN
- Indexes
- Foreign keys
- TTYTUL\_KSIAZKI
  - ID
  - DTYPE
  - ISBN
  - TYTUL
  - WYDAWNICTWO
  - AUTOR
  - AKTOR
- Indexes
- Foreign keys
- Views
- Procedures

**ID - Propert...**

Properties

Name	ID
Type	TABLE
Nulls allowed	<input type="checkbox"/>
Data type	BIGINT
Default value	null
Column size	19
Decimal digits	0
Position	1
Notes	

**ID** Primary Key

Close

NetBeans IDE 6.0.1

File Edit View Naviga Sourc Refact Buil Rui Profi Versjoni Too Windc Hel

Projects Files Serv... Classes

jdbc:derby://localhost:1527/katalog [kruk on KRUK]

Tables

- SEQUENCE
  - SEQ\_NAME
  - SEQ\_COUNT
- Indexes
- Foreign keys
- TEGZEMPLARZ
  - ID
  - DTYPE
  - NUMER
  - MTYTUL\_KSIAZKI\_ID
  - TERMIN
- Indexes
- Foreign keys
- TTYTUL\_KSIAZKI
  - ID**
  - DTYPE
  - ISBN
  - TYTUL
  - WYDAWNICTWO
  - AUTOR
  - AKTOR
- Indexes
- Foreign keys
- Views
- Procedures

**ID - Propert...**

Properties

Name	ID
Type	TABLE
Nulls allowed	<input type="checkbox"/>
Data type	BIGINT
Default value	null
Column size	19
Decimal digits	0
Position	1
Notes	

**ID** Primary Key

Close

Przykład wielowarstwowej aplikacji

[Strona główna](#)  
[Dodaj tytuły w aplikacji](#)  
[Dodaj książki w aplikacji](#)  
[Dodaj tytuł do bazy](#)  
Dodaj książkę do bazy  
[Przepisz tytuły do bazy](#)  
[Przepisz książki do bazy](#)

Select Page Fragment

Select a page fragment to include on the page.

Page Fragment: książkapiplikacja.jspf

Create New Page Fragment...

Close

Page1 - Properties

General

id Page1

Appearance

Background [255,255,255]

Background Image

Page Layout Grid Layout

Style Sheet /resources/stylesheet....

Title

Advanced

Response Encoding UTF-8

Language

Page1

Page1 (request)

Wstawianie fragmentu do strony JSP i wiązanie go z istniejącym plikiem typu JSPF lub tworzenie nowej strony typu JSPF typu Page Fragment Box dla wstawianego komponentu