Tworzenie systemów informatycznych

Inżynieria oprogramowania

Zofia Kruczkiewicz

Zagadnienia

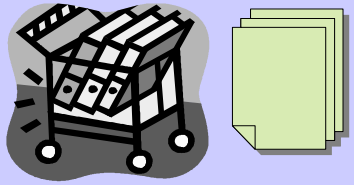
1. Wielowarstwowa architektura systemu informatycznego
2. Refaktoryzacja architektury wielowarstwowej systemu informatycznego
3. Wzorce projektowe
4. Przykład warstwy biznesowej stosującej wzorce obiektowe
5. Przykłady architektury wielowarstwowej w środowisku Visual Web Java server Faces
6. Tworzenie warstwy zasobów - bazy danych w systemie baz danych Derby
7. Tworzenie warstwy integracji w projekcje 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 prezentacj

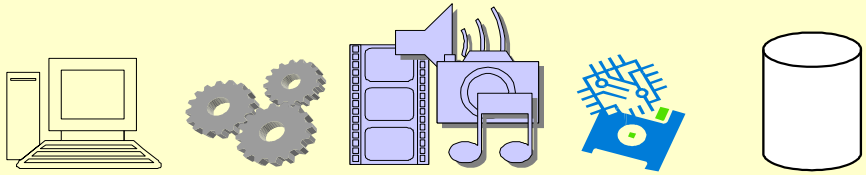
1. Wielowarstwowa architektura systemu informatycznego

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 zewnetrznymi

Adaptery zasobów, systemy zewnętrzne, mechanizmy zasobów, przepływ sterowania

Warstwa zasobów
Bazy danych, systemy zewnętrzne i pozostałe zasoby

Zasoby, dane i usługi zewnętrzne

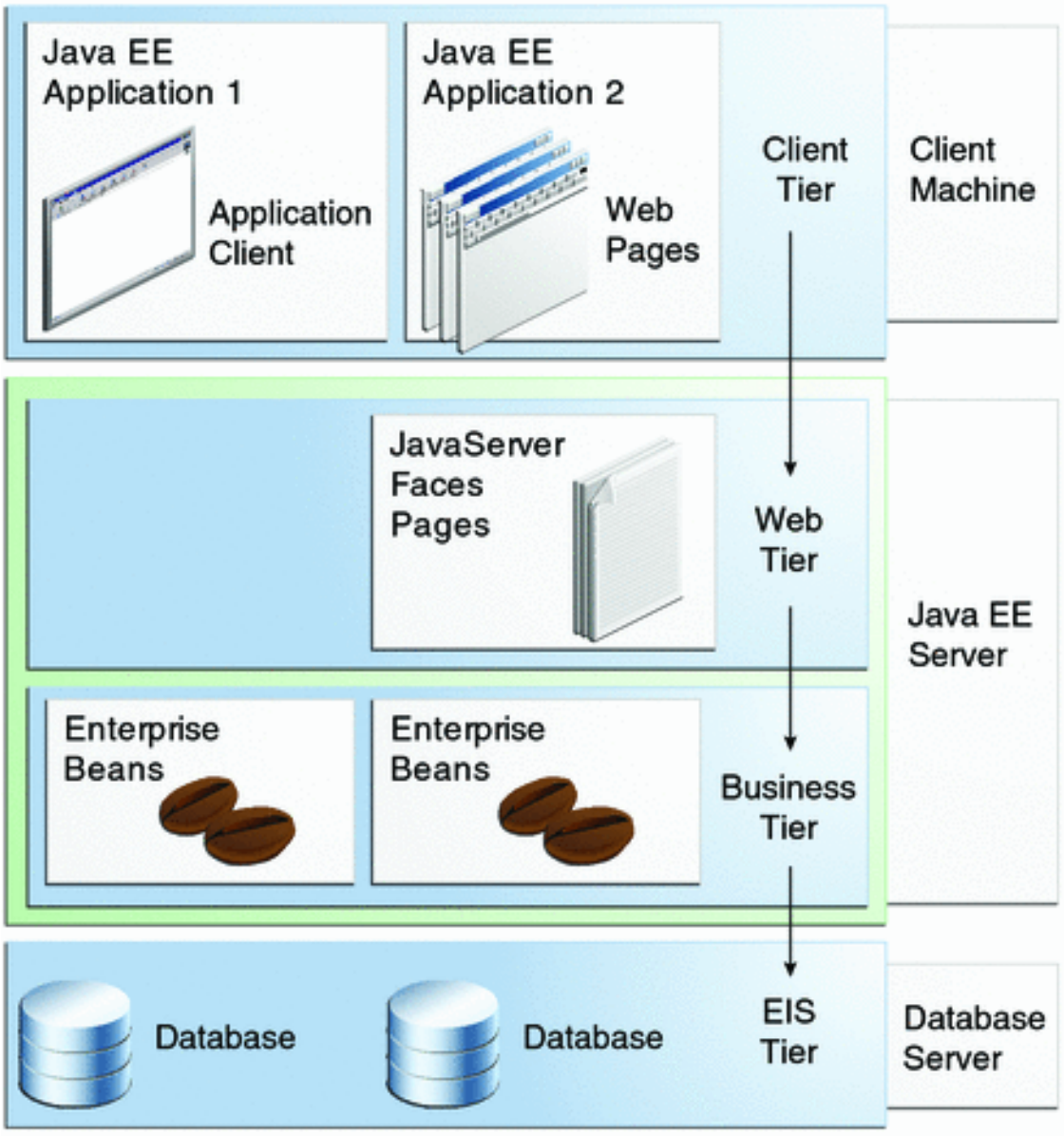
(1) Warstwy aplikacji (Java EE)

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

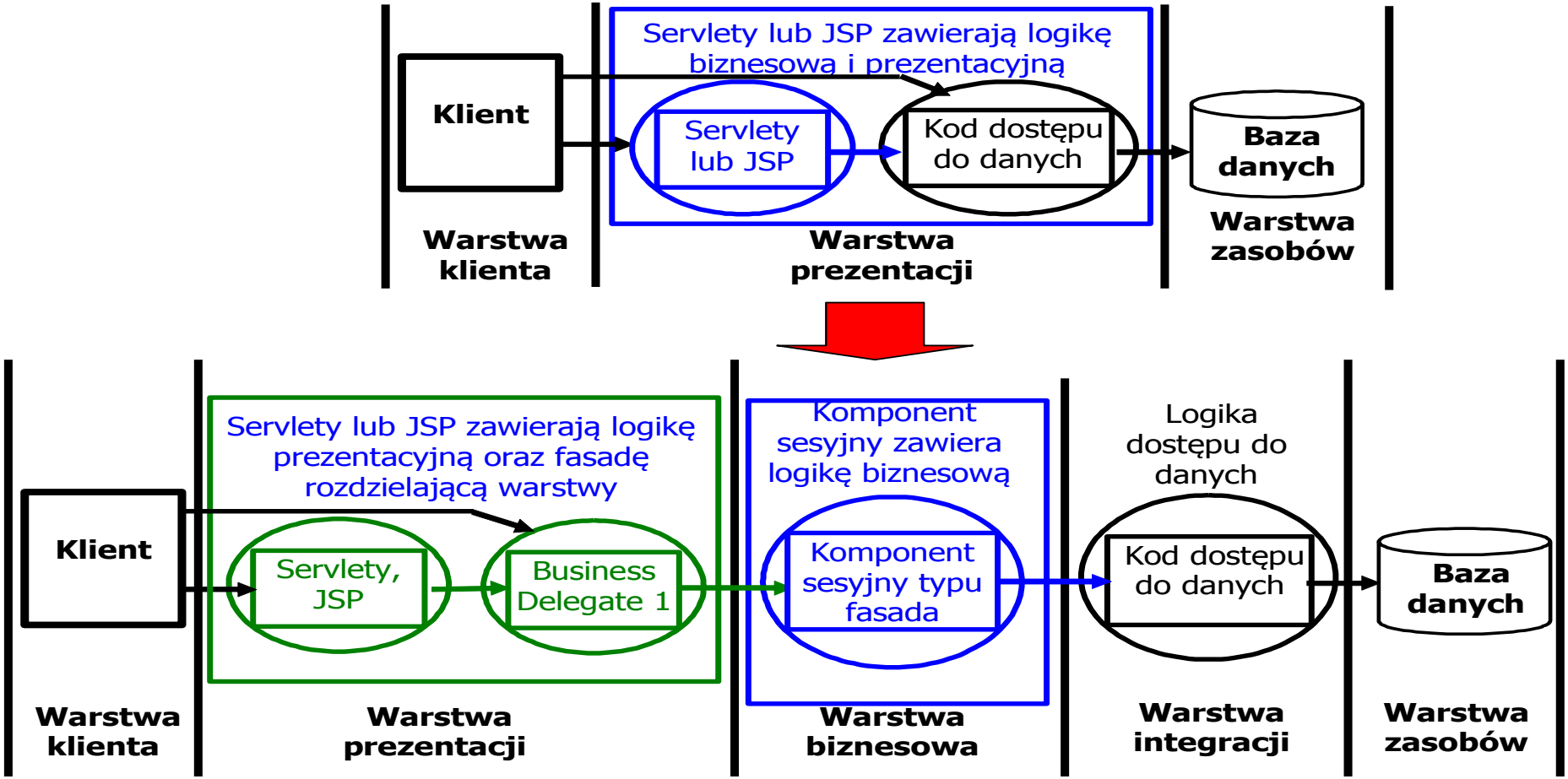
(wg. D.Alur, J.Crupi, D. Malks, Core J2EE. Wzorce projektowe.)

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

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

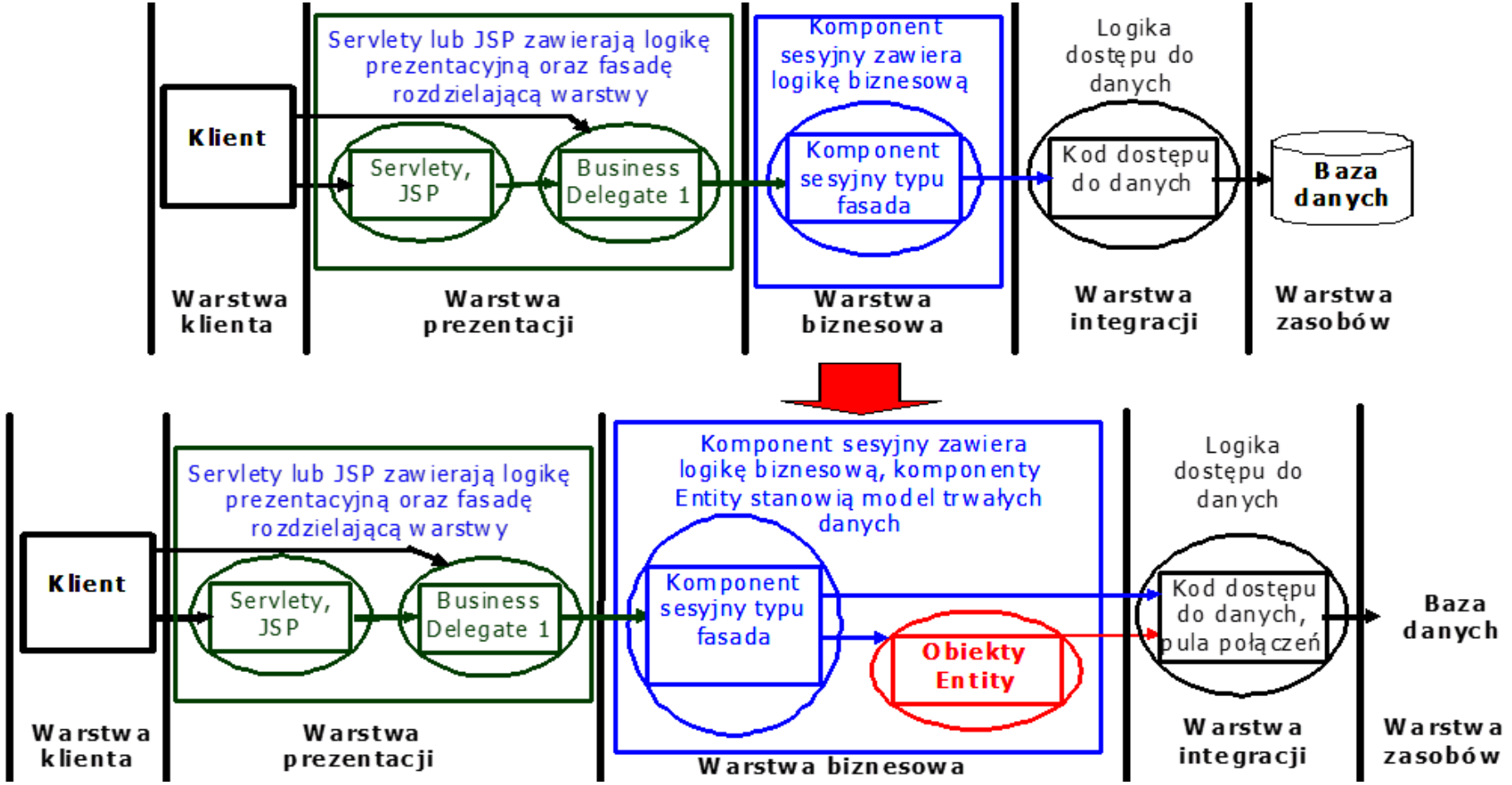
Refaktoryzacja architektury wielowarstwowej 1

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 hermetyzujace dostep do tych komponentów fasadowe komponenty sesyjne typu „Control”.** Komponenty Business Delegate typu „Control” hermetyzują dostęp do warstwy biznesowej z warstwy prezentacji.



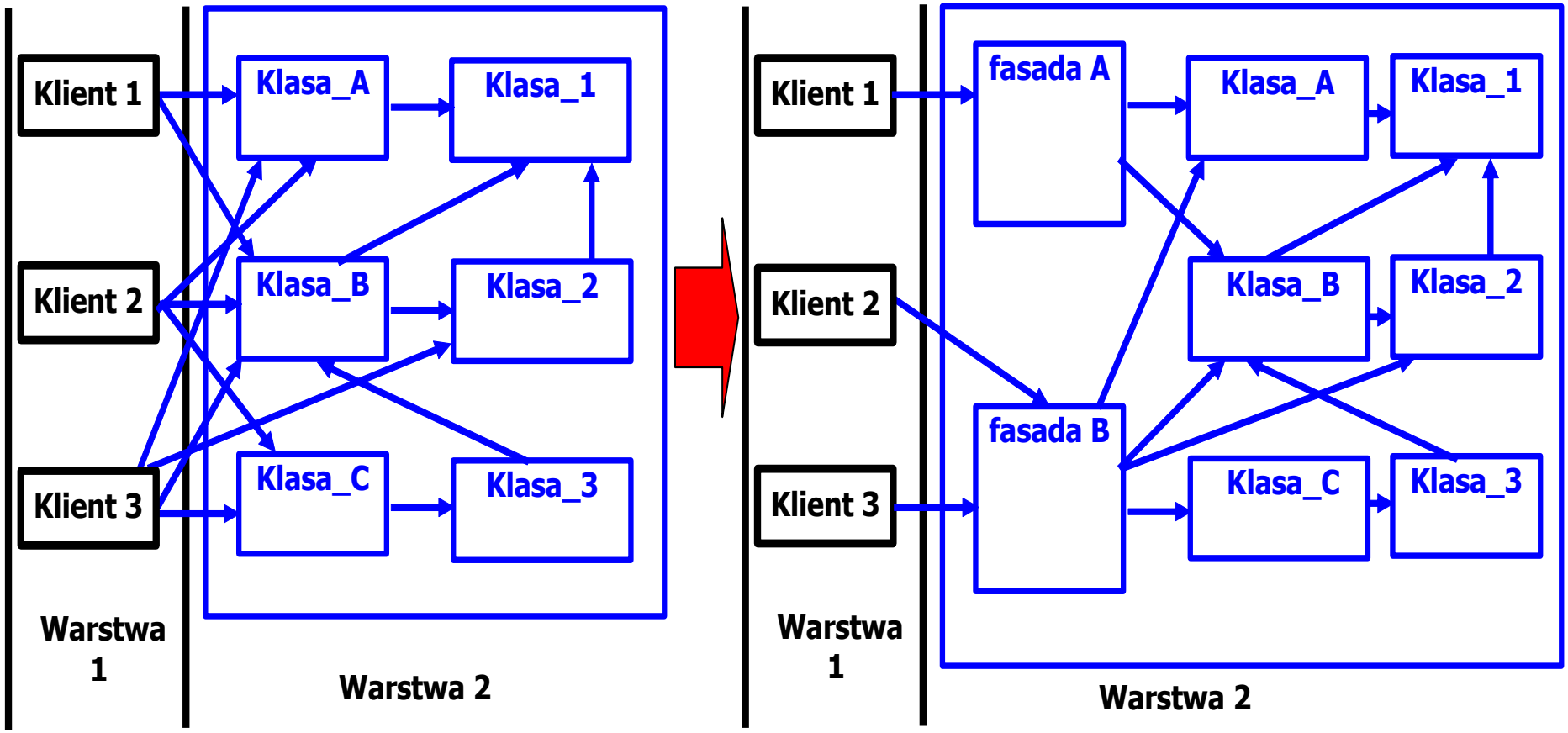
3. Wzorce projektowe

Wzorce projektowe

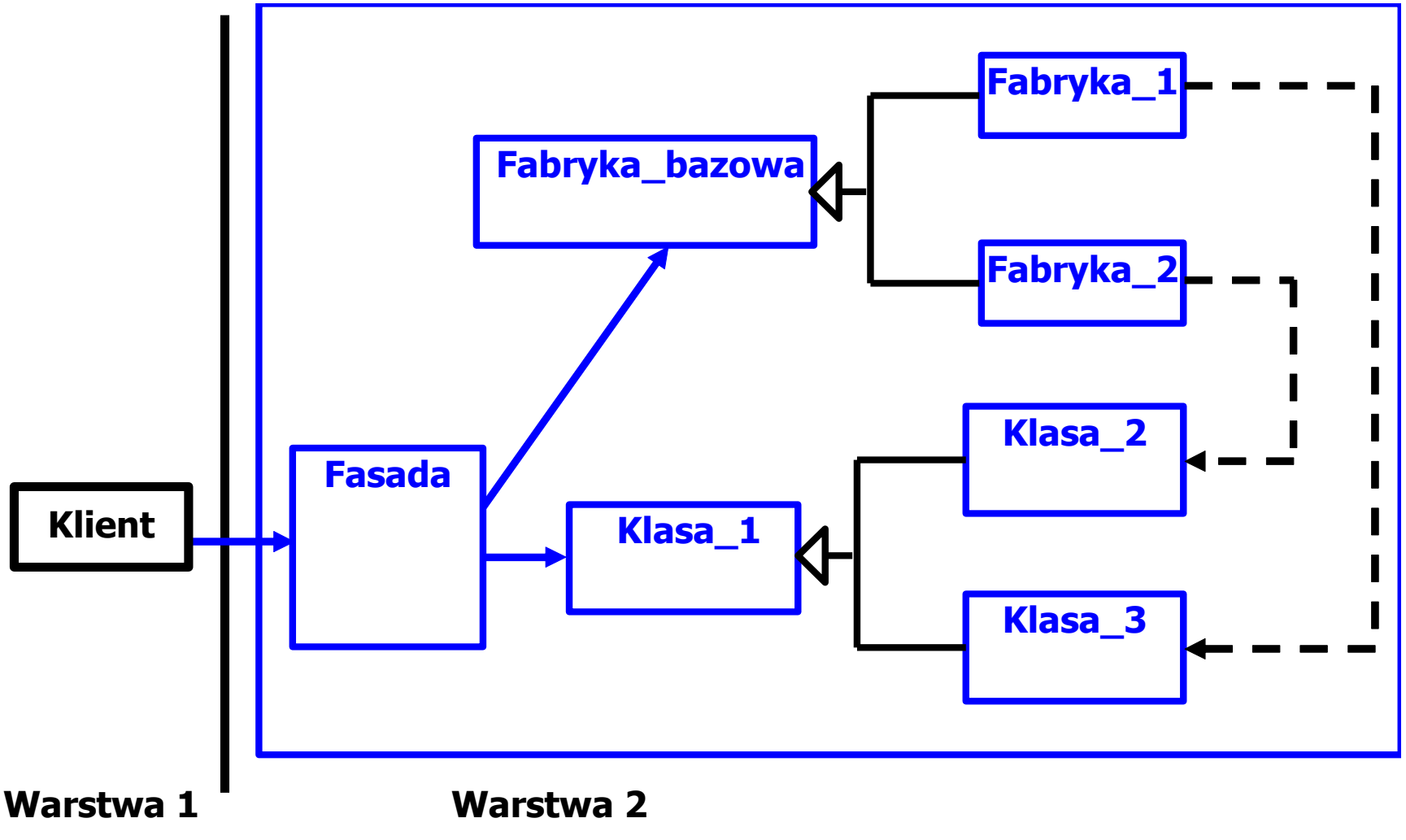
- Dobrze zbudowany system obiektowy jest pełen wzorców obiektowych
- Wzorzec to zwyczajowo przyjęte rozwiązanie typowego problemu w danym kontekście
- Strukturę wzorca przedstawia się w postaci diagramu klas
- Zachowanie się wzorca przedstawia się za pomocą diagramu sekwencji
- Wzorce projektowe: Wzorzec 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)**

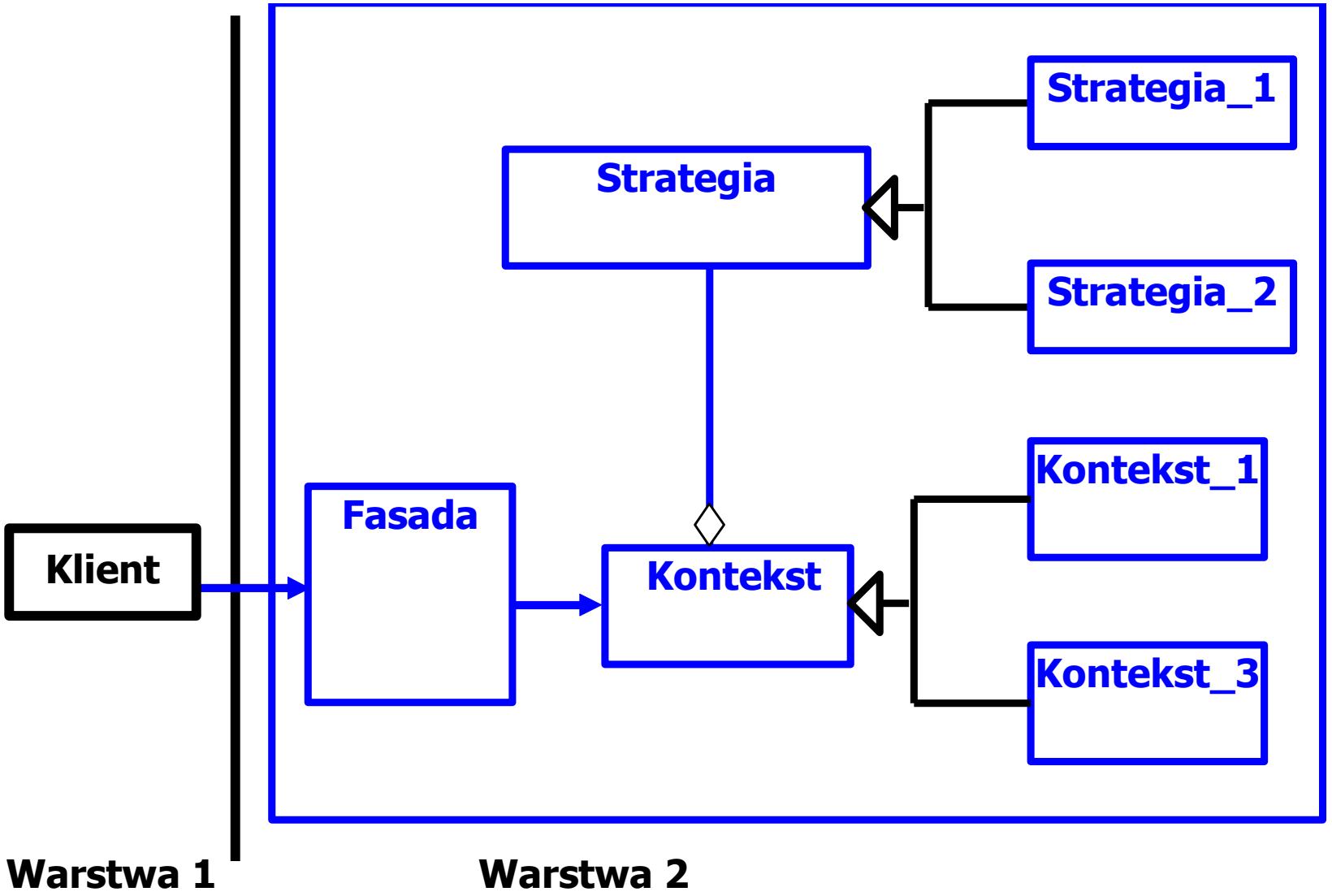
3.1. Wzorzec fasady (wzorzec strukturalny) – hermetyzacja logiki biznesowej



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 zawierajacą 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 bilioteki (bibliotekarz) może dodawać nowe tytuły i egzemplarze oraz je przeszukiwać, natomiast klient może jedynie przeszukiwać tyutły 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 tytuiłach oraz egzemplarzach. Zakłada się, że klientów jednocześnie przeglądającdających 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.

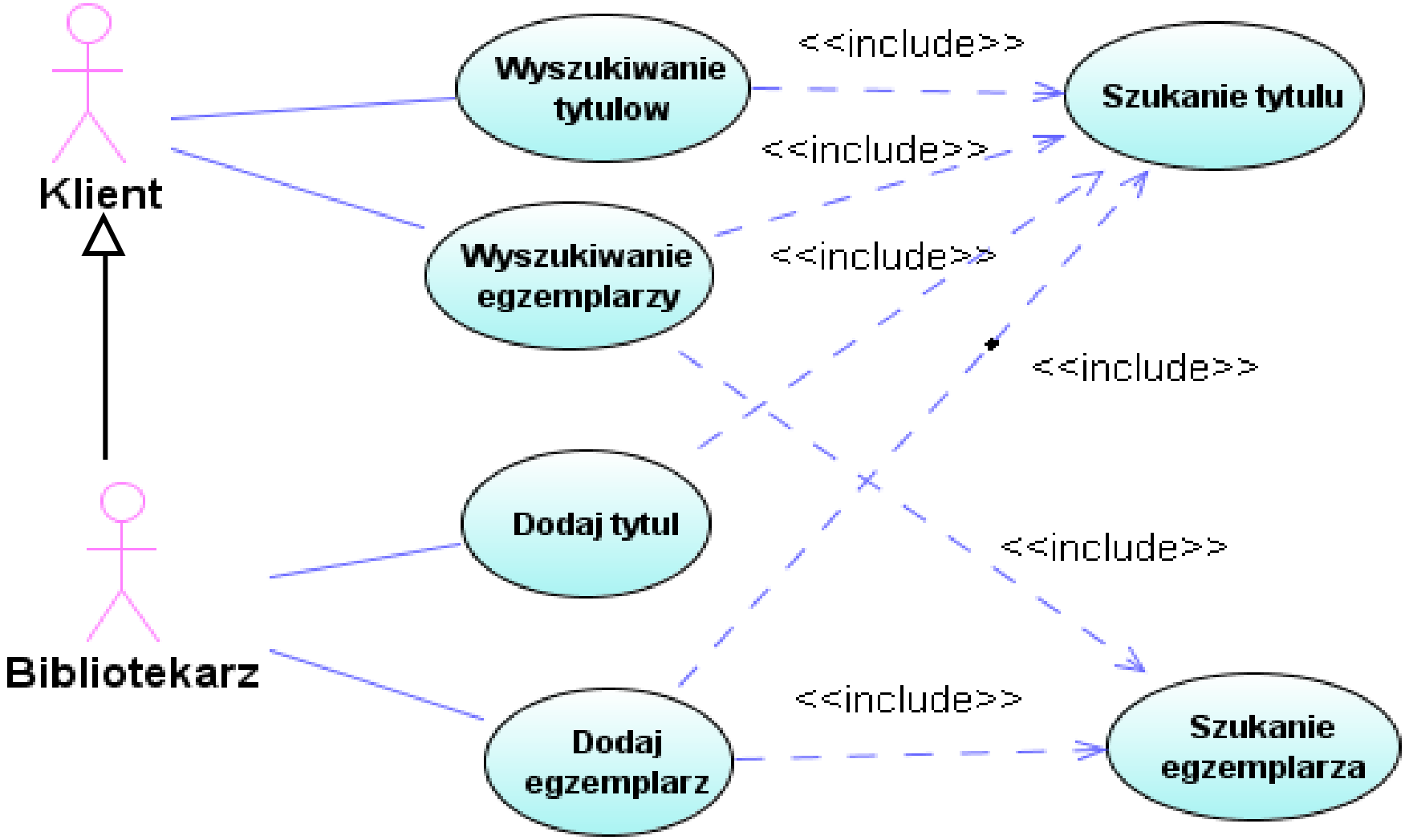
4.2. Lista wymagań funkcjonalnych całej aplikacji

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 kaseta, 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 kaseta oraz numer egzemplarza.
6. Zarówno egzemplarze typu książka lub kaseta, mogą być przeznaczane do wypożyczenia na okres umowny oraz na okres ściśle określony.

Lista wymagań niefunkcjonalnych

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”



AKTOR	OPIS	PRZYPADKI UŻYCIA
Bibliotekarz	*Bibliotekarz jest odpowiedzialny za utrzymywanie zasobów biblioteki (wstawianie i usuwanie: tytułów książek, egzemplarzy książek). Może on również przeszukiwać zasoby katalog tytułów i egzemplarzy książek*	• Dodaj tytul • Dodaj egzemplarz • Wyszukiwanie tytulow • Wyszukiwanie egzemplarzy
Klient	*Klient może jedynie przeszukiwać zasoby katalog tytułów i egzemplarzy książek*	• Wyszukiwanie tytulow • Wyszukiwanie egzemplarzy

PU Szukanie tytulu
OPIS
CEL: Poszukiwanie tytulu
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 tytulow
OPIS
CEL: Wyszukiwanie tytulow
WS (warunki wstępne): inicjalizacja przez uruchomienie programu (np. otwarcie strony WWW, start aplikacji)
WK (warunki końcowe): wyszukanie tytułu o podanym 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 tytul
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: tytul, autor, ISBN, wydawnictwo oraz jeśli jest to nagranie dźwiękowe, to nazwisko aktora lub informacja o istnieniu takiego tytułu
PRZEBIEG:
1. Należy podać atrybuty tytułu: tytul, 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 tytulu**. Należy sprawdzić, czy tytuł o podanych atrybutach juz 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
PU Szukanie tytulu **PU Wyszukiwanie tytulow** **PU Dodaj tytul**	klasa typu „Entity”: TTytul_ksiazki
PU Szukanie egzemplarza **PU Wyszukiwanie egzemplarzy** **PU Dodaj egzemplarz**	klasy typu „Entity”: TTytul_ksiazki (zawiera atrybuty tytułu, posiada kiążki – wstawia i wyszukuje je), TEgzemplarz (książka)

4.4.2. Analiza zmienności - identyfikacja podklas

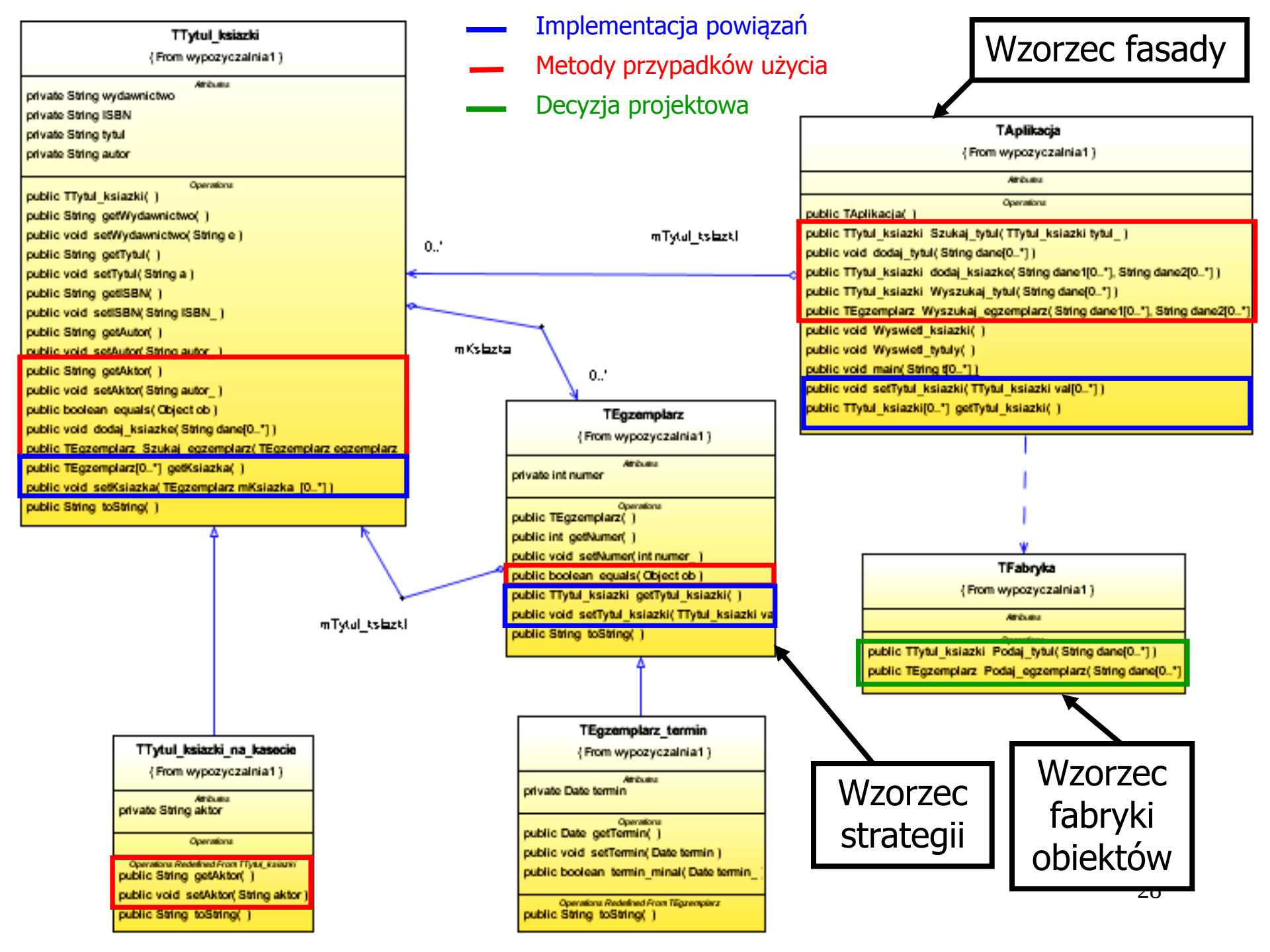
Przypadki użycia	Wykryte podklasy
PU Szukanie tytulu **PU Wyszukiwanie tytulow** **PU Dodaj tytul**	klasa TTytul_ksiazki_na_kasecie typu „Entity”, która dziedziczy od klasy TTytul_ksiazki
PU Szukanie egzemplarza **PU Wyszukiwanie egzemplarzy** **PU Dodaj egzemplarz**	Wyróżniono egzemplarze zwykłe typu TEgzemplarz, oraz egzemplarze TEgzemplarz_termin z dodatkowo oznaczonym terminem oddania - rozróżniane w ramach danego tytułu książki zwyklej (TTytul_ksiazki) lub nagranej w postaci dźwiękowej (TTytul_ksiazki_na_kasecie) za pomocą numeru

4.4.3. Analiza wspólności i zmienności - identyfikacja typów relacji

- Zależność między obiektami typu TTytul_ksiazki oraz TEgzemplarz są w relacji 1 do 0..*. Związek ten dziedziczą obiekty typu TTytul_ksiazki_na_kasecie.
- Związek 0..* do 1 między obiektami typu TEgzemplarz oraz TTytul_ksiazki 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 klasęTAplikacja 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.


Implementacja powiązań
Metody przypadków użycia
Decyzja projektowa
Wzorzec fasady
TTytul_ksiazki
{ From wypozyczalnia1 }
Attributes
private String wydawnictwo
private String ISBN
private String tytul
private String autor
Operations
public TTytul_ksiazki( )
public String getWydawnictwo( )
public void setWydawnictwo(String e)
public String getTytul( )
public void setTytul(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 autor_)
public boolean equals(Object ob)
public void dodaj_ksiazke(String dane[0..*])
public TEgzemplarz Szukaj_egzemplarz(TEgzemplarz egzemplarz
public TEgzemplarz[0..*] getKsiazka( )
public void setKsiazka(TEgzemplarz mKsiazka [0..*])
public String toString( )
TAplikacja
{ From wypozyczalnia1 }
Attributes
Operations
public TAplikacja( )
public TTytul_ksiazki Szukaj_tytul(TTytul_ksiazki tytul_)
public void dodaj_tytul(String dane[0..*])
public TTytul_ksiazki dodaj_ksiazke(String dane1[0..*], String dane2[0..*])
public TTytul_ksiazki Wyszukaj_tytul(String dane[0..*])
public TEgzemplarz Wyszukaj_egzemplarz(String dane1[0..*], String dane2[0..*]
public void Wyswietl_ksiazki( )
public void Wyswietl_tytuly( )
public void main(String t[0..*])
public void setTytul_ksiazki(TTytul_ksiazki val[0..*])
public TTytul_ksiazki[0..*] getTytul_ksiazki( )
mTytul_ksiazki
0..*
mKsiazka
0..*
TEgzemplarz
{ From wypozyczalnia1 }
Attributes
private int numer
Operations
public TEgzemplarz( )
public int getNumer( )
public void setNumer(int numer_)
public boolean equals(Object ob)
public TTytul_ksiazki getTytul_ksiazki( )
public void setTytul_ksiazki(TTytul_ksiazki va
public String toString( )
mTytul_ksiazki
TFabryka
{ From wypozyczalnia1 }
Attributes
public TTytul_ksiazki Podaj_tytul(String dane[0..*])
public TEgzemplarz Podaj_egzemplarz(String dane[0..*]
TTytul_ksiazki_na_kasecie
{ From wypozyczalnia1 }
Attributes
private String aktor
Operations
Operations Redefined From TTytul_ksiazki
public String getAktor( )
public void setAktor(String aktor)
public String toString( )
TEgzemplarz_termin
{ From wypozyczalnia1 }
Attributes
private Date termin
Operations
public Date getTermin( )
public void setTermin(Date termin)
public boolean termin_minal(Date termin_
Operations Redefined From TEgzemplarz
public String toString( )
Wzorzec strategii
Wzorzec fabryki obiektów

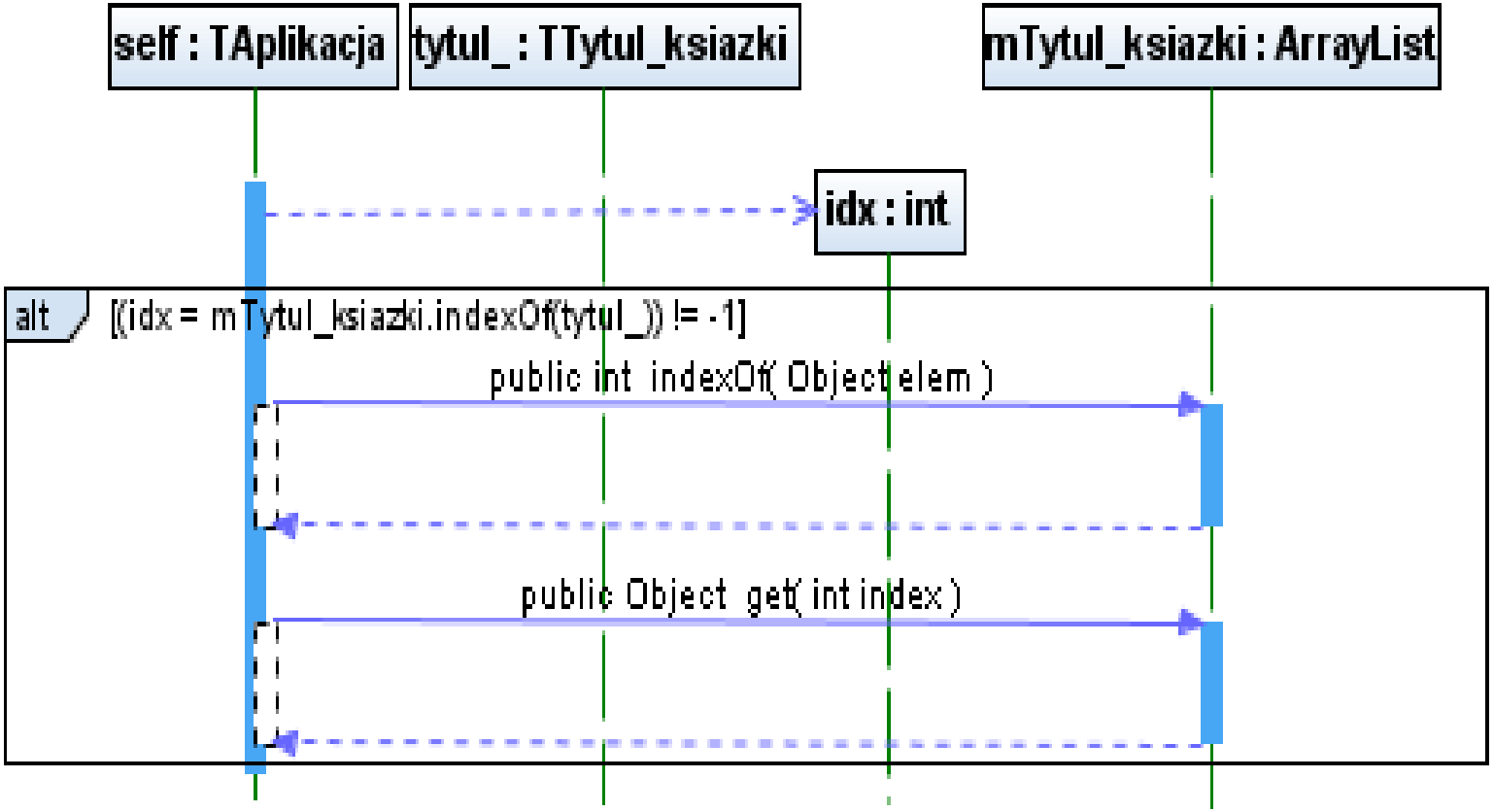
4.5. Model projektowy warstwy biznesowej oparty na diagramie klas i diagramie sekwencji tworzony metodą iteracyjno-rozwojową sterowany realizacją przypadków użycia

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

Projekt przypadku użycia
„ **Szukanie tytulu**”
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 tytulu

(TTytul_ksiazki TAplikacja::Szukaj_tytul(TTytul_ksiazki tytul_))

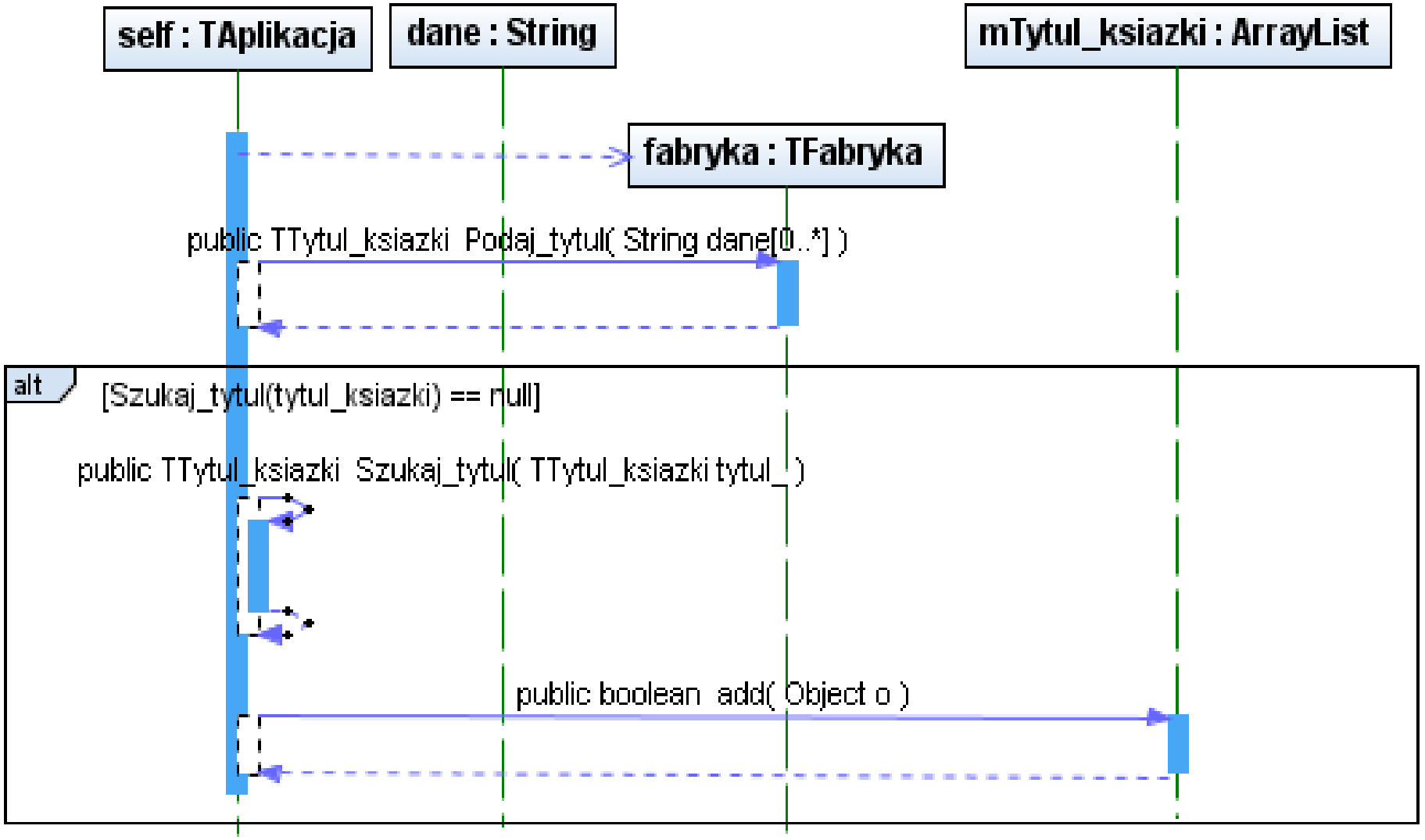


boolean TTytul_ksiazki::equals(Object ob)

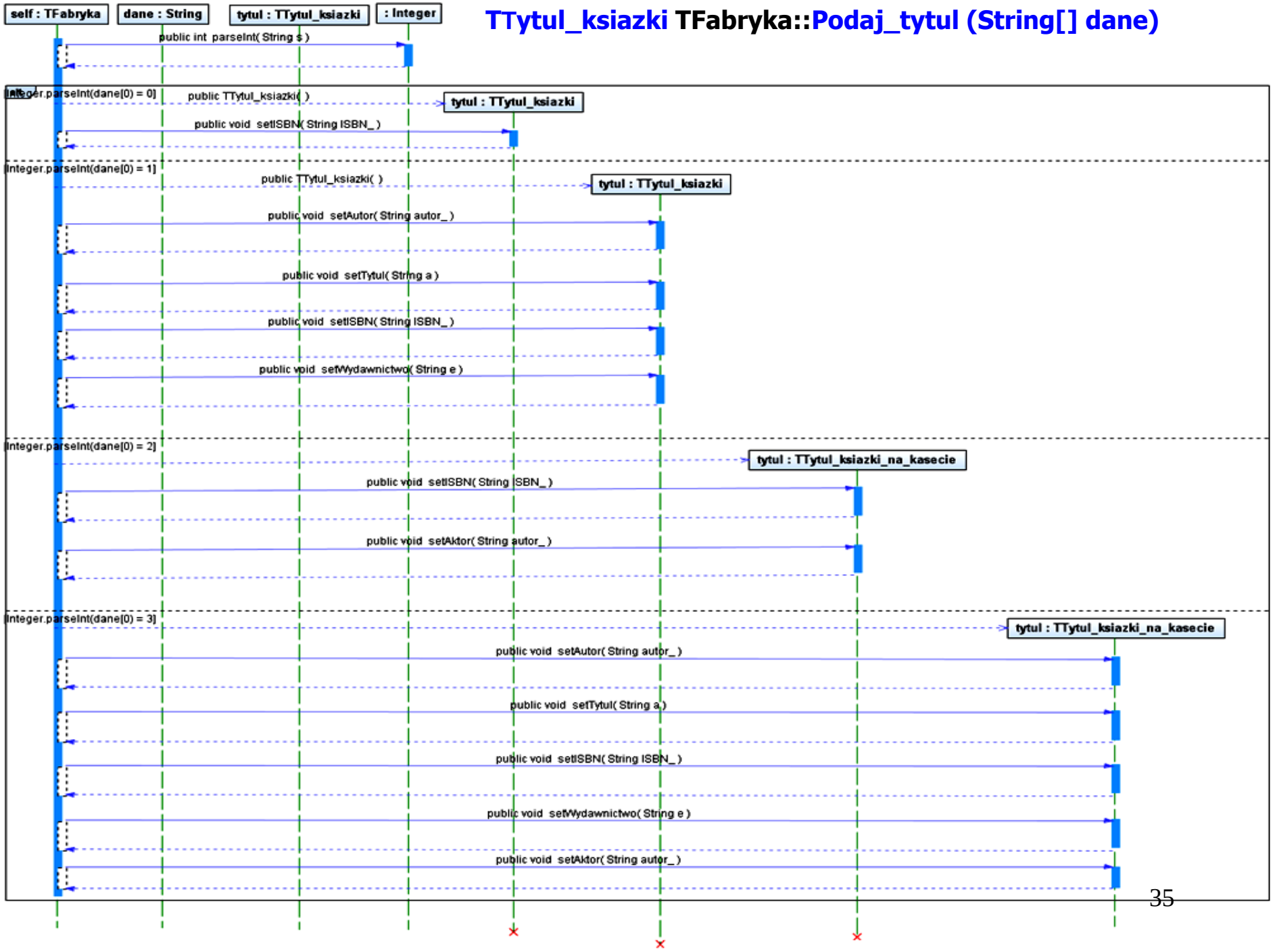
Projekt przypadku użycia
„ **Dodaj tytul**”
za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.
Definiowanie kodu metod realizujących przypadek użycia na podstawie diagramów sekwencji

(2) Dodaj tytul

void TAplikacja::dodaj_tytul(String dane[])



TTytul_ksiazki TFabryka::Podaj_tytul (String[] dane)



```
C:\Users\kruczkiewicz>java -jar "E:\Dydaktyka\Wzorce\Wypozyczalnia1\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]
```

// ten kod powinien działać po uzupełnieniu kodu dla wskazanych klas
// biorących udział w wykonanych przypadkach użycia oraz
// po wykonaniu metody **toString()** w tych klasach (**TTytul_ksiazki** oraz **TTytul_ksiazki_na_kasecie**) i **getTytul_ksiazki()** w klasie **TAplikacja**

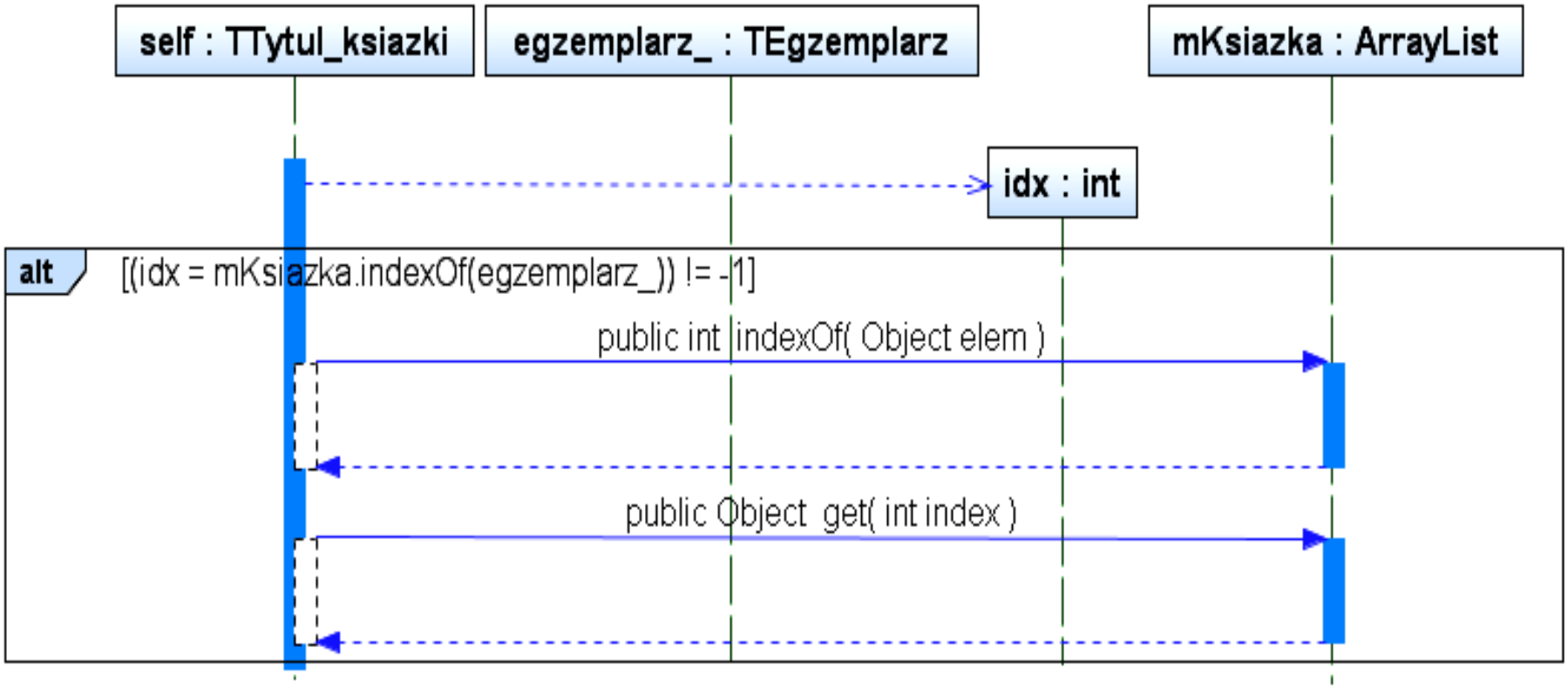
```
public static void main(String t[]) // your code here
   { TAplikacja ap = new TAplikacja();
     String t1[] = {"1", "1", "1", "1", "1"}; //t1, t2,  t3 – tablice łańcuchów  do tworzenia tytułu książki zwyklej – pierwszy łańcuch
     String t2[] = {"1", "2", "2", "2", "2"}; // jest informacją dla 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);
     }
```

```
Tytul: 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

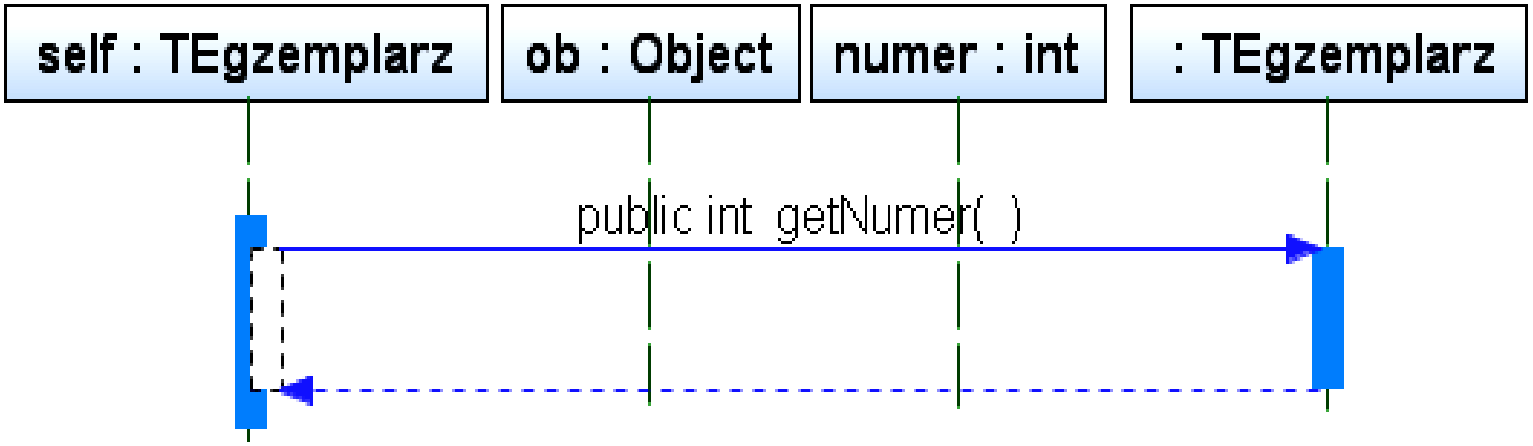
(3) Szukaj egzemplarz

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;
 }
```

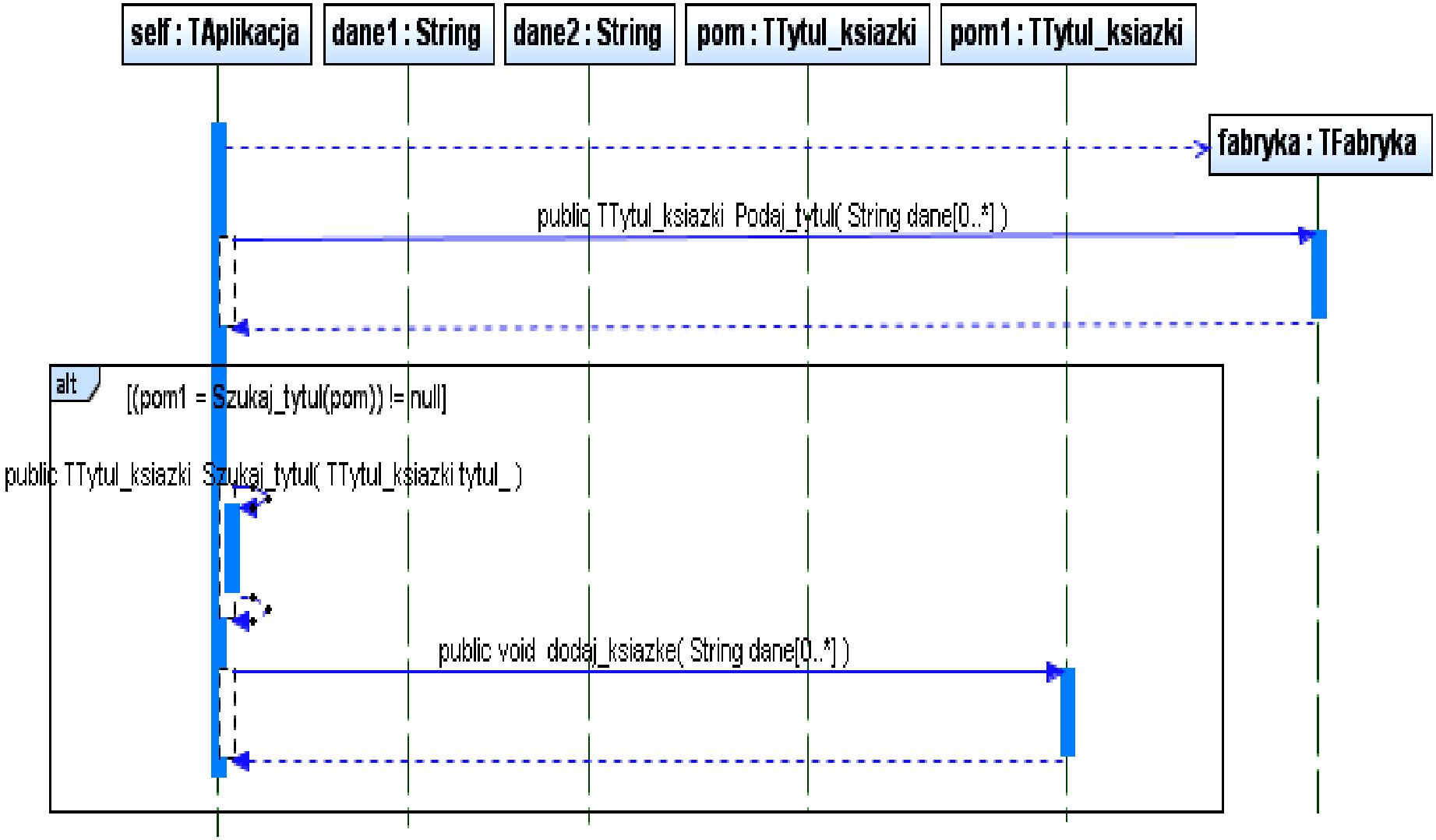
boolean TEgzemplarz::equals(Object ob)



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

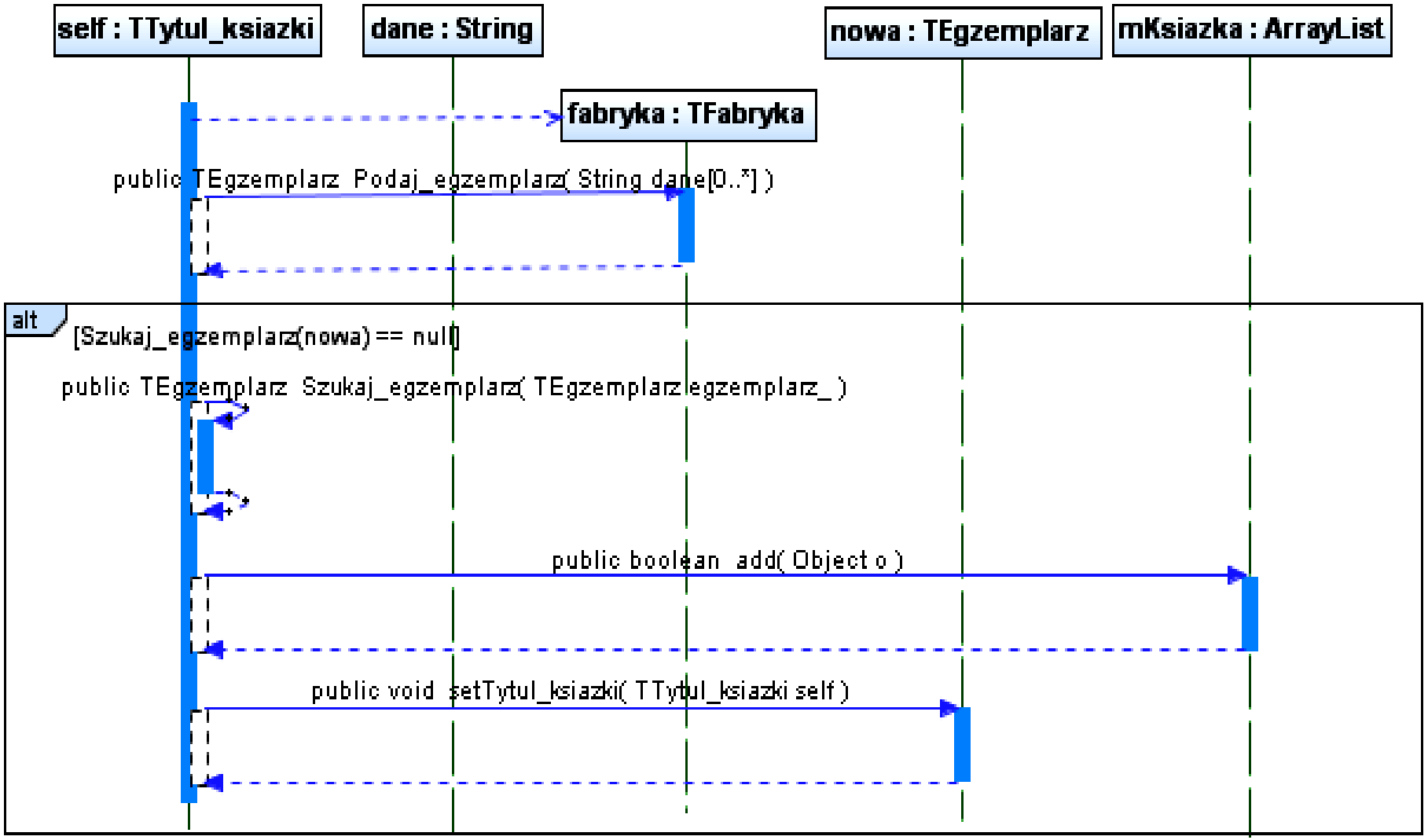
(4) Dodaj egzemplarz

TTytul_ksiazki TAplikacja::dodaj_ksiazke(String dane1[], String dane2[]))



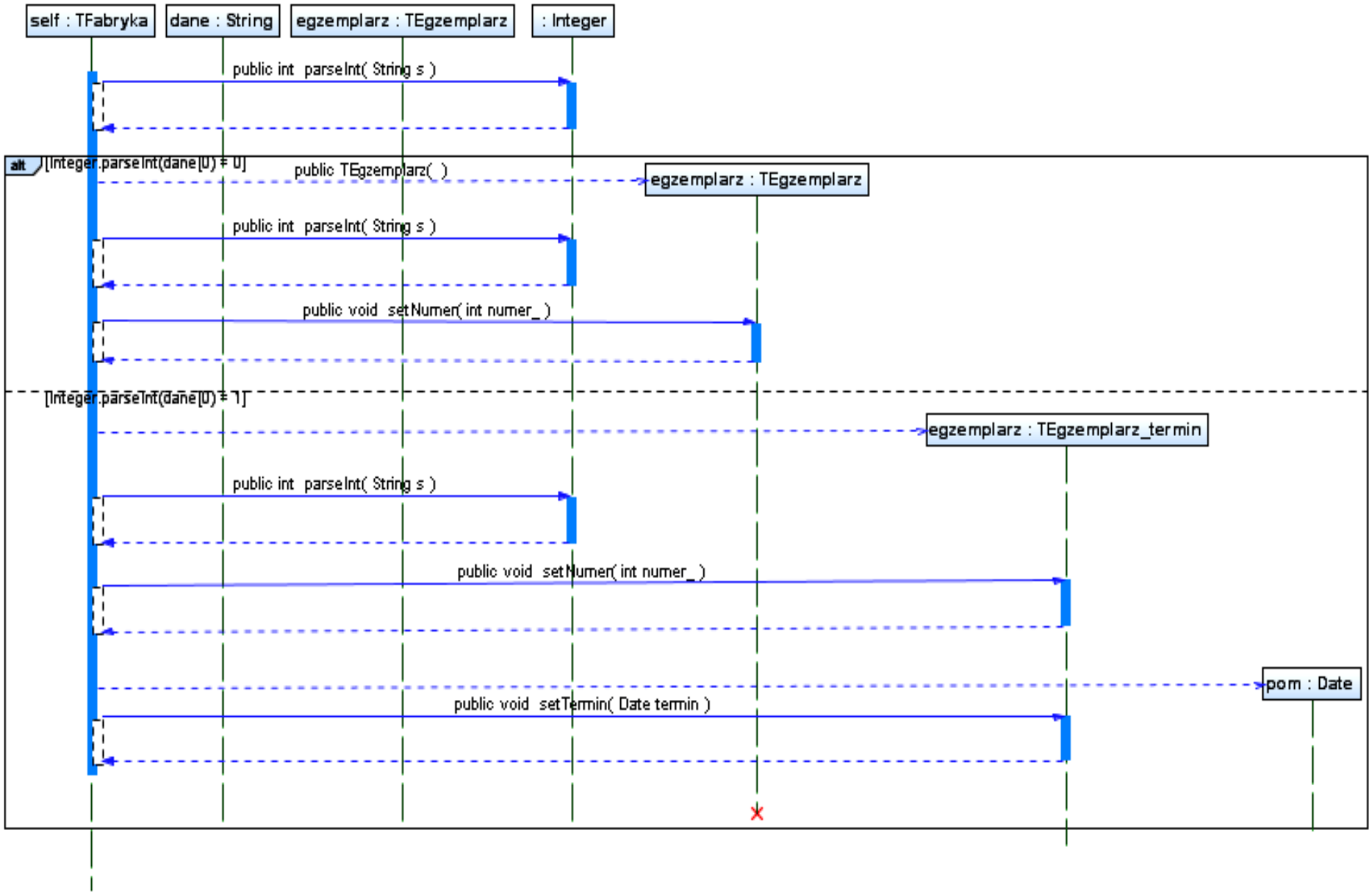
```
//TAplikacja

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

void TTytul_ksiazki::dodaj_ksiazke(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)

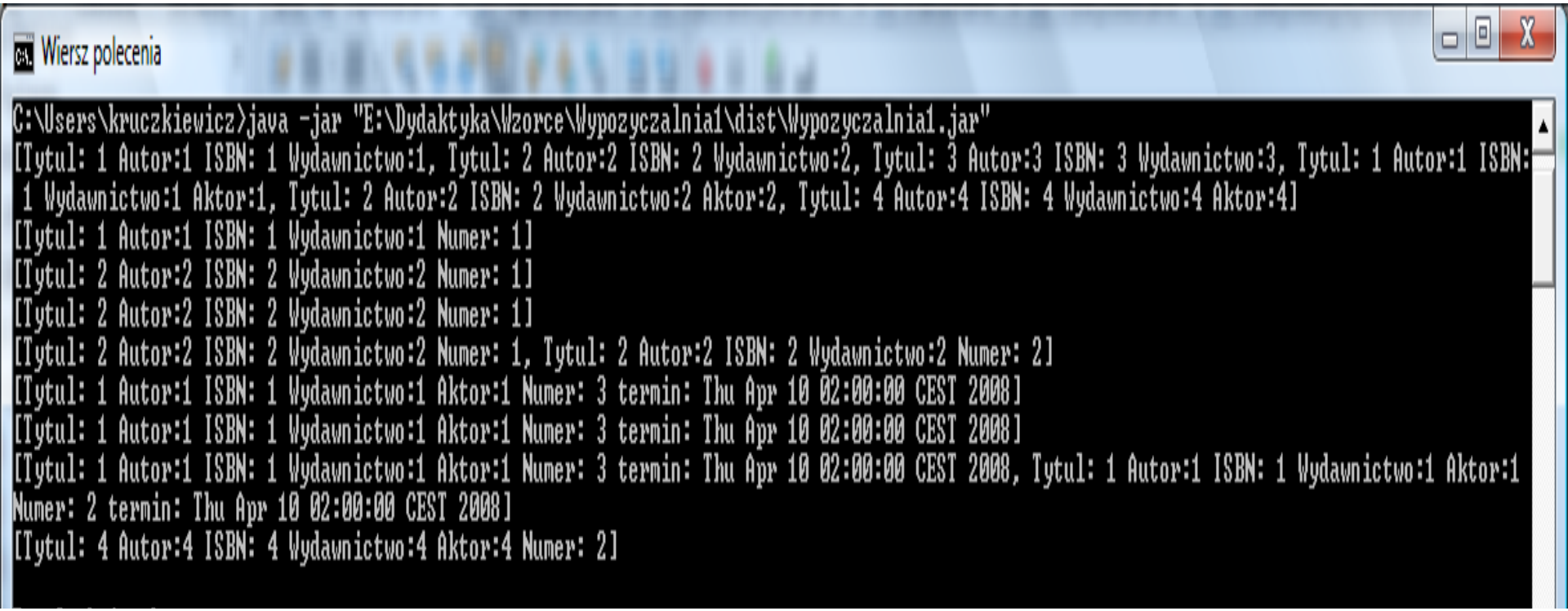


Proponowany kod funkcji main w klasie fasadowej TAplikacja

```
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 zwyklej – 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 zwyklej  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 - 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_ksiazki pom = ap.dodaj_ksiazke(d1, tr1); //dodana
if (pom != null) {        System.out.println(pom.getKsiazka().toString());      }

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

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

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

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

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

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

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

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

}
```

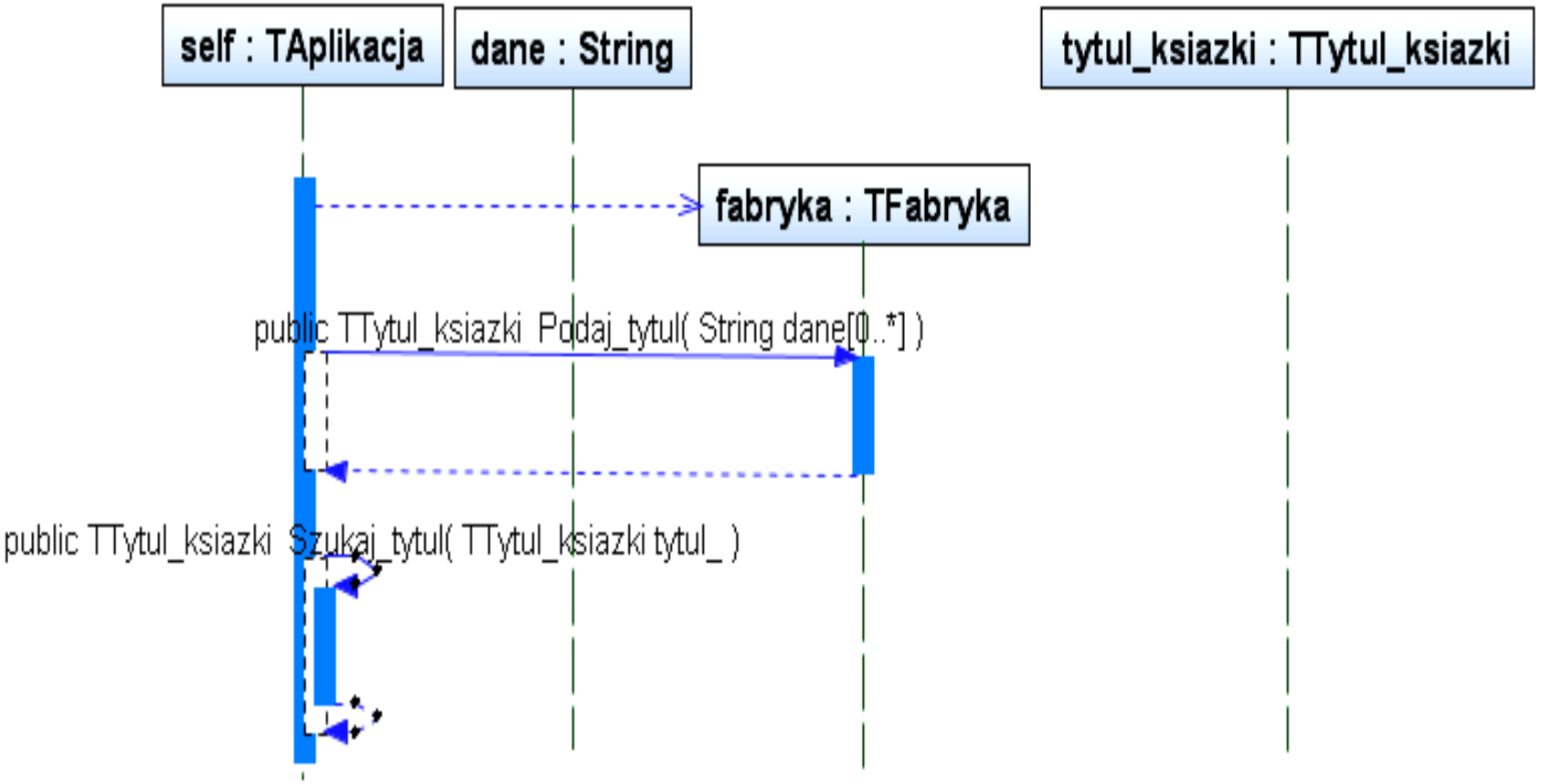
Tak może działać aplikacja po wykonaniu poszczególnych przypadków użycia



Projekt przypadku użycia
„ **Wyszukiwanie tytulow**”
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 tytulow

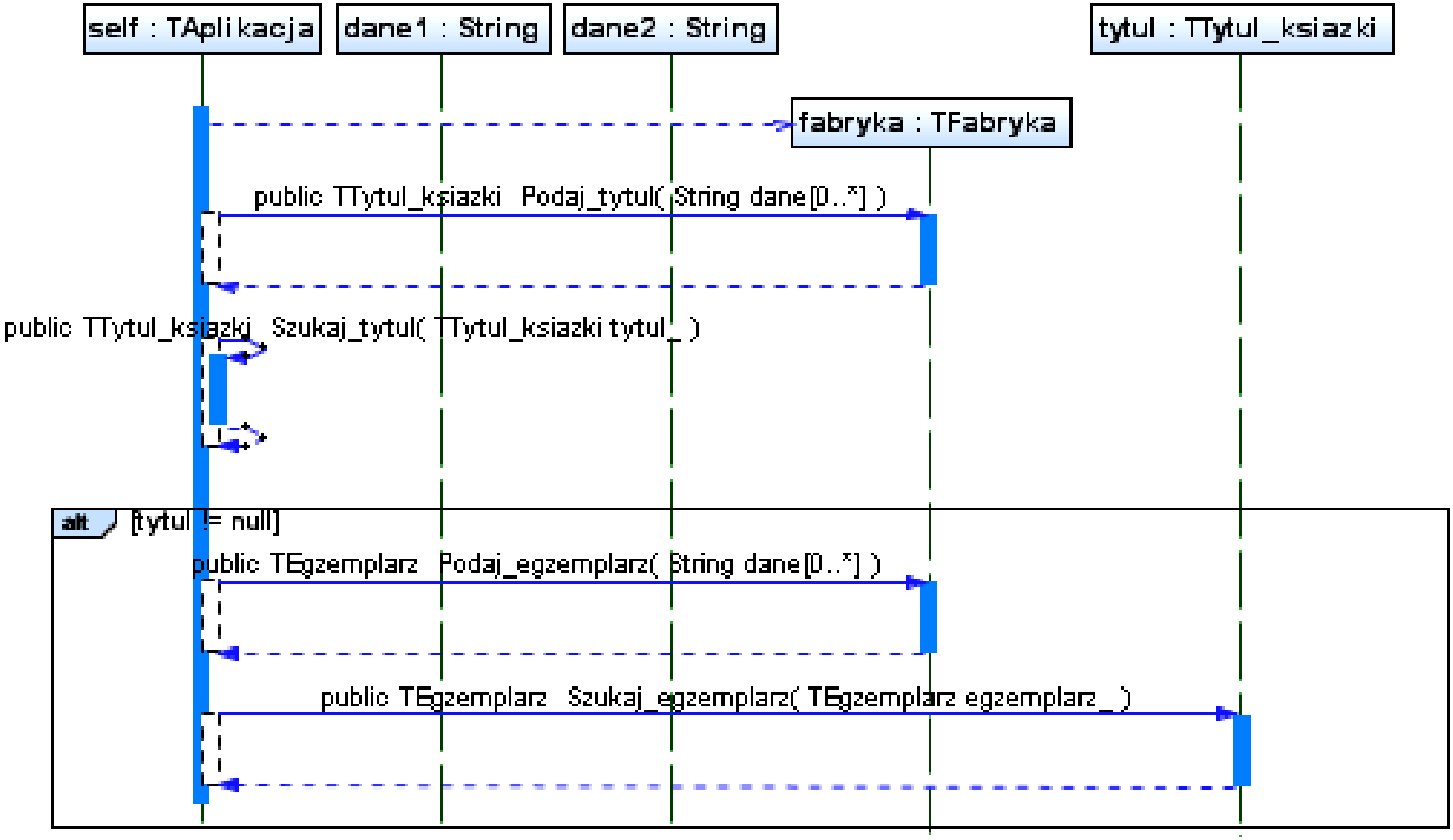
TTytul_ksiazki TAplikacja::Wyszukaj_tytul (String[] dane)



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 egzemplarzy

TEgzemplarz TAplikacja::Wyszukaj_egzemplarz (String[] dane1, String[] dane2)



```
package wypozyczalnia1;
import java.util.ArrayList;

public class TAplikacja {
   private ArrayList<TTytul_ksiazki> mTytul_ksiazki = new ArrayList<TTytul_ksiazki>();

   public TAplikacja ()                                                    {/* */}
   public static void main (String[] t)                                   {/* */}
   public ArrayList<TTytul_ksiazki> getTytul_ksiazki ()                    {/* */}
   public void setTytul_ksiazki (ArrayList<TTytul_ksiazki> val)            {/* */}

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

```
package wypozyczalnia1;

public class TEgzemplarz {
   private int numer;
   private TTytul_ksiazki mTytul_ksiazki;

   public TEgzemplarz ()                                {/* */}
   public int getNumer ()                               {/* */}
   public void setNumer (int numer_)                    {/* */}
   public TTytul_ksiazki getTytul_ksiazki ()            {/* */}
   public void setTytul_ksiazki (TTytul_ksiazki val)    {/* */}
   public boolean equals (Object ob)                    {/* */}
   public String toString ()                            {/* */}
}
```

```
package wypozyczalnia1;
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 wypozyczalnia1;

public class TFabryka {

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

```
package wypozyczalnia1;
import java.util.ArrayList;

public class TTytul_ksiazki {
   private String wydawnictwo;
   private String ISBN;
   private String tytul;
   private String autor;
   private ArrayList<TEgzemplarz> mKsiazka = new java.util.ArrayList<TEgzemplarz>();
   public TTytul_ksiazki ()                                          {/* */      }
   public String getWydawnictwo ()                                   {/* */      }
   public void setWydawnictwo (String e)                             {/* */      }
   public String getTytul ()                                         {/* */      }
   public void setTytul (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 autor_)                              {/* */      }
   public ArrayList<TEgzemplarz> getKsiazka ()                       {/* */      }
   public void setKsiazka (ArrayList<TEgzemplarz> mKsiazka_)         {/* */      }
   public String toString ()                                         {/* */      }
   public boolean equals (Object ob)                                 {/* */      }
   public void dodaj_ksiazke (String[] dane)                         {/* */      }
   public TEgzemplarz Szukaj_egzemplarz (TEgzemplarz egzemplarz_)    {/* */      }
}
```

```
package wypozyczalnia1;
public class TTytul_ksiazki_na_kasecie extends TTytul_ksiazki {
   private String aktor;
   public String getAktor ()                {/* */      }
   public void setAktor (String aktor)      {/* */      }
   public String toString ()                {/* */      }
}
```

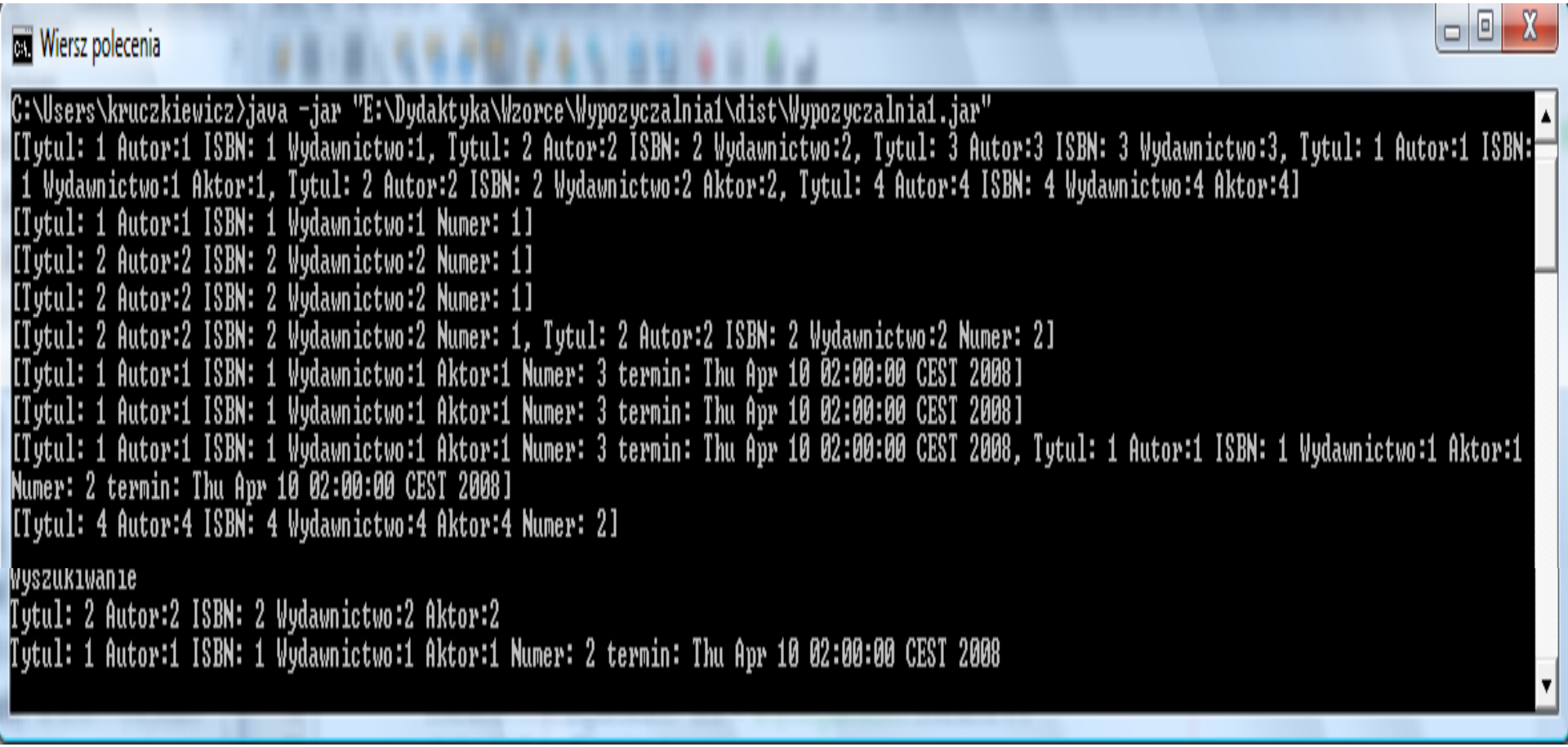
Proponowany kod funkcji main w klasie fasadowej TAplikacja

```
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 zwyklej – 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 zwyklej  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-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
```

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

```
TTytul_ksiazki pom = ap.dodaj_ksiazke(d1, tr1);
 if (pom != null) {        System.out.println(pom.getKsiazka().toString());      }
 pom = ap.dodaj_ksiazke(d2, tr1);
 if (pom != null) {        System.out.println(pom.getKsiazka().toString());      }
 pom = ap.dodaj_ksiazke(d2, tr1);
 if (pom != null) {        System.out.println(pom.getKsiazka().toString());      }
 pom = ap.dodaj_ksiazke(d2, tr2);
 if (pom != null) {        System.out.println(pom.getKsiazka().toString());      }
 pom = ap.dodaj_ksiazke(d3, tr2);
 if (pom != null) {        System.out.println(pom.getKsiazka().toString());      }
 pom = ap.dodaj_ksiazke(d4, tr3);
 if (pom != null) {        System.out.println(pom.getKsiazka().toString());      }
 pom = ap.dodaj_ksiazke(d4, tr3);
 if (pom != null) {        System.out.println(pom.getKsiazka().toString());      }
 pom = ap.dodaj_ksiazke(d4, tr4);
 if (pom != null) {        System.out.println(pom.getKsiazka().toString());      }
 pom = ap.dodaj_ksiazke(d5, tr2);
 if (pom != null) {        System.out.println(pom.getKsiazka().toString());      }

 System.out.println("Wyszukiwanie");
 System.out.println(ap.Wyszukaj_tytul(t5).toString());
 System.out.println(ap.Wyszukaj_egzemplarz(d4, tr4).toString()); */
}
```

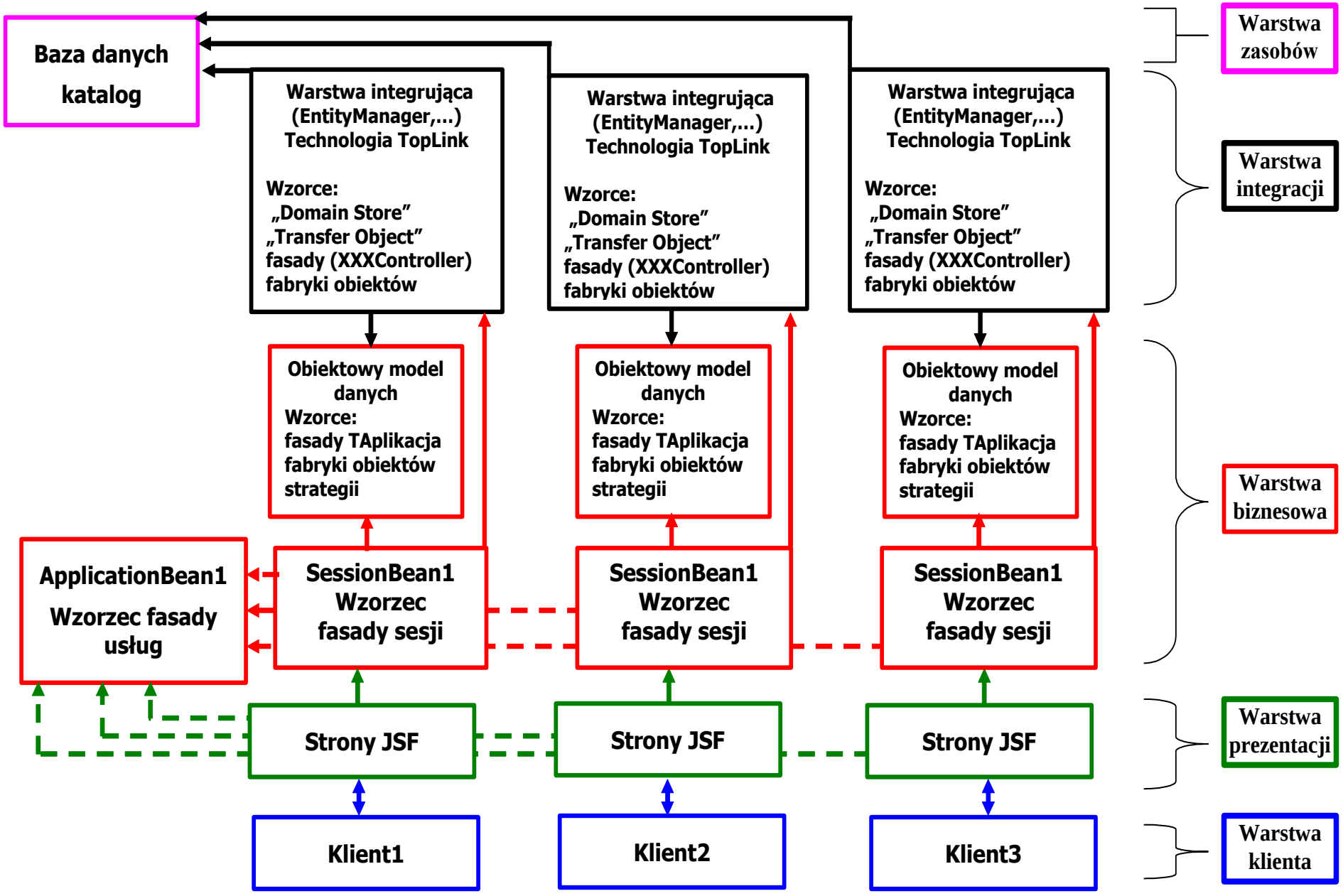
Tak może działać aplikacja po wykonaniu poszczególnych przypadków użycia



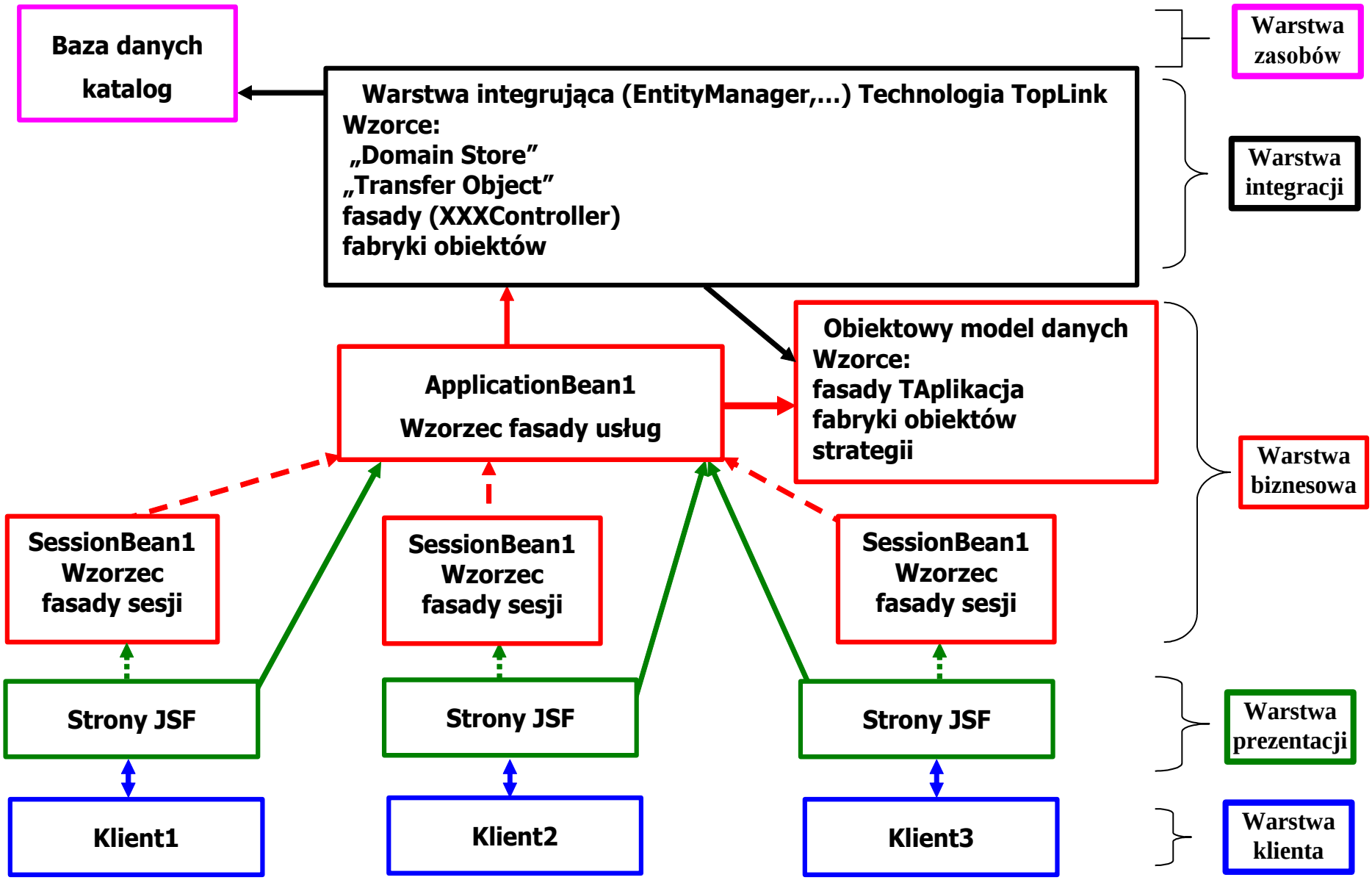
5. Implementacja wielowarstwowego systemu informatycznego

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



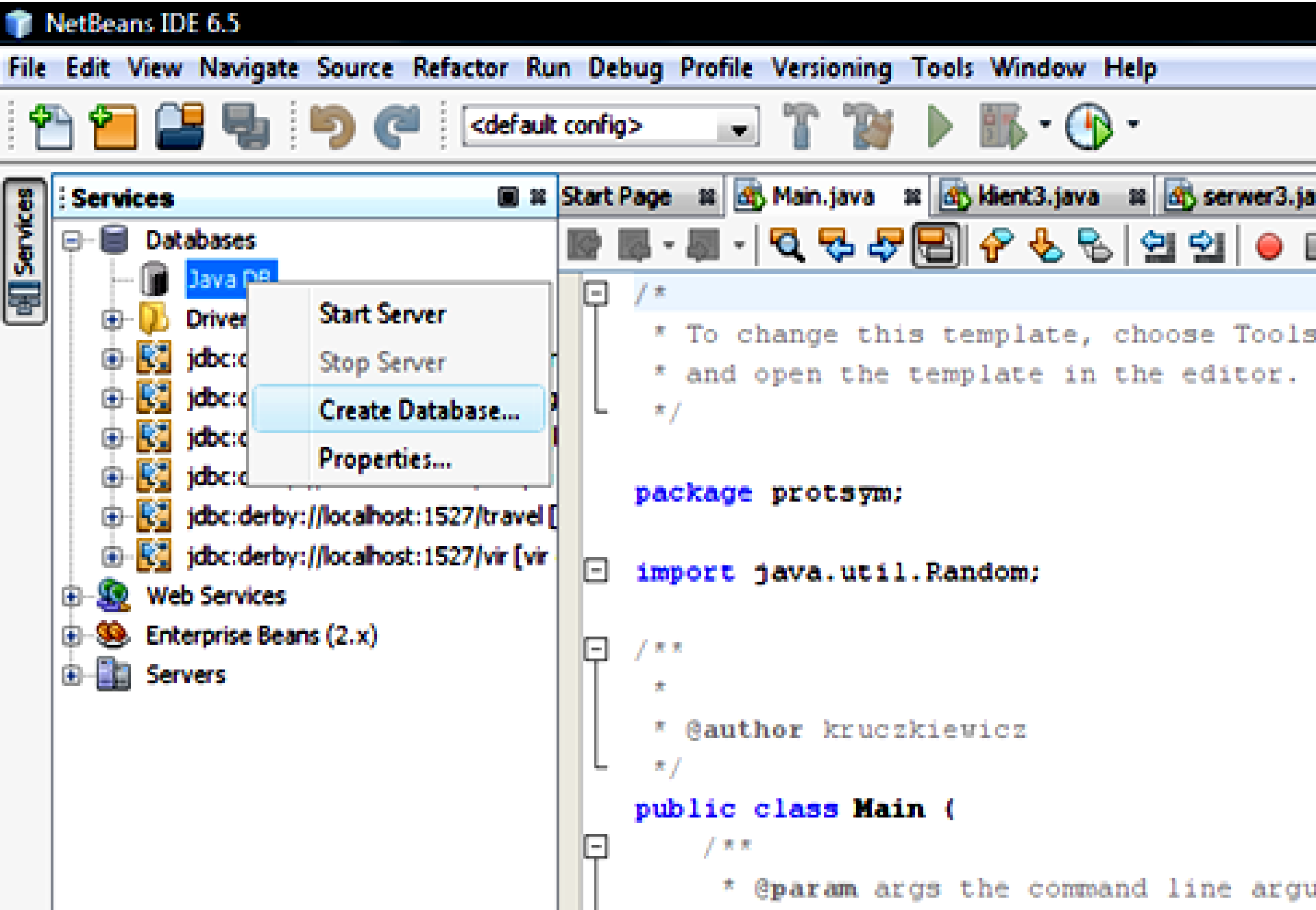
Architektura aplikacji pięciowarstwowej Java EE 5.0 Visual Web Java Server Faces - linie przerywane oznaczają powiązania niewykorzystane w aplikacji



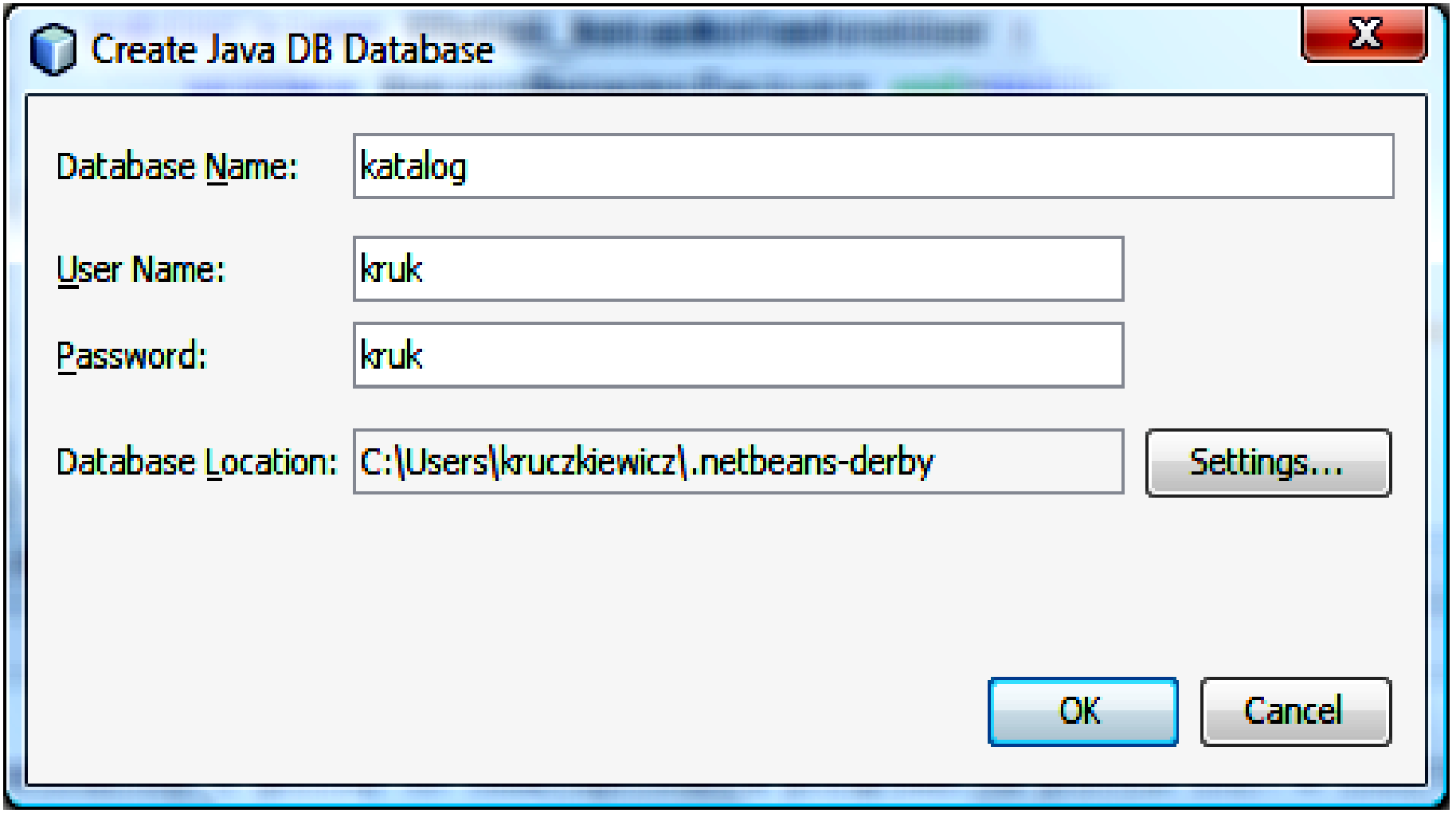
6. Tworzenie bazy danych w systemie baz danych Derby

- przykład **warstwy zasobów** (EIS)

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

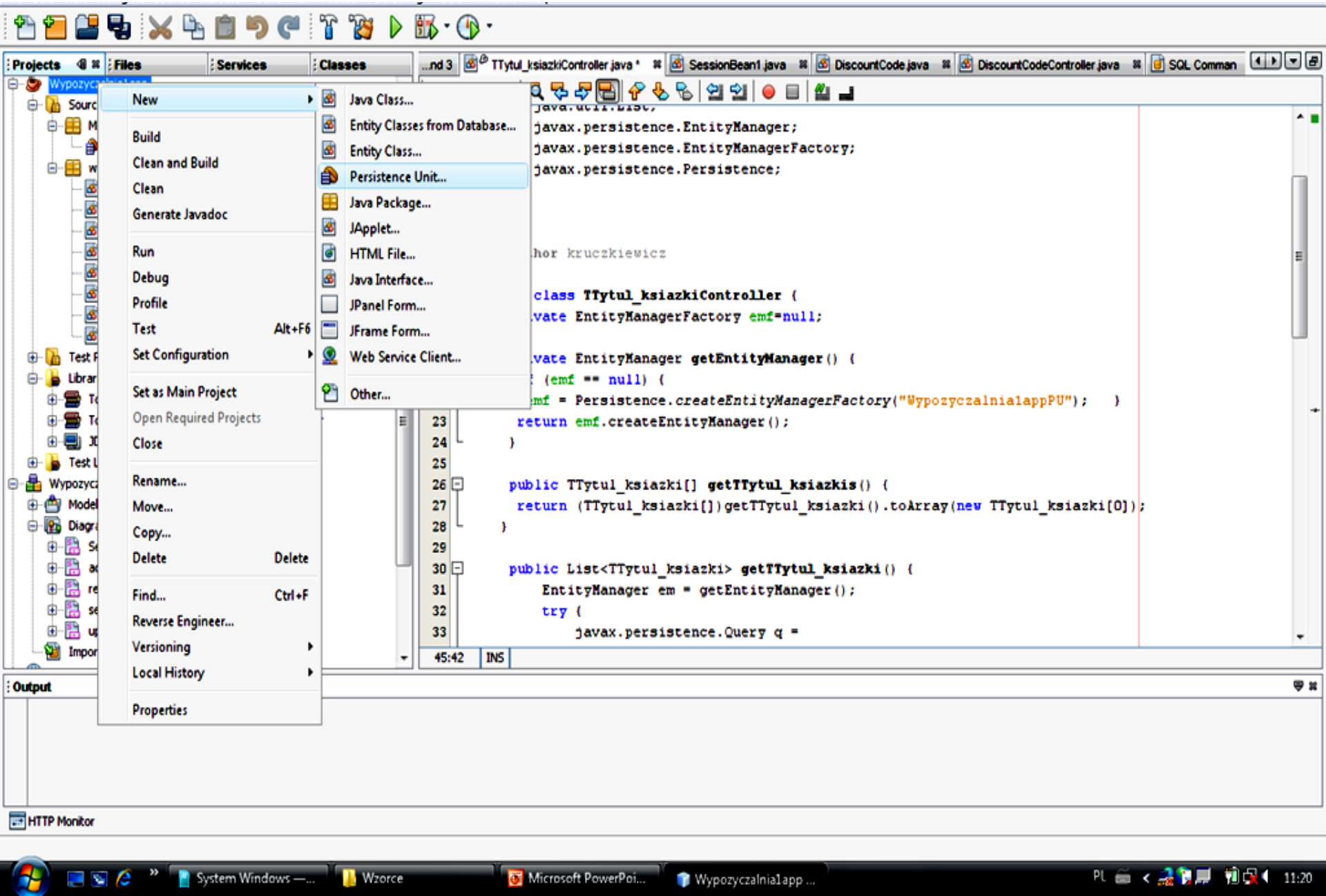


Zakładanie pustej bazy danych **katalog** w systemie baz danych Derby (2)

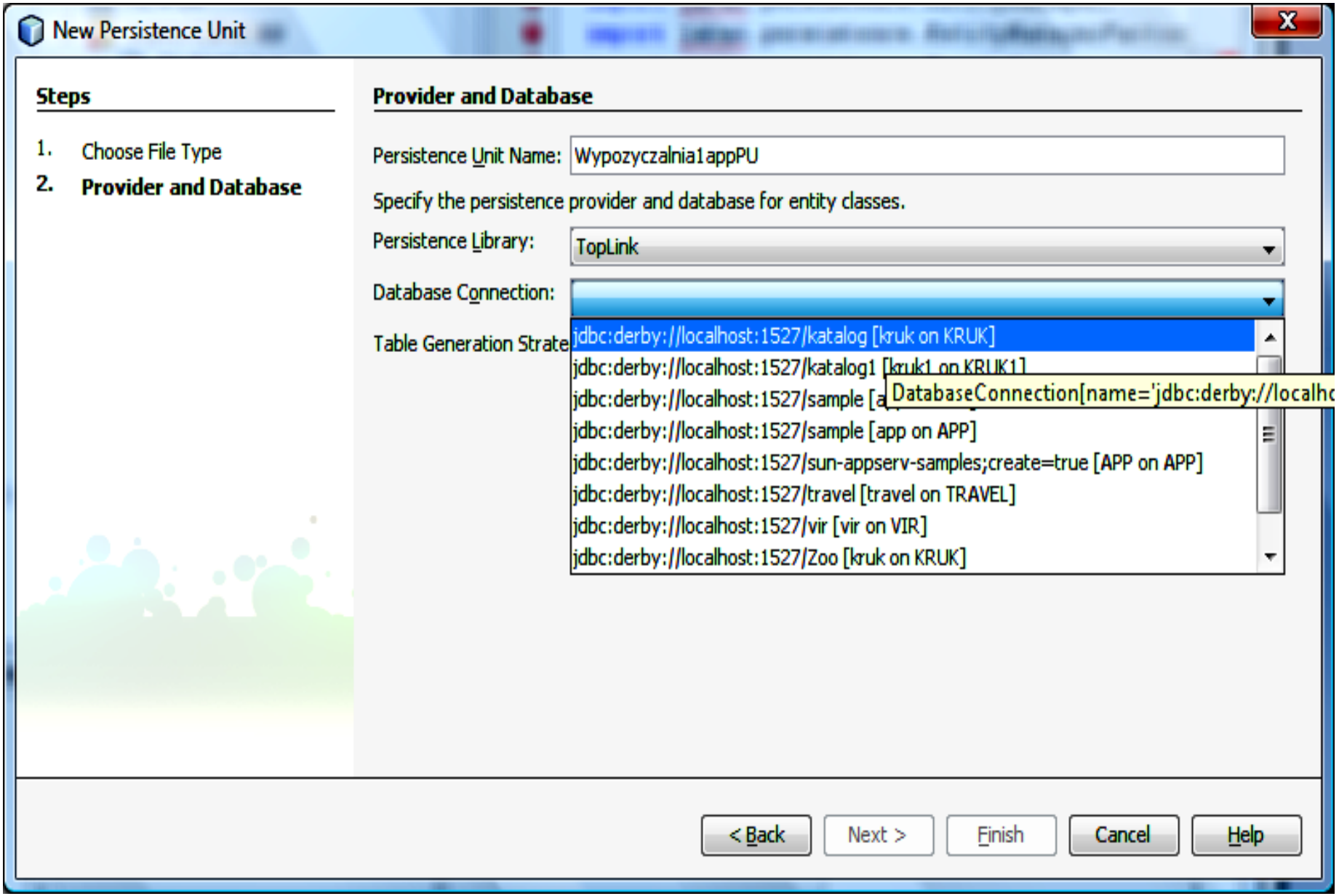


7. Tworzenie warstwy integracji w projekcje Java Application. Zastosowanie wzorców projektowych typu Domain Store i Transfer Object.

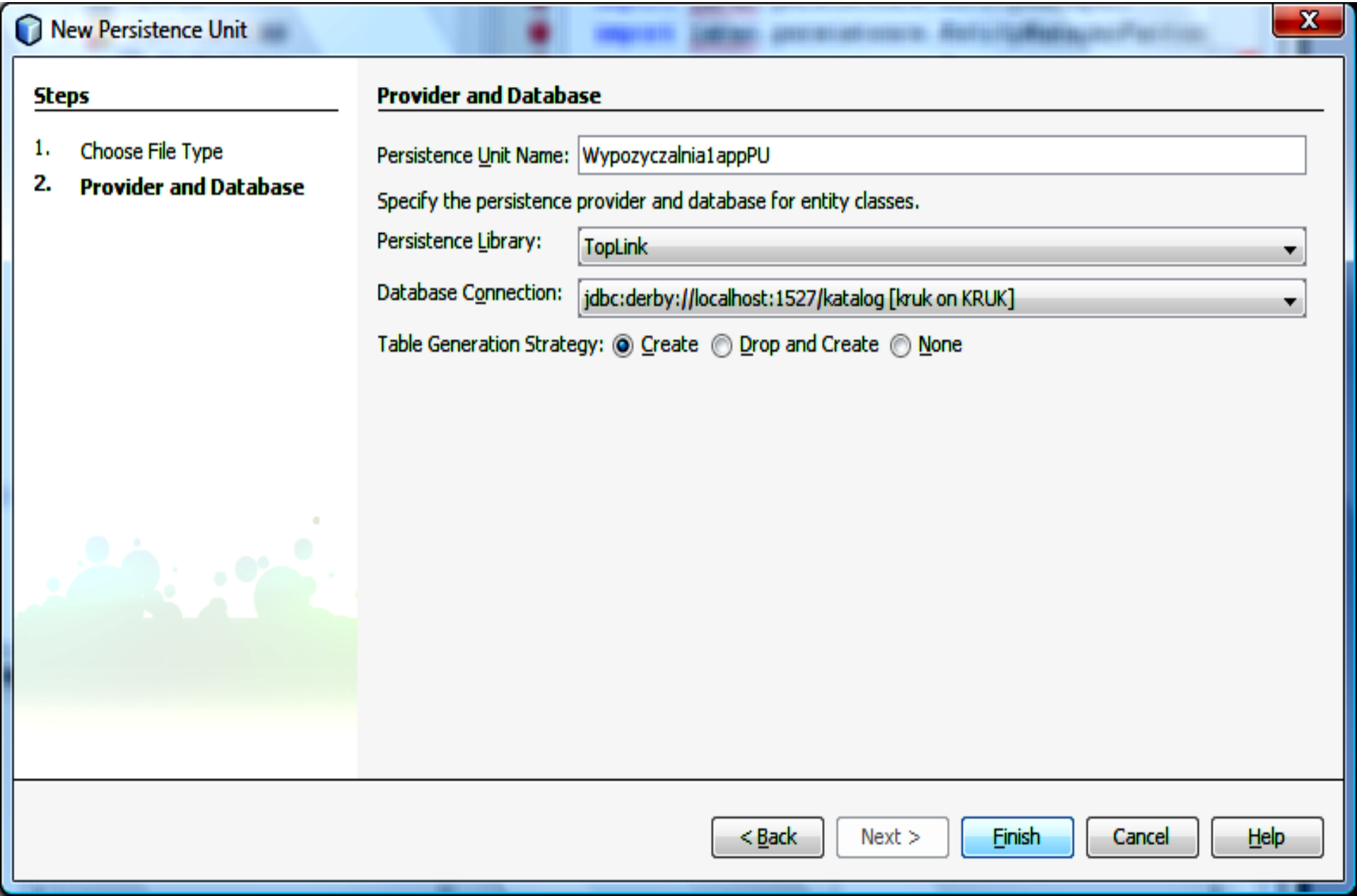
Wstawianie do projektu typu **Java Application** modułu typu **Persistence Unit (1)**



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



Tworzenie modułu utrwalania danych dla technologii TopLink (3)



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

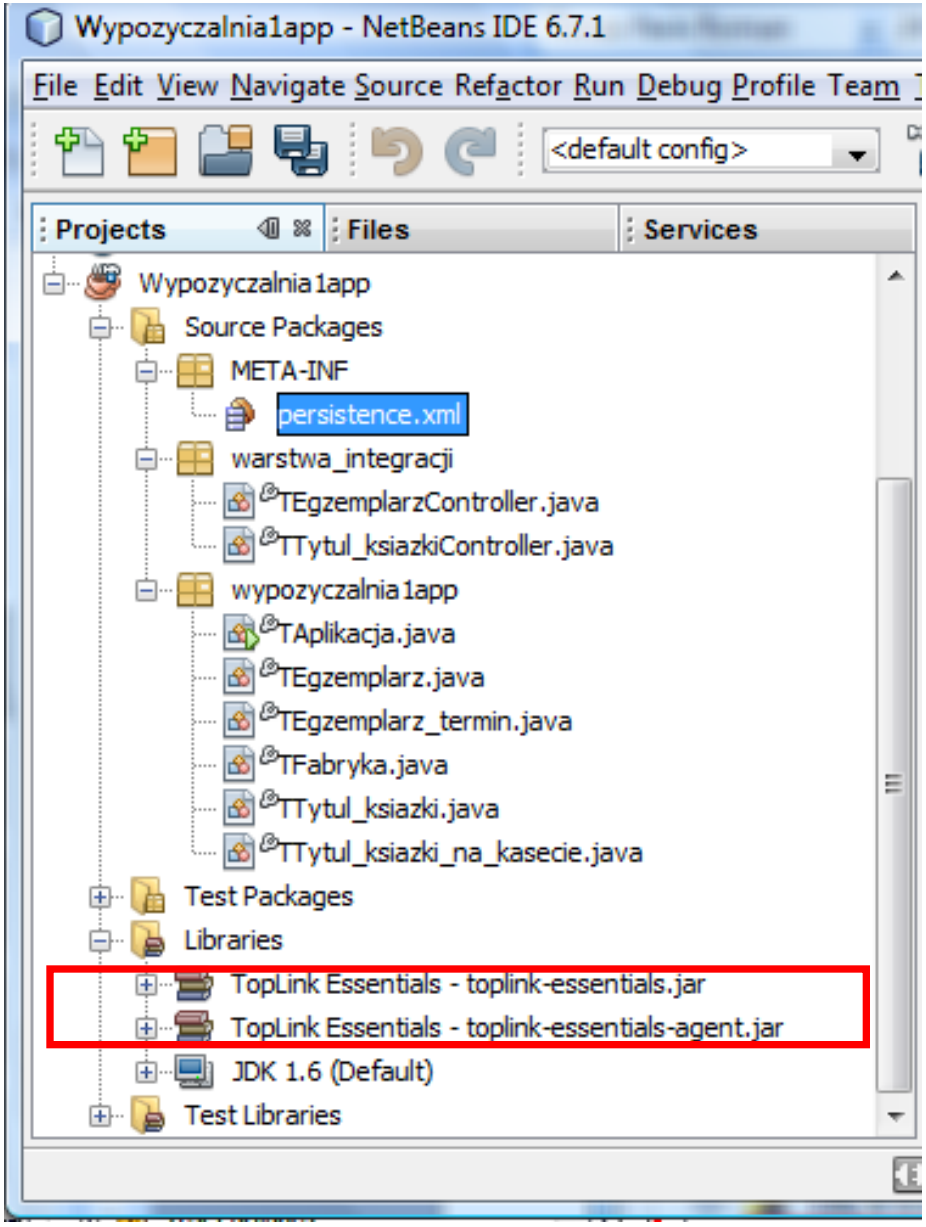
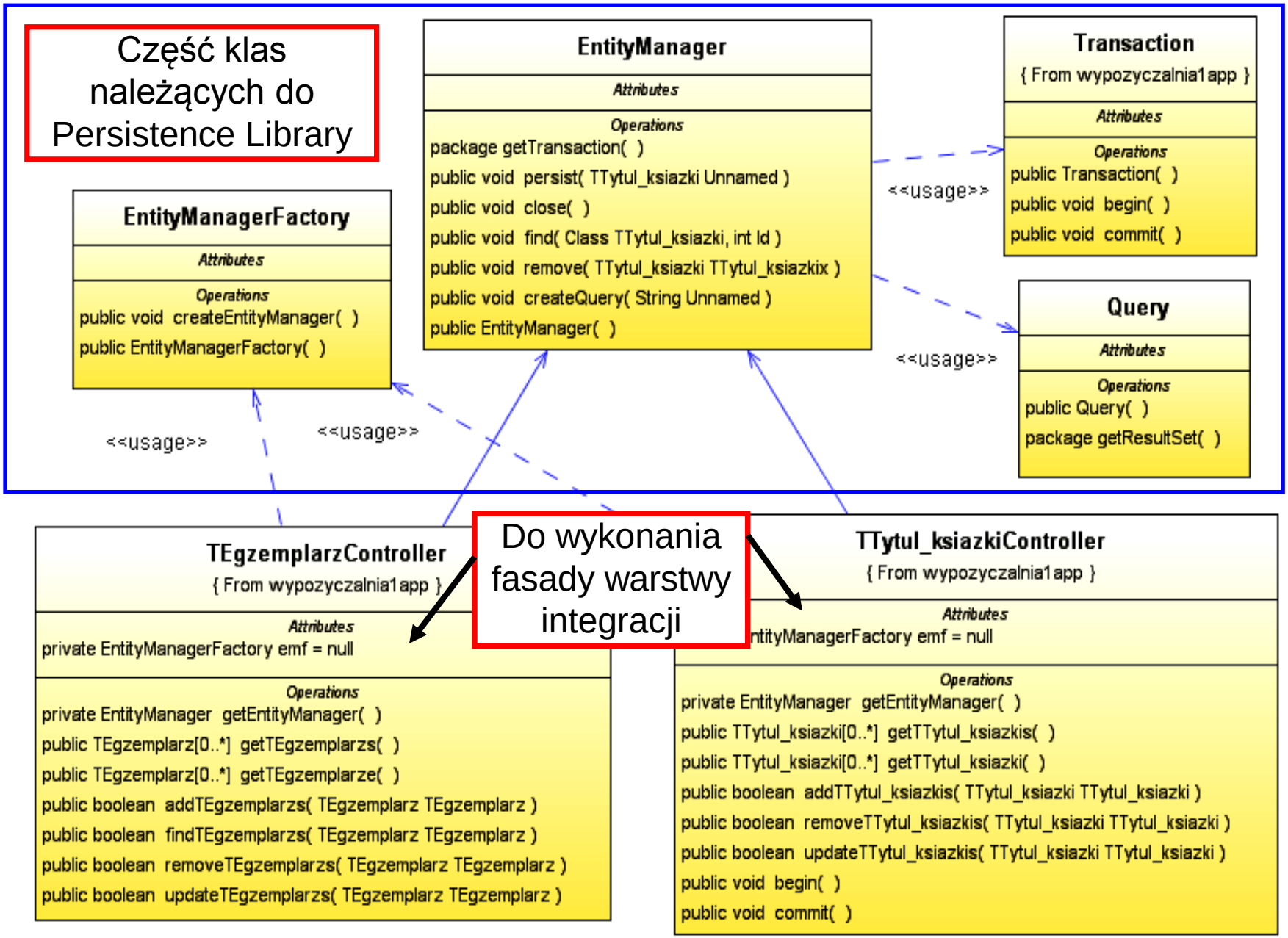
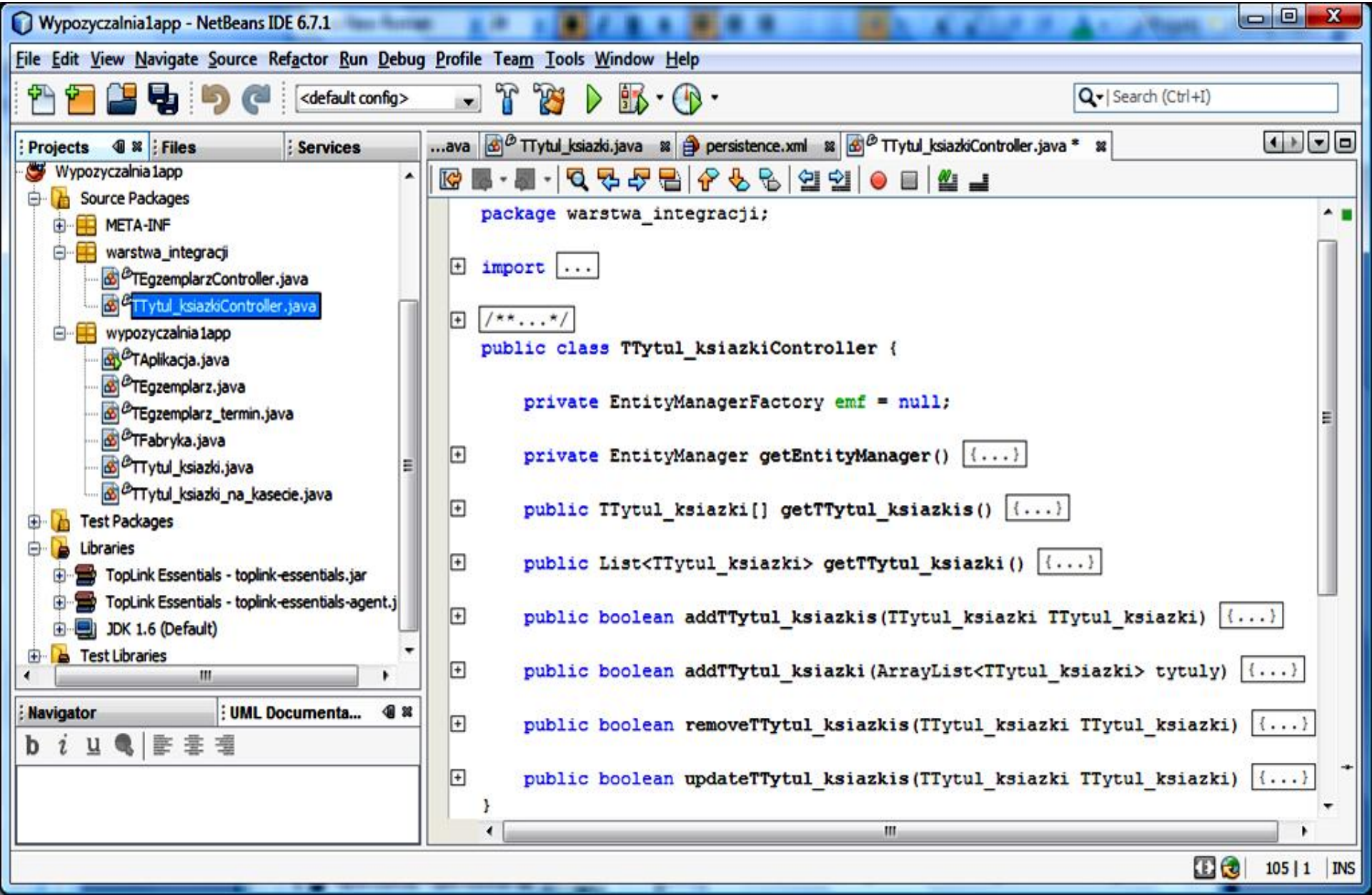


Diagram klas – uproszczony schemat warstwy integracji (5)



Przykład klasy warstwy integracji – dla każdej klasy typu **Entity** (tutaj dla **TTytul_ksiazki**) (6)



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("Wypozyczalnia1appPU");
     }
     return emf.createEntityManager();
    }
```

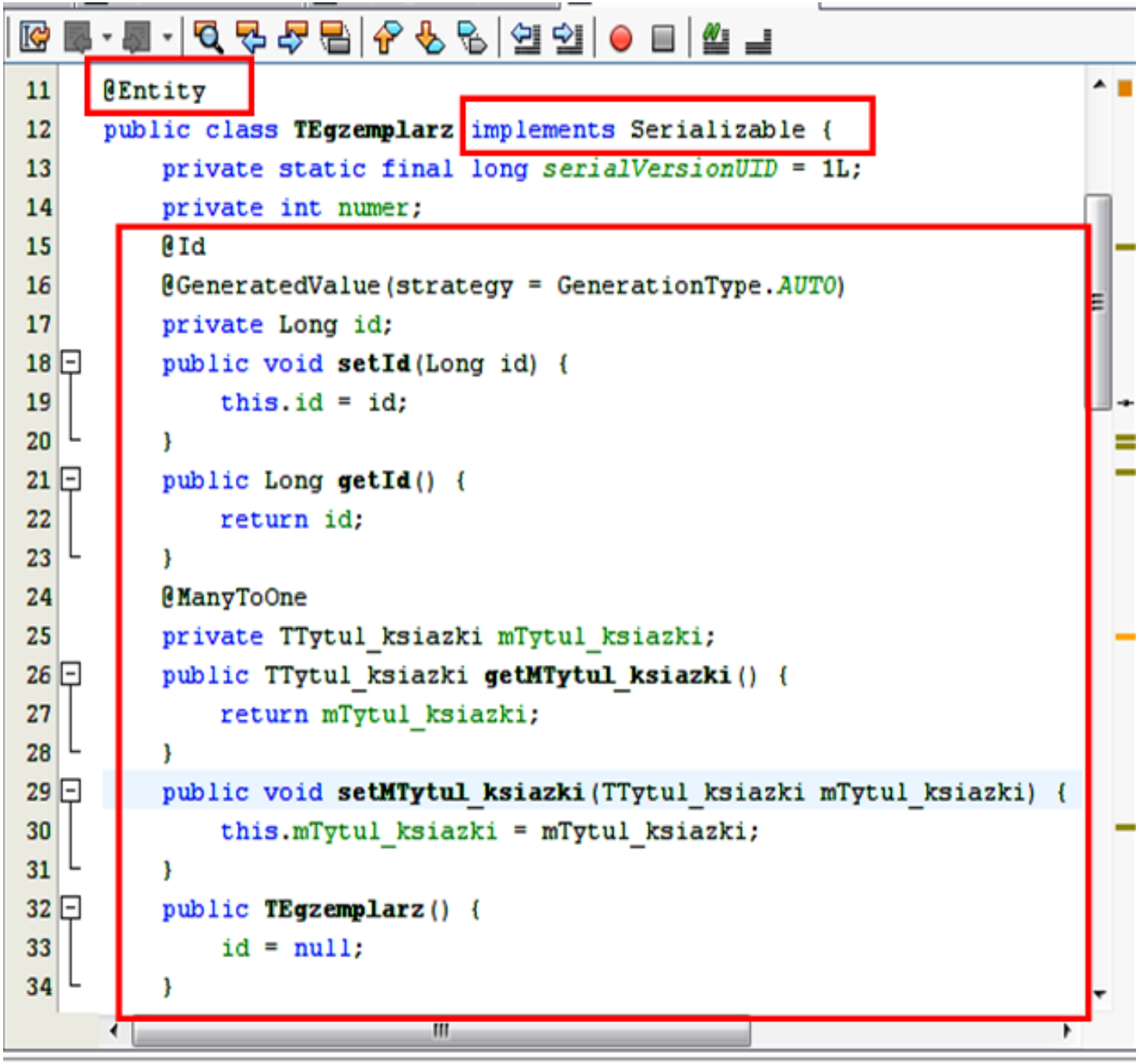
```
public boolean addTTytul_ksiazkis(TTytul_ksiazki TTytul_ksiazki)
 {
      EntityManager em = getEntityManager();
     try {
         em.getTransaction().begin();
         em.persist(TTytul_ksiazki);
         em.getTransaction().commit();
      } finally {
         em.close();
         return false;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

Itd….

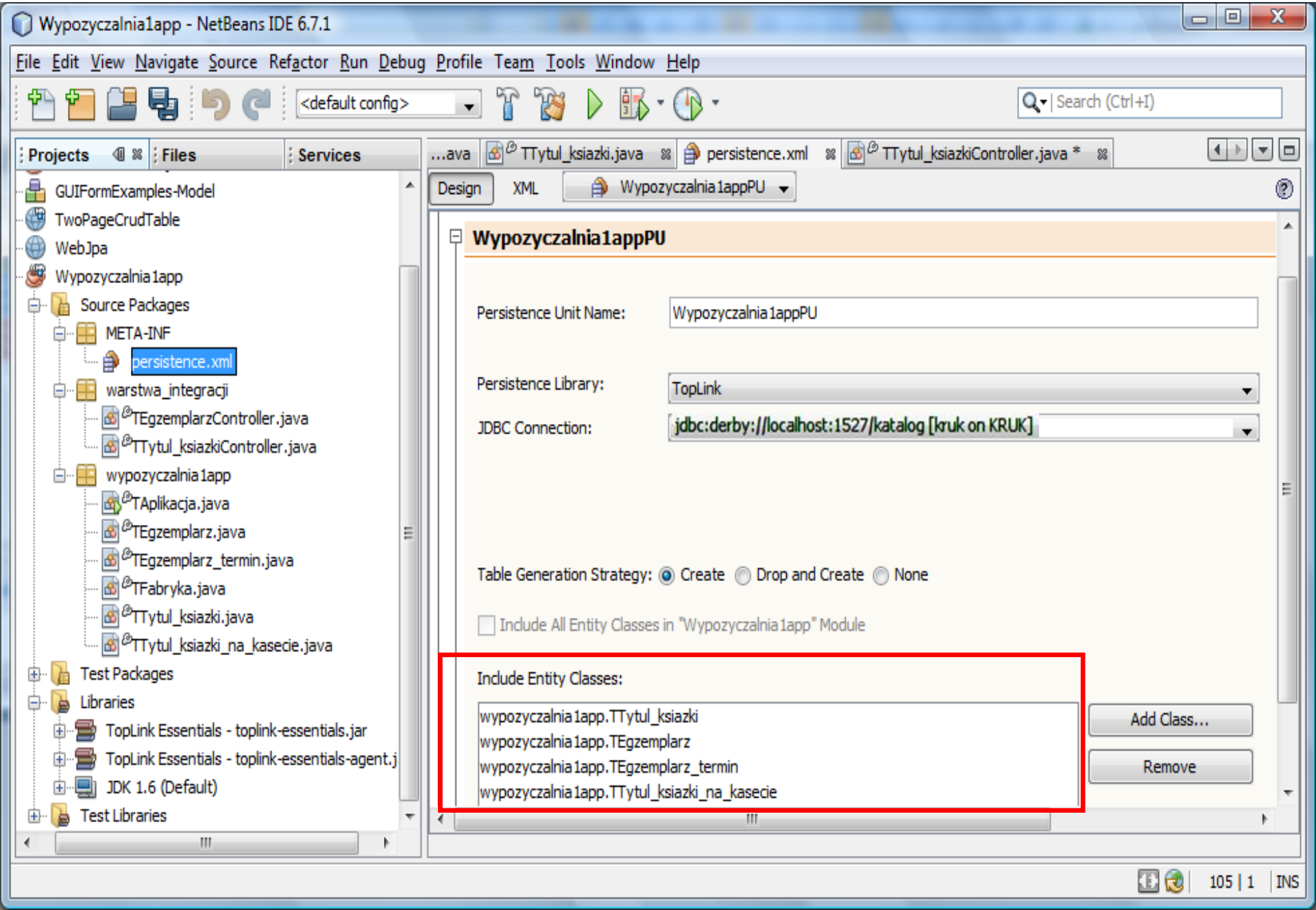
Dodanie adnotacji do klas typu dane („Entity”) wspierających mapowanie obiektowo-relacyjne (8)

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       @Id
21       @GeneratedValue(strategy = GenerationType.AUTO)
22       private Long id;
23       public Long getId()
24           {   return id;        }
25       public void setId(Long id)
26           {   this.id = id;    }
27       @OneToMany(mappedBy = "mTytul_ksiazki")
28       private Collection<TEgzemplarz> Ksiazka;
29       public Collection<TEgzemplarz> getKsiazka()
30           {   return Ksiazka;     }
31       public void setKsiazka(Collection<TEgzemplarz> Ksiazka)
32           {   this.Ksiazka = Ksiazka;   }
33       public TTytul_ksiazki()
34           {   id = null;   }
35       @Transient
36       private ArrayList<TEgzemplarz> mKsiazka =
37               new java.util.ArrayList<TEgzemplarz>();
38       public ArrayList<TEgzemplarz> getMKsiazka()
39           {   return mKsiazka;    }
40       public void setMKsiazka(ArrayList<TEgzemplarz> mKsiazka)
41           {   this.mKsiazka = mKsiazka;   }
```


@Entity
public class TEgzemplarz implements Serializable {
private static final long serialVersionUID = 1L;
private int numer;
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Long id;
public void setId(Long id) {
this.id = id;
}
public Long getId() {
return id;
}
@ManyToOne
private TTytul_ksiazki mTytul_ksiazki;
public TTytul_ksiazki getMTytul_ksiazki() {
return mTytul_ksiazki;
}
public void setMTytul_ksiazki(TTytul_ksiazki mTytul_ksiazki) {
this.mTytul_ksiazki = mTytul_ksiazki;
}
public TEgzemplarz() {
id = null;
}

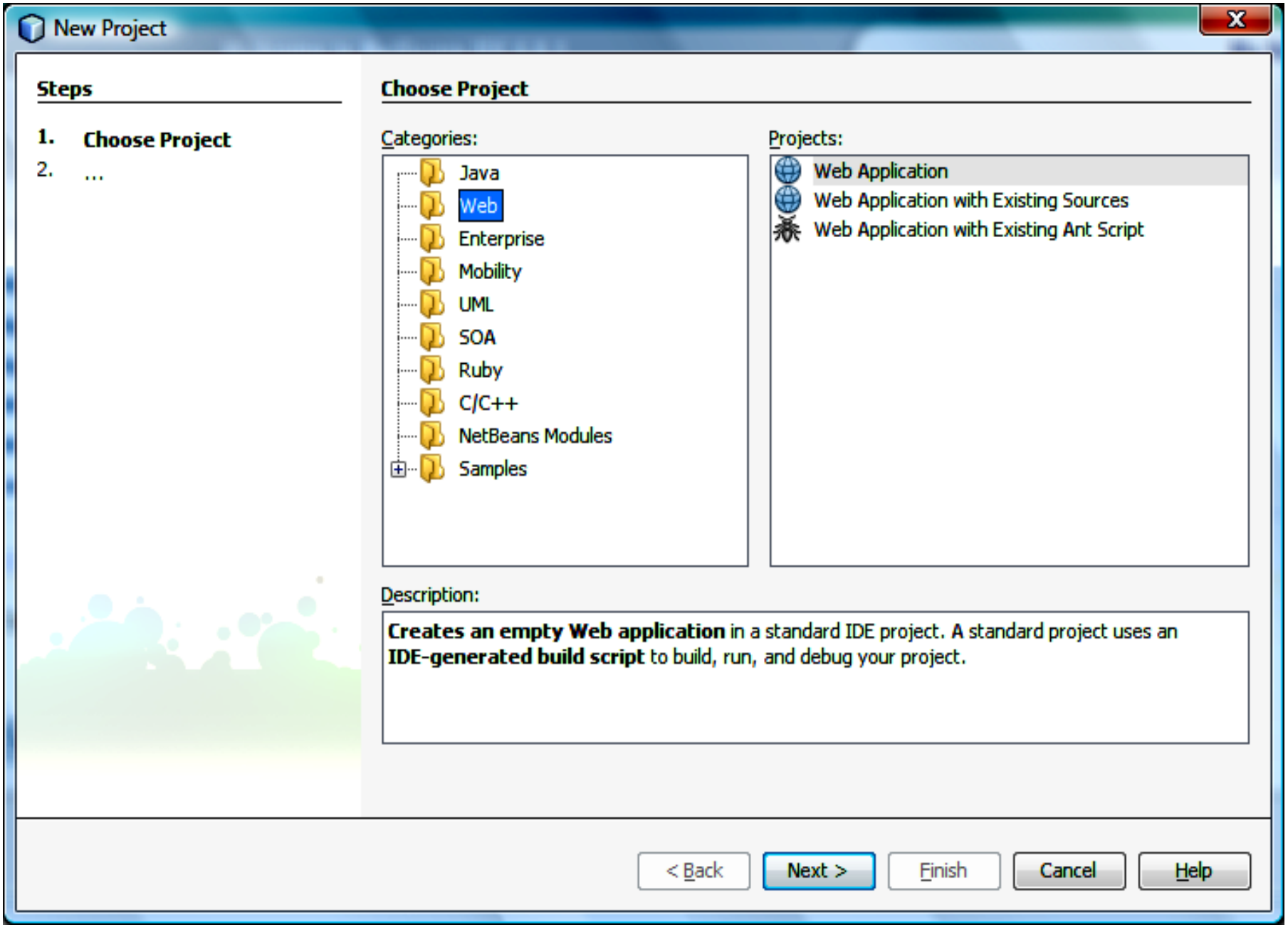
Wstawienie do modułu typu **Persistence Unit** wybrane klasy typu **Entity (9)**



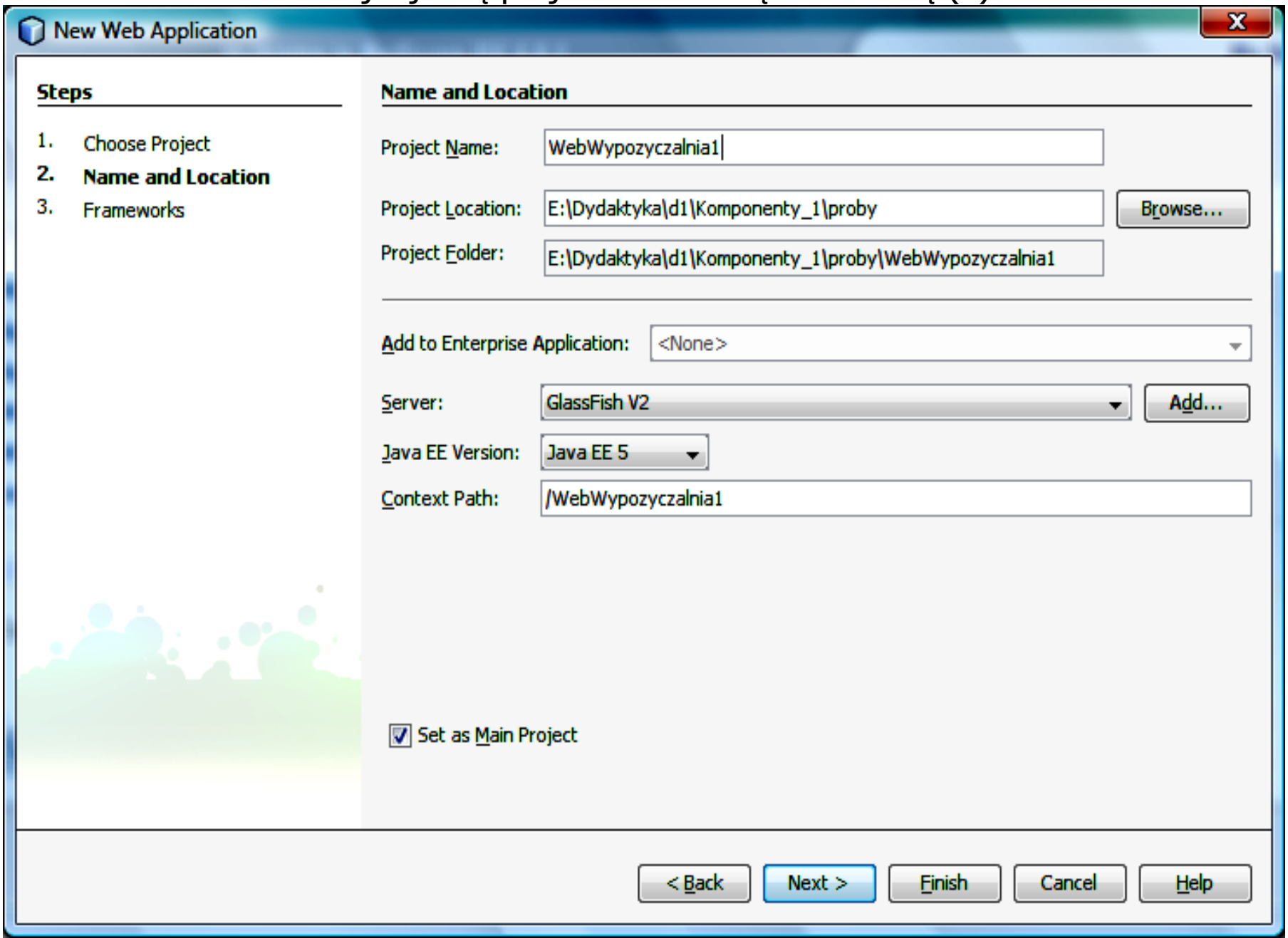
8. Tworzenie warstwy prezentacji
Pierwszy etap – tworzenie stron typu JSP

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

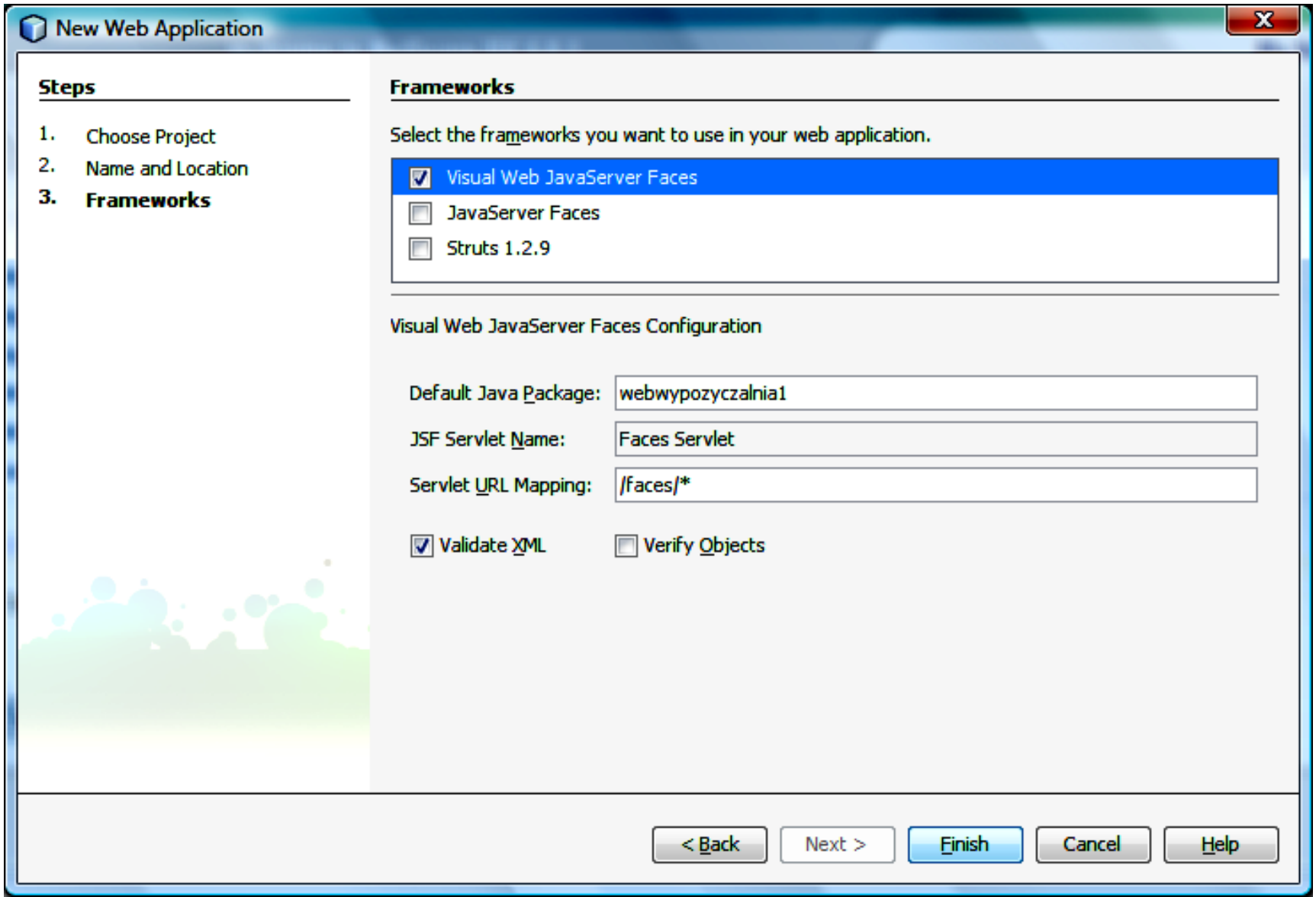
Tworzenie projektu kategorii Web typu Web Application (1)

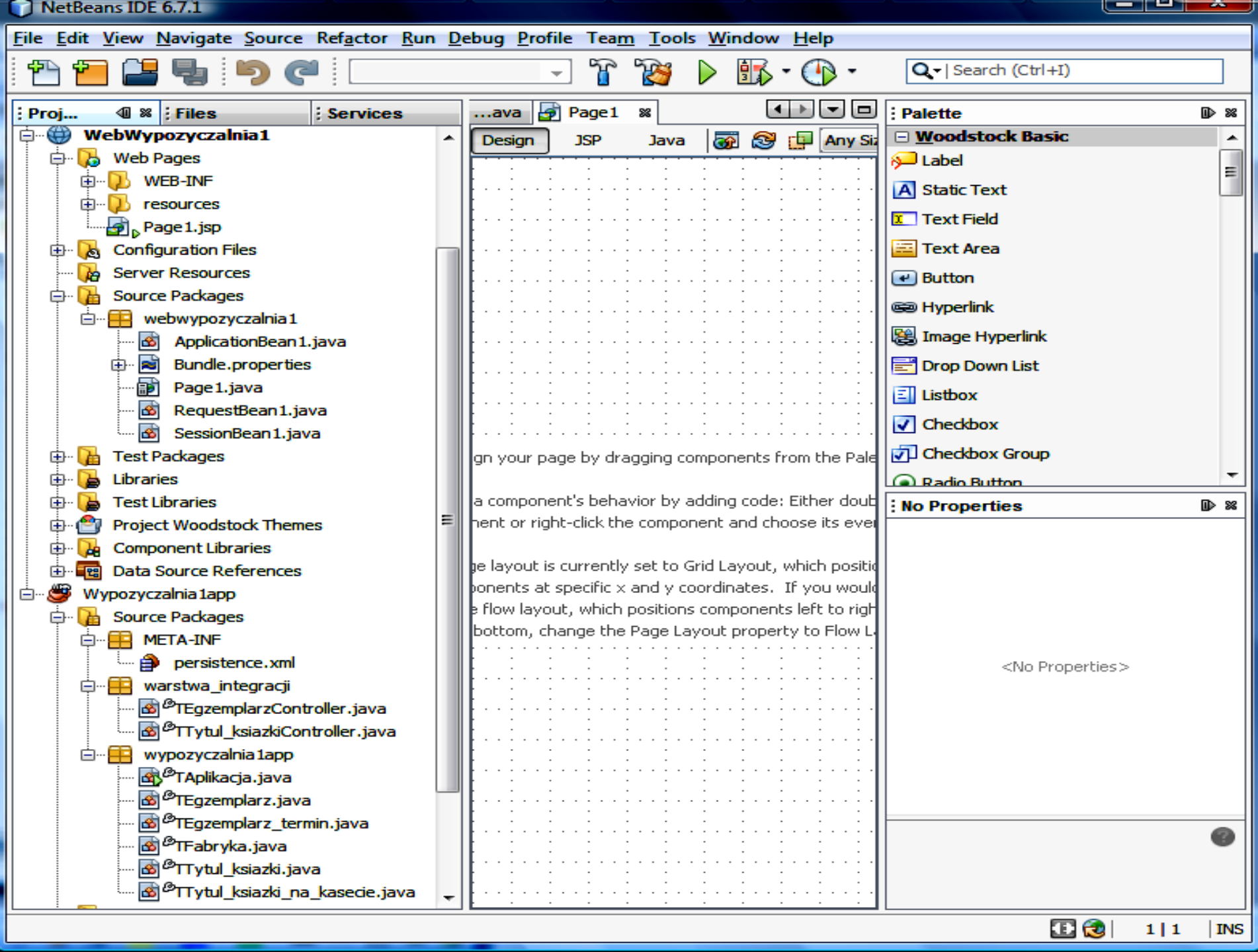


Projekt typu WebApplication tworzony w tym samym katalogu, w którym znajduje się projekt z warstwą biznesową (2)

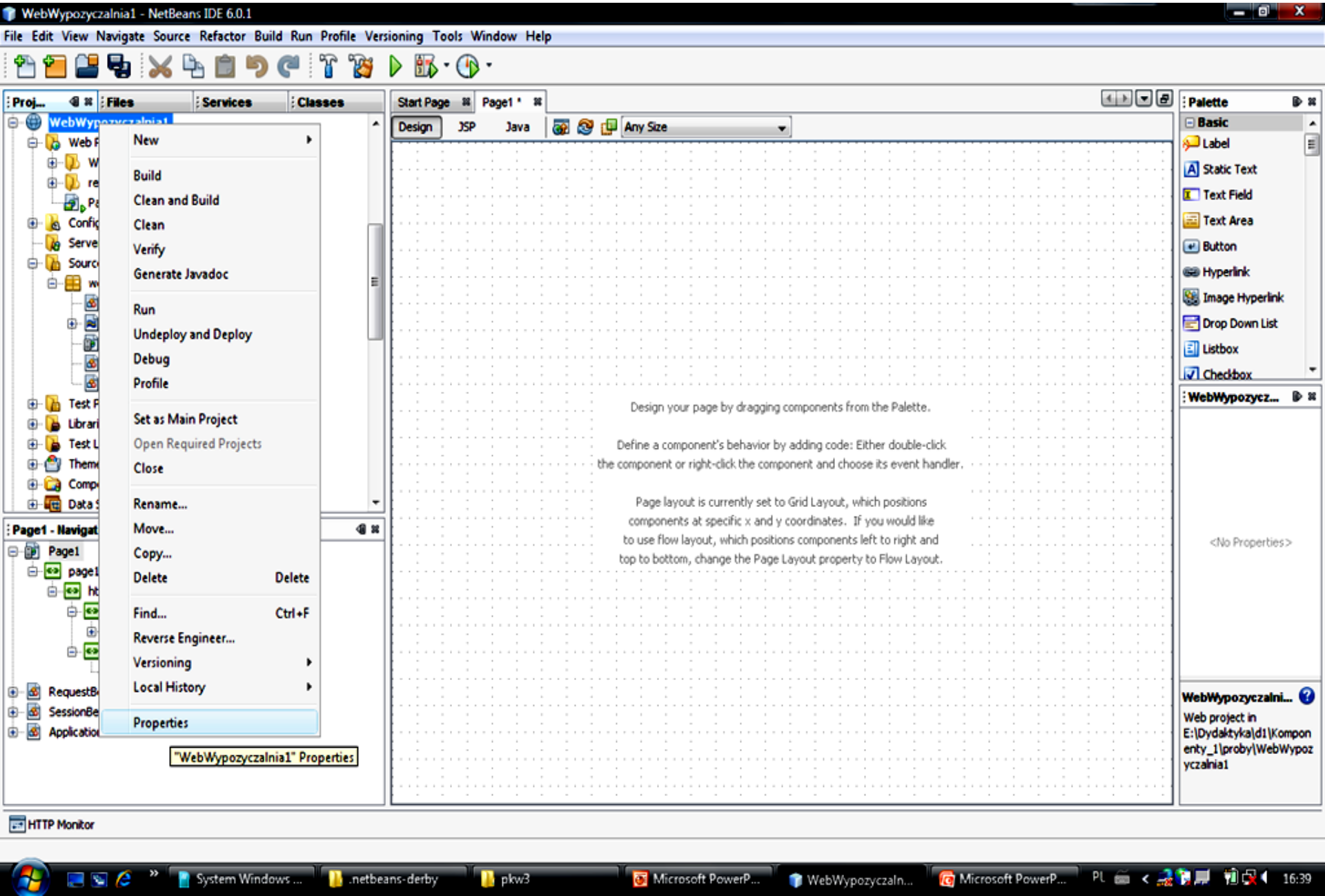


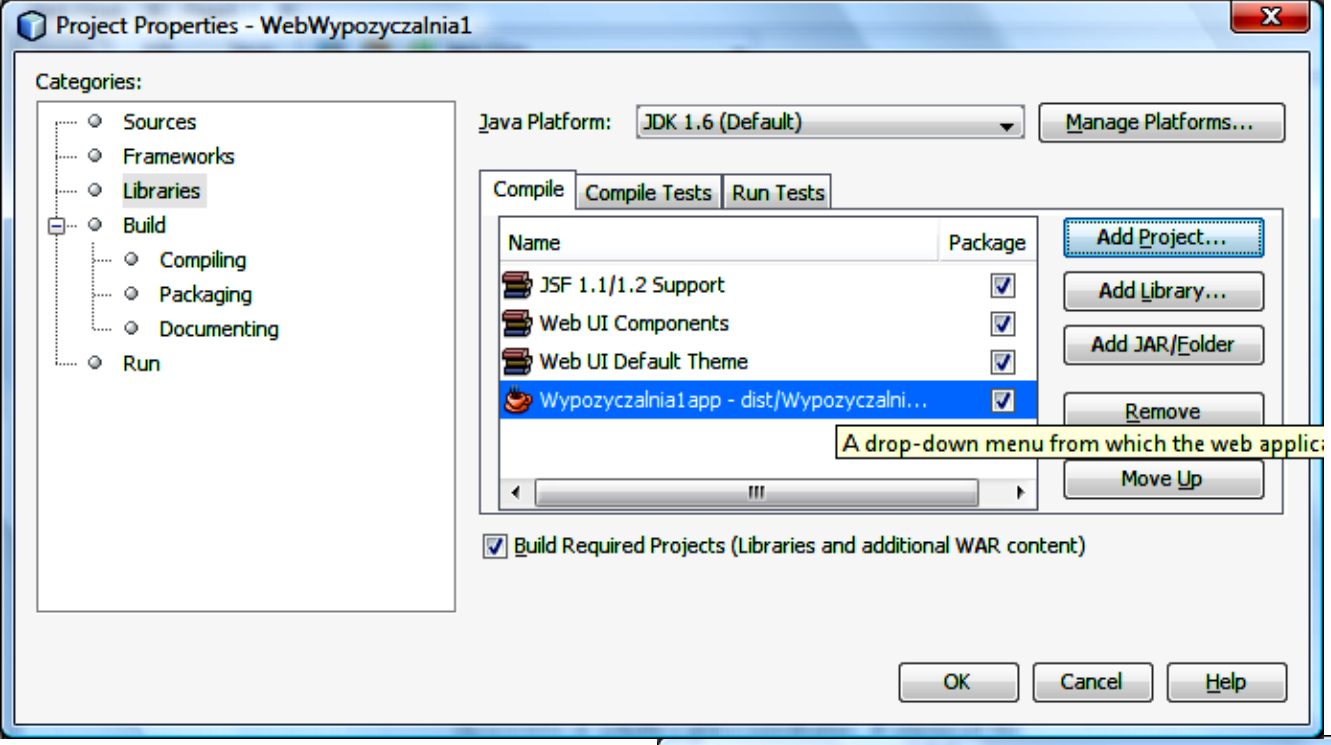
Projekt może być oparty np. na Visual Web JavaSever Faces (3) – przykład metody tworzenia warstwy prezentacji metoda „przeciągnij i upuść”

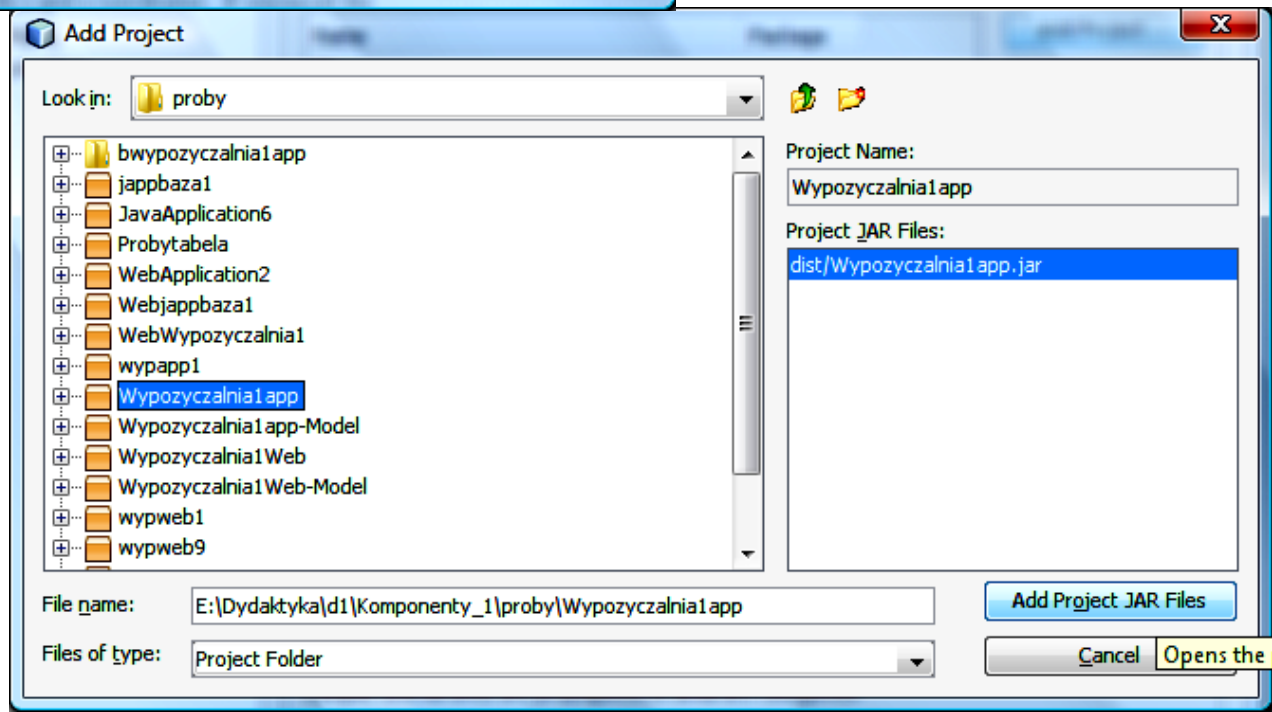



NetBeans IDE 6.7.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Proj... Files Services
WebWypozyczalnia1
Web Pages
WEB-INF
resources
Page1.jsp
Configuration Files
Server Resources
Source Packages
webwypozyczalnia1
ApplicationBean1.java
Bundle.properties
Page1.java
RequestBean1.java
SessionBean1.java
Test Packages
Libraries
Test Libraries
Project Woodstock Themes
Component Libraries
Data Source References
Wypozyczalnia1app
Source Packages
META-INF
persistence.xml
warstwa_integracji
TEgzemplarzController.java
TTytul_ksiazkiController.java
wypozyczalnia1app
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
nent or right-click the component and choose its ever
ge layout is currently set to Grid Layout, which positi
onents at specific x and y coordinates. If you would
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>
1 | 1
INS

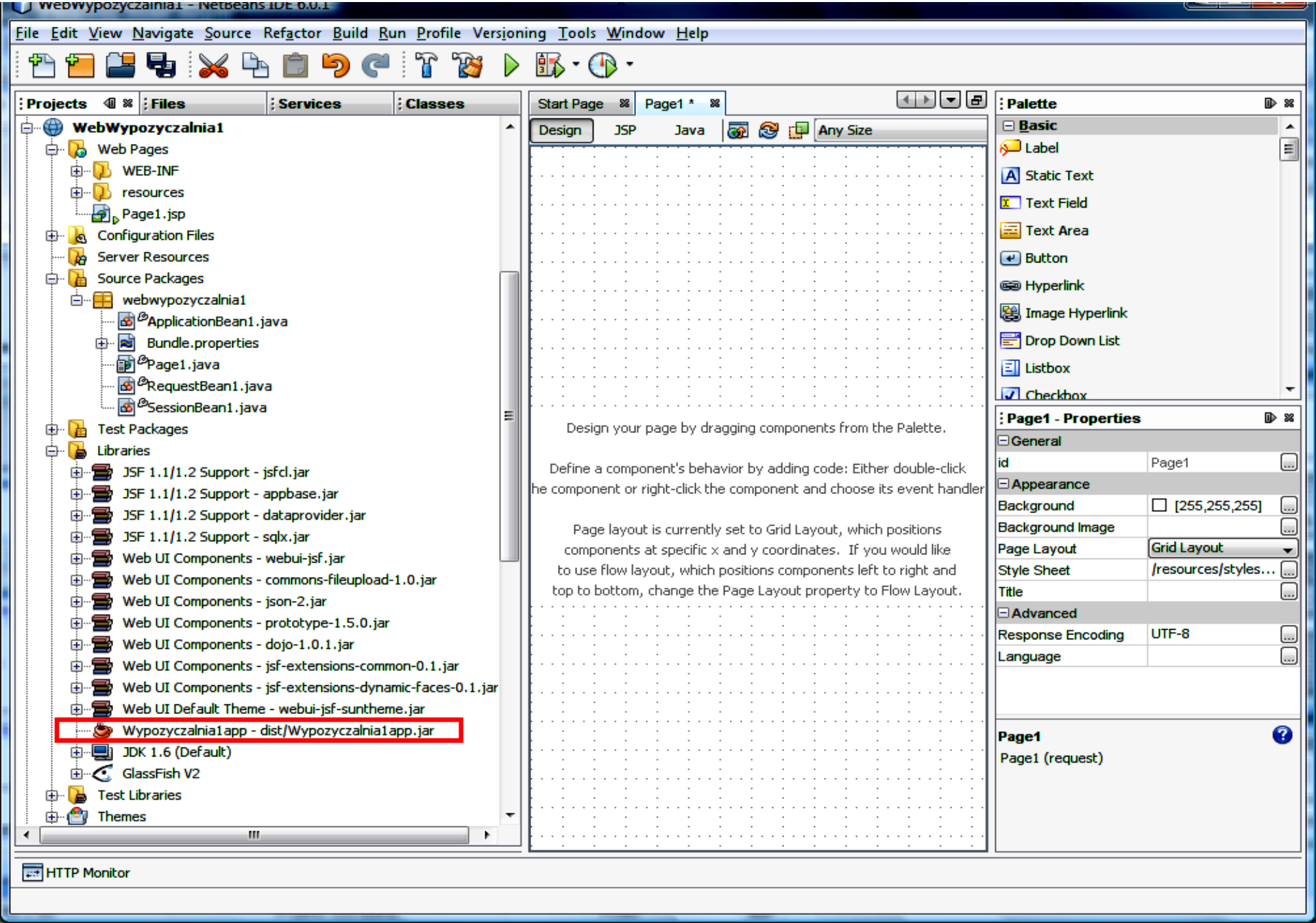
Łączenie projektu typu JavaApplication zawierającym warstwę biznesową i integracji z projektem Web Application zawierającym warstwy prezentacji i klienta (4)



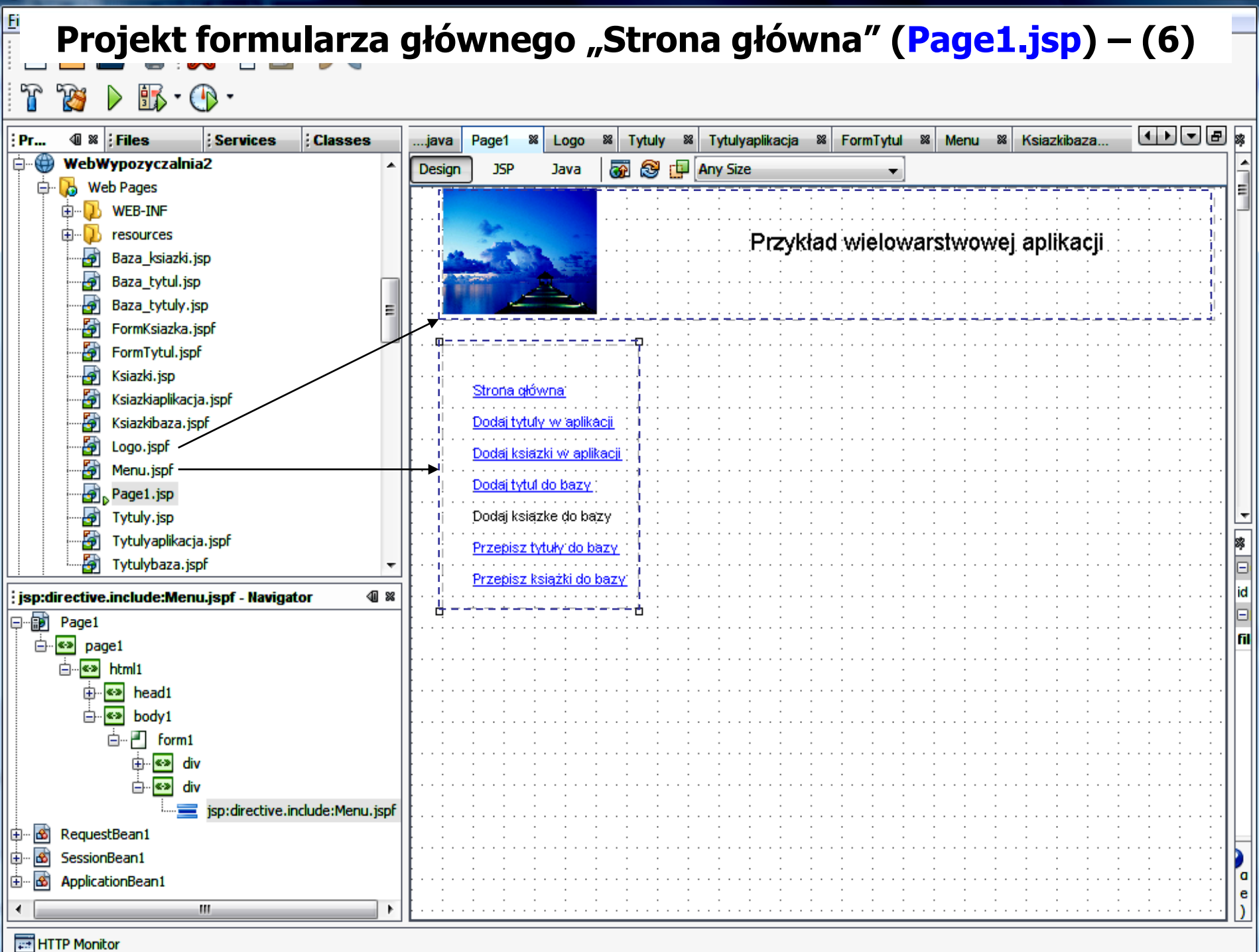

Project Properties - WebWypozyczalnia1
Categories:
Sources
Frameworks
Libraries
Build
Compiling
Packaging
Documenting
Run
Java Platform:
JDK 1.6 (Default)
Manage Platforms...
Compile
Compile Tests
Run Tests
Name
Package
JSF 1.1/1.2 Support
Web UI Components
Web UI Default Theme
Wypozyczalnia1app - dist/Wypozyczalni...
Add Project...
Add Library...
Add JAR/Folder
Remove
A drop-down menu from which the web applic
Move Up
Build Required Projects (Libraries and additional WAR content)
OK
Cancel
Help

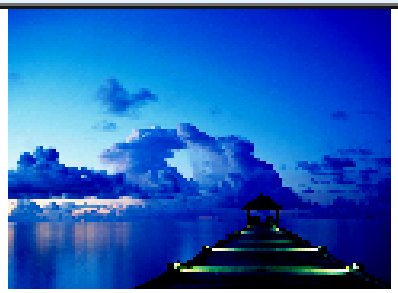

Add Project
Look in:
proby
bwypozyczalnia1app
jappbaza1
JavaApplication6
Probytabela
WebApplication2
Webjappbaza1
WebWypozyczalnia1
wypapp1
Wypozyczalnia1app
Wypozyczalnia1app-Model
Wypozyczalnia1Web
Wypozyczalnia1Web-Model
wypweb1
wypweb9
Project Name:
Wypozyczalnia1app
Project JAR Files:
dist/Wypozyczalnia1app.jar
File name:
E:\Dydaktyka\d1\Komponenty_1\proby\Wypozyczalnia1app
Files of type:
Project Folder
Add Project JAR Files
Cancel
Opens the

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



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





Przykład wielowarstwowej aplikacji

Strona główna

Dodaj tytuły w aplikacji

Dodaj ksiazki w aplikacji

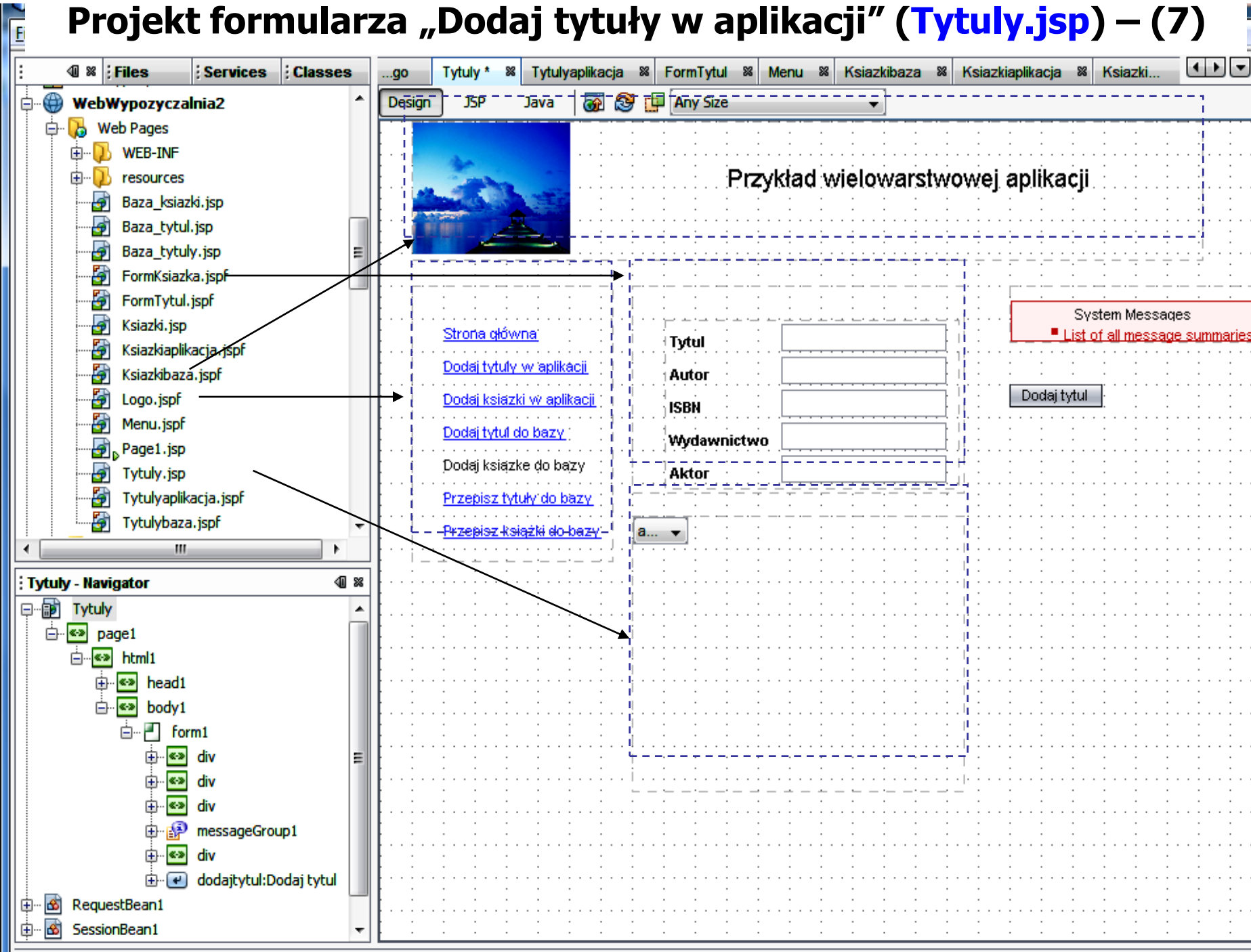
Dodaj tytul do bazy

Dodaj ksiazke do bazy

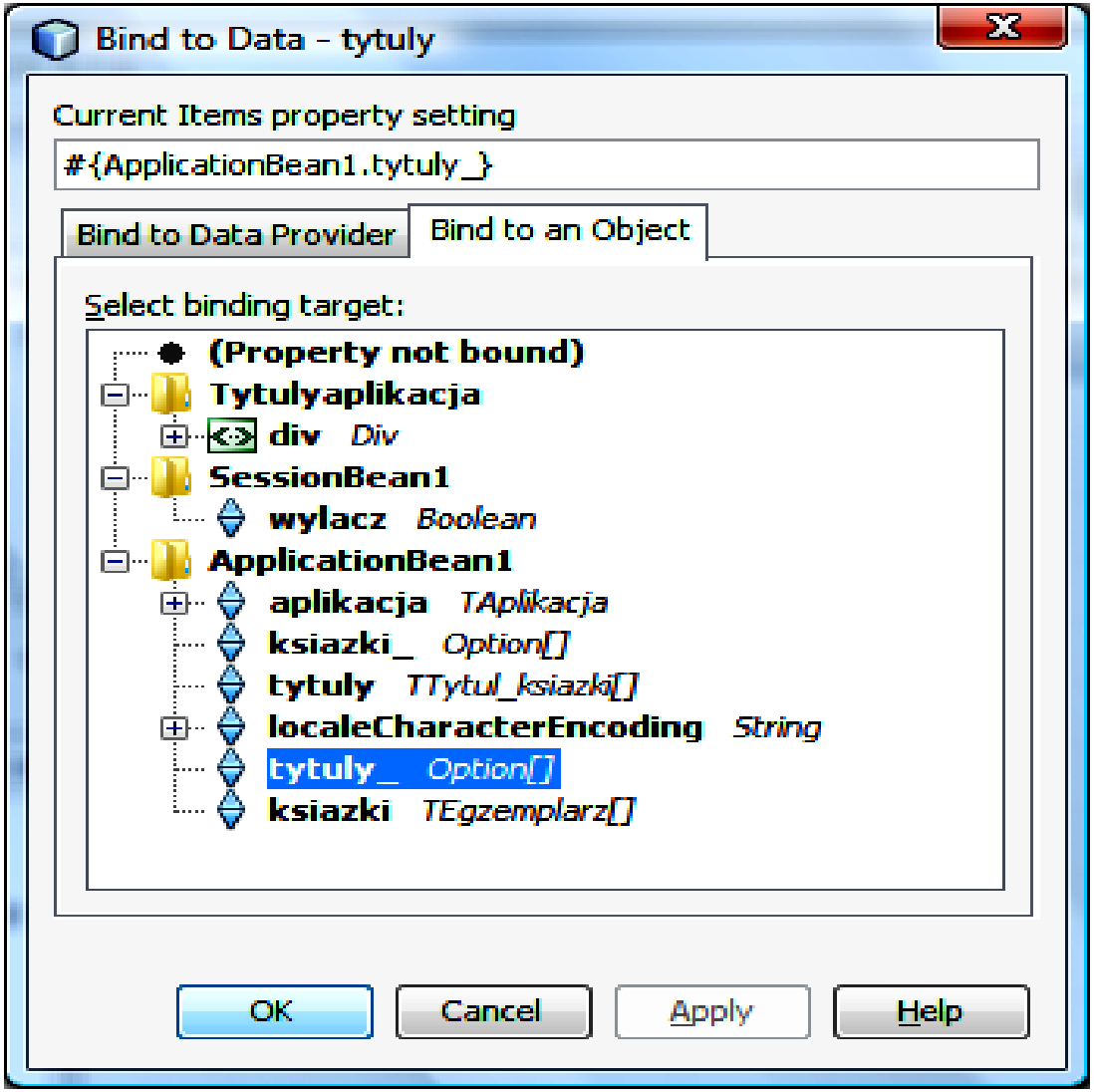
Przepisz tytuły do bazy

Przepisz książki do bazy

Projekt formularza „Dodaj tytuły w aplikacji” (Tytuly.jsp) – (7)



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 ksiazki w aplikacji
Dodaj tytul do bazy
Dodaj ksiazke do bazy
Przepisz tytuły do bazy
Przepisz książki do bazy

Tytul

Autor

ISBN

Wydawnictwo

Aktor

Dodaj tytul

Tytul: 1 Autor: 1 ISBN: 1 Wydawnictwo: 1

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

Oprogramowanie dotyczące formularza Tytuly.jsp

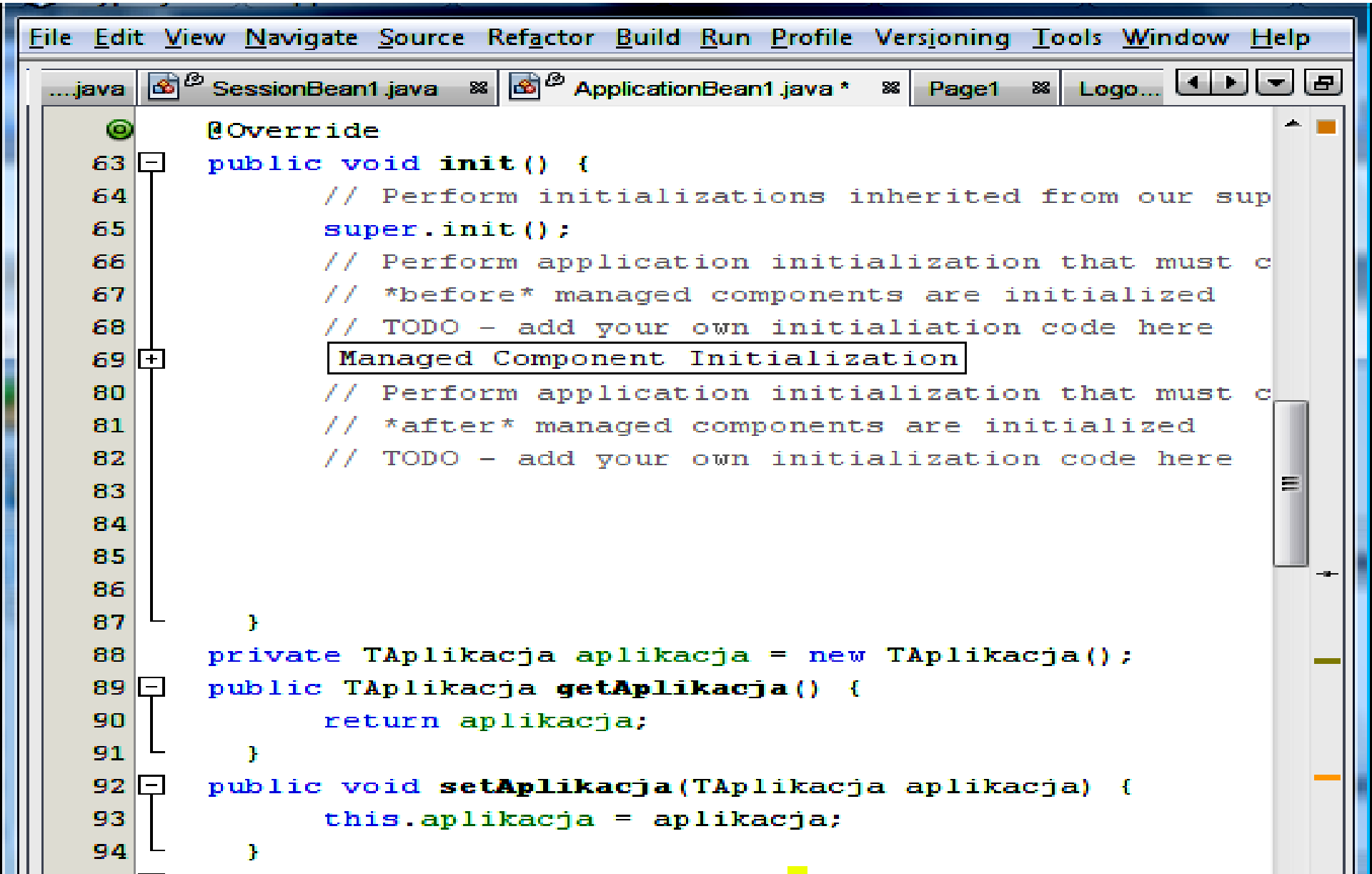
Definicje metod w klasie *Tytuly* dla strony typu JSP – do wstawiania nowego tytułu do warstwy biznesowej (obsługa zdarzenia *dodajtytul_action*) oraz generowania widoku metodą *prerender* (wygaszanie linku do bieżącej strony w formularzu *Menu* typu JSPF i czyszczenie pól formularza *FormTytul* typu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



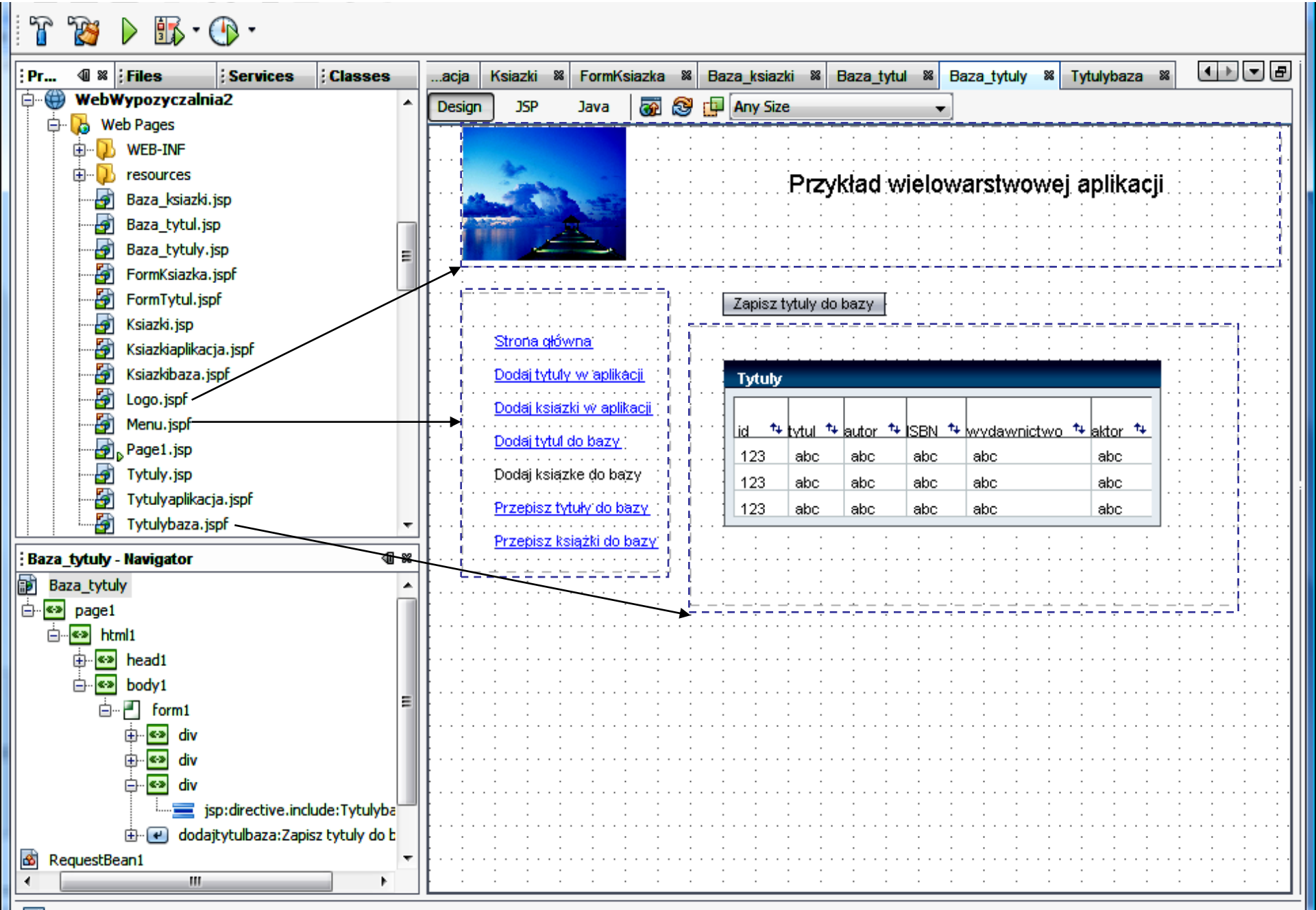
Definicje metod w klasie *ApplicationBean1* związanych z zapisem (*dodaj_tytul*) i odczytem (*przygotujtytuly*) danych typu kolekcja obiektów *TTytul_ksiazki* i *TTytul_ksiazki_na_kasecie* w warstwie biznesowej – odczytane dane wstawiane są do tablicy *tytuly_*, która jest wyświetlana w komponencie typu DropDown List na stronie *Tytulyaplikacja* typu JSPF

...java | SessionBean1.java | ApplicationBean1.java | Page1 | Logo | Tytuly...

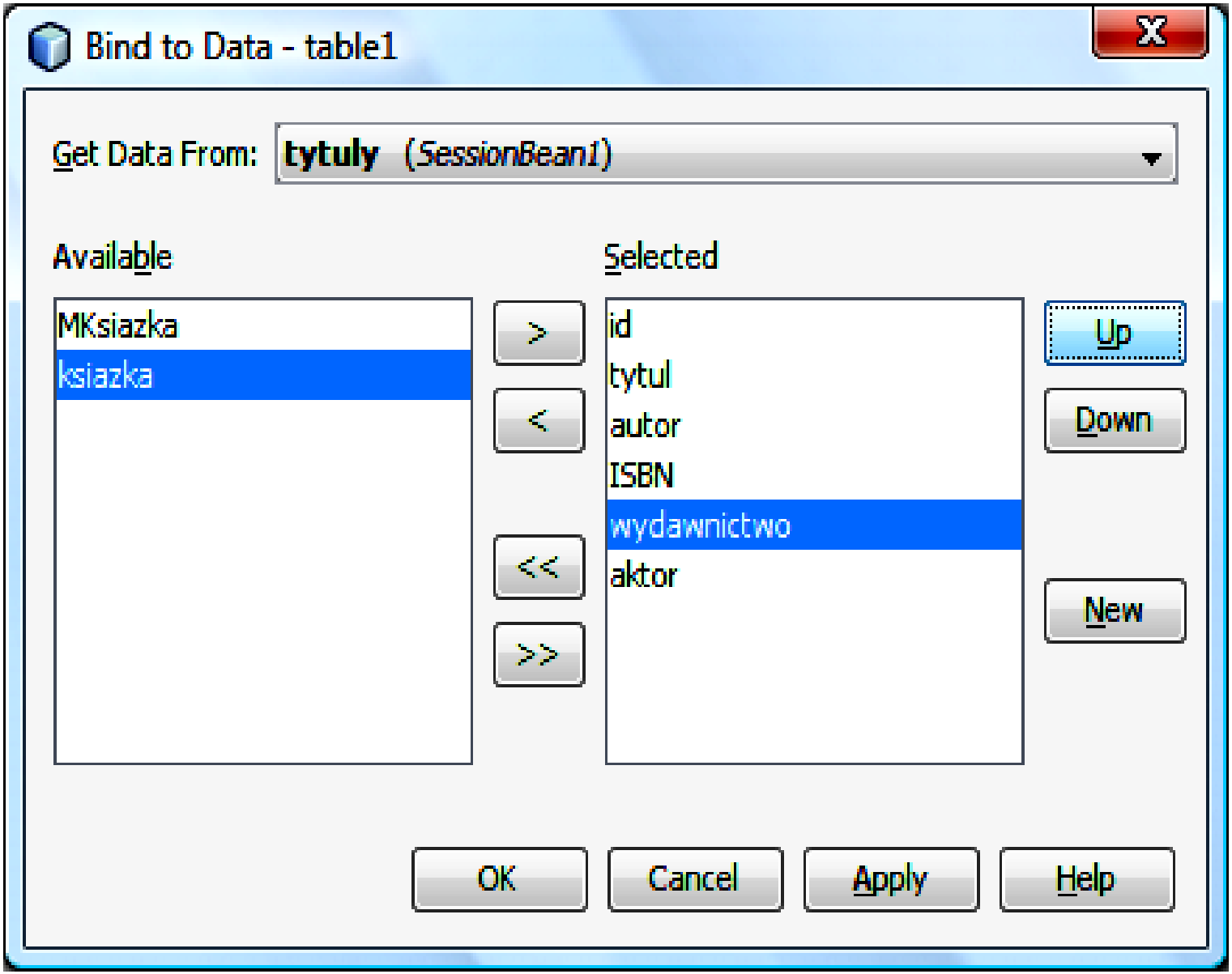
```
135   public void  dodaj_tytul(String dane[])
136     {getAplikacja().dodaj_tytul(dane);//przypadek użycia "dodaj tytu
137       przygotujtytuly();//wyswietlenie kolek
138     }
139   private Option tytuly_[] = new Option[0];
140   public Option[] getTytuly_() {
141         return tytuly_;
142     }
143   public void setTytuly_(Option[] tytuly_) {
144         this.tytuly_ = tytuly_;
145     }
146   public void przygotujtytuly() {
      ArrayList<TTytul_ksiazki> tytuly = aplikacja.getTytul_ksiazki();
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  }
```

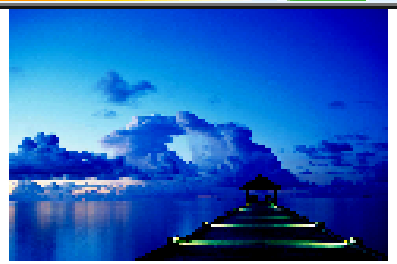
1. **PU Dodaj_tytul**
2. **Odświeżenie widoku**

Projekt formularza „Przepisz tytuły do bazy” (Baza_tytuly.jsp) – (8)



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





Przykład wielowarstwowej aplikacji

Zapisz tytuły do bazy

Strona główna

Dodaj tytuły w aplikacji

Dodaj ksiazki w aplikacji

Dodaj tytul do bazy

Dodaj ksiazke do bazy

Przepisz tytuły do bazy

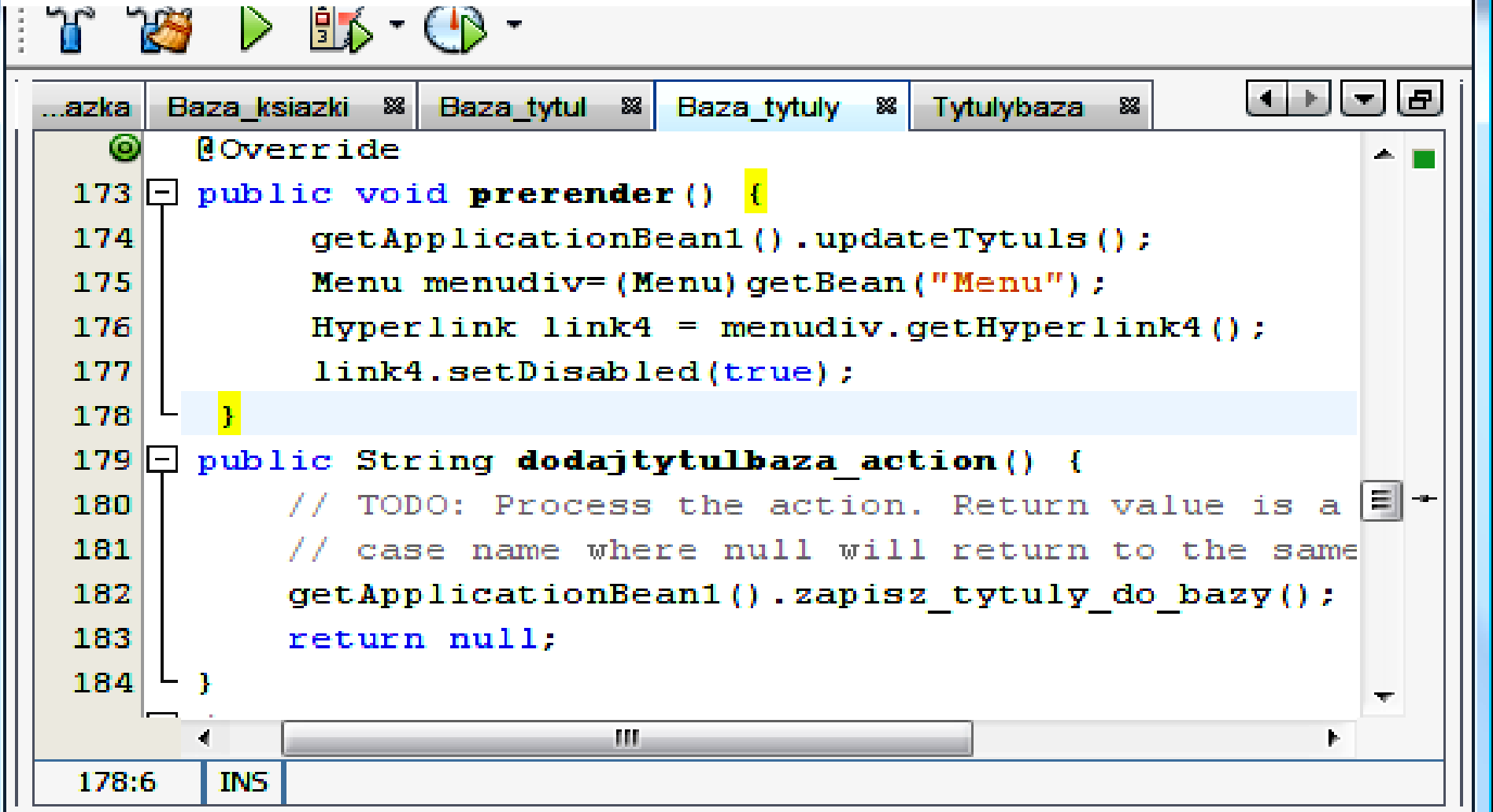
Przepisz książki do bazy

Tytuły

id	tytul	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 tytuly metodą *updateTytuls* w klasie *ApplicationBean1* wyświetlanej w komponencie *Table* strony *Tytulybaza* typu JSPF)

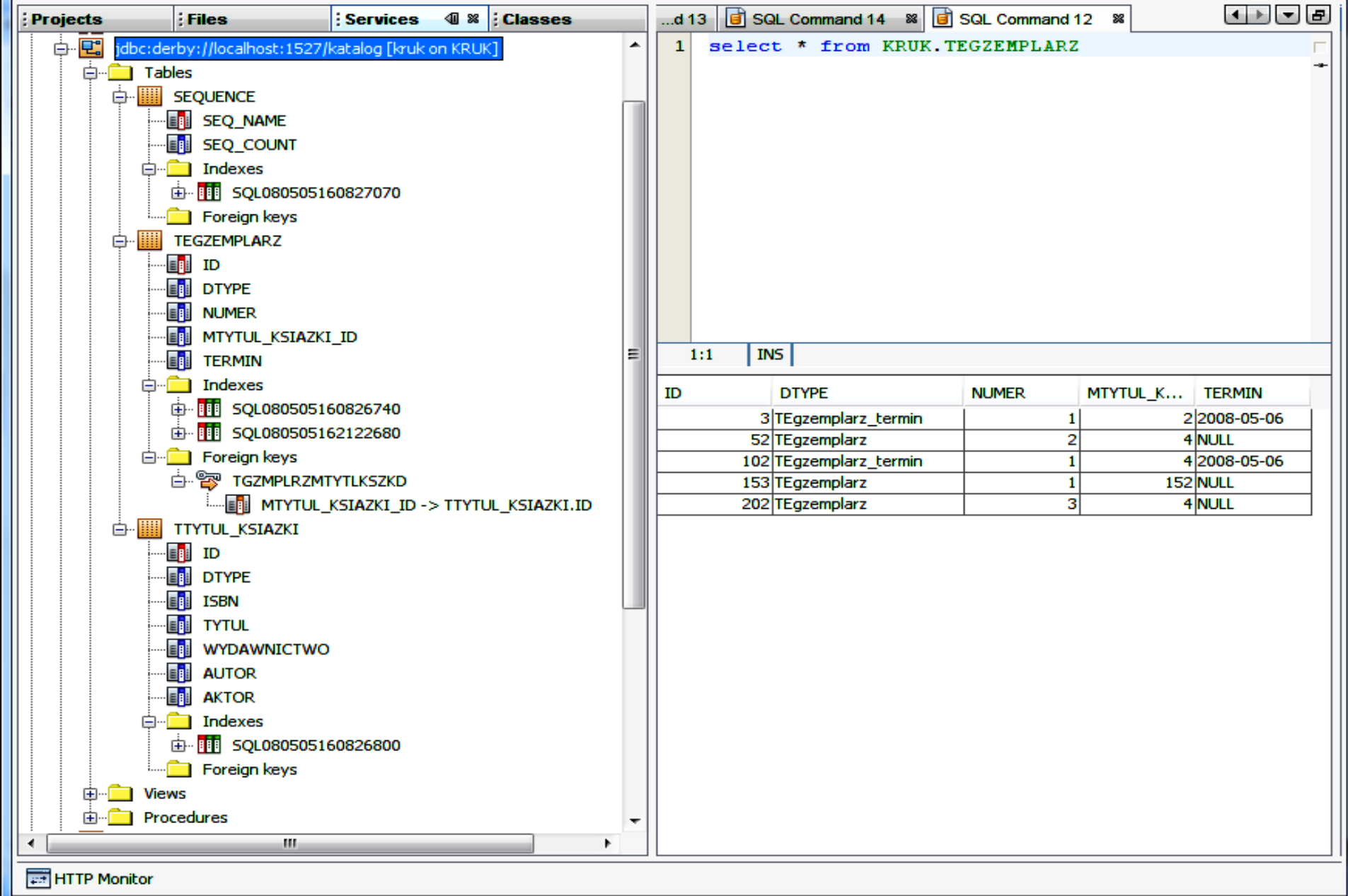


Definicje metod z *zapisz_tytuly_do_bazy* oraz *updateTytuls* w klasie *ApplicationBean1* związanej z zapisem danych typu kolekcja obiektów TTytul_ksiazki i TTytul_ksiazki_na_kasecie w bazie danych

...ava | ApplicationBean1.java * | TTytul_ksiazkiController.java | persistence.xml | SQL Command 1 | SQL

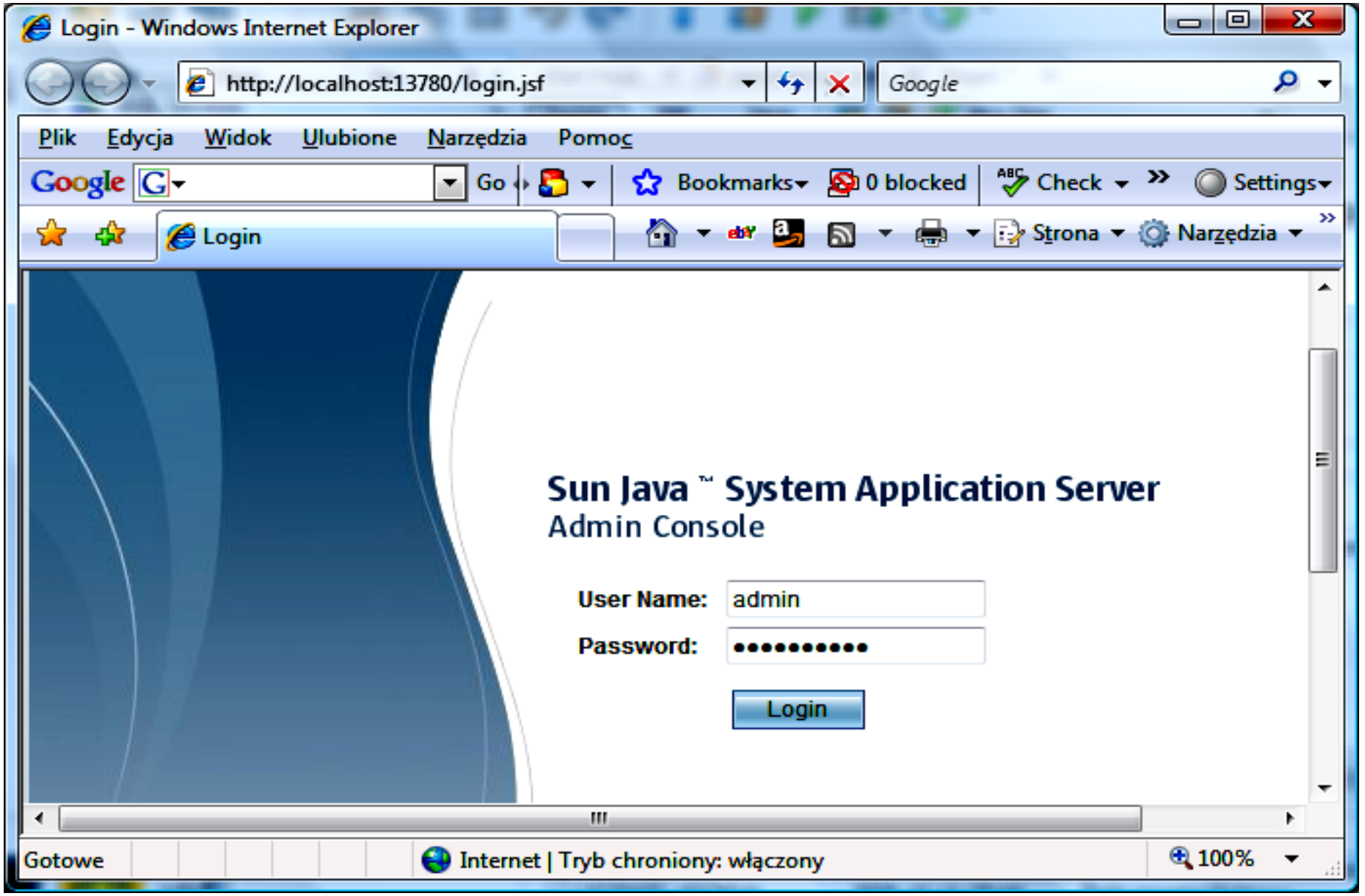
```
116     private TTytul_ksiazki tytuly[];
117     public TTytul_ksiazki[] getTytuly() {
118             return tytuly;
119         }
120     public void setTytuly(TTytul_ksiazki[] tytuly) {
121             this.tytuly = tytuly;
122         }
123     public void updateTytuls() {
124             TTytul_ksiazkiController tytulController = new TTytul_ksiazkiController();
125             tytuly = tytulController.getTTytul_ksiazkis();
126         }
127     public void zapisz_tytuly_do_bazy() {
128          // Add the new Entity to the database using UserController
129             TTytul_ksiazkiController Tytul_ksiazkiController =
130                     new TTytul_ksiazkiController();
131             Tytul_ksiazkiController.addTTytul_ksiazki(getAplikacja().getTytul_ksiazki());
132         }
```

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

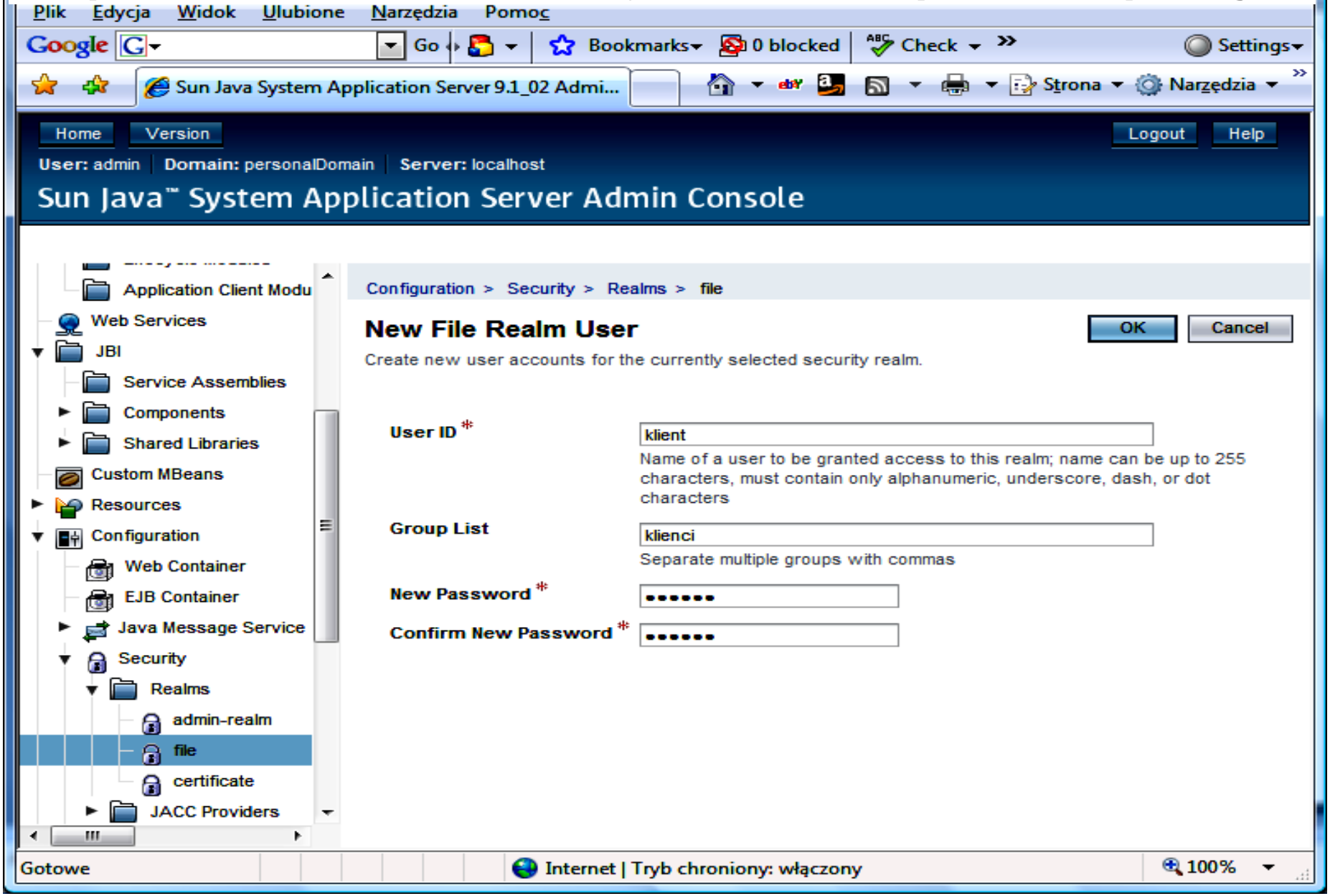


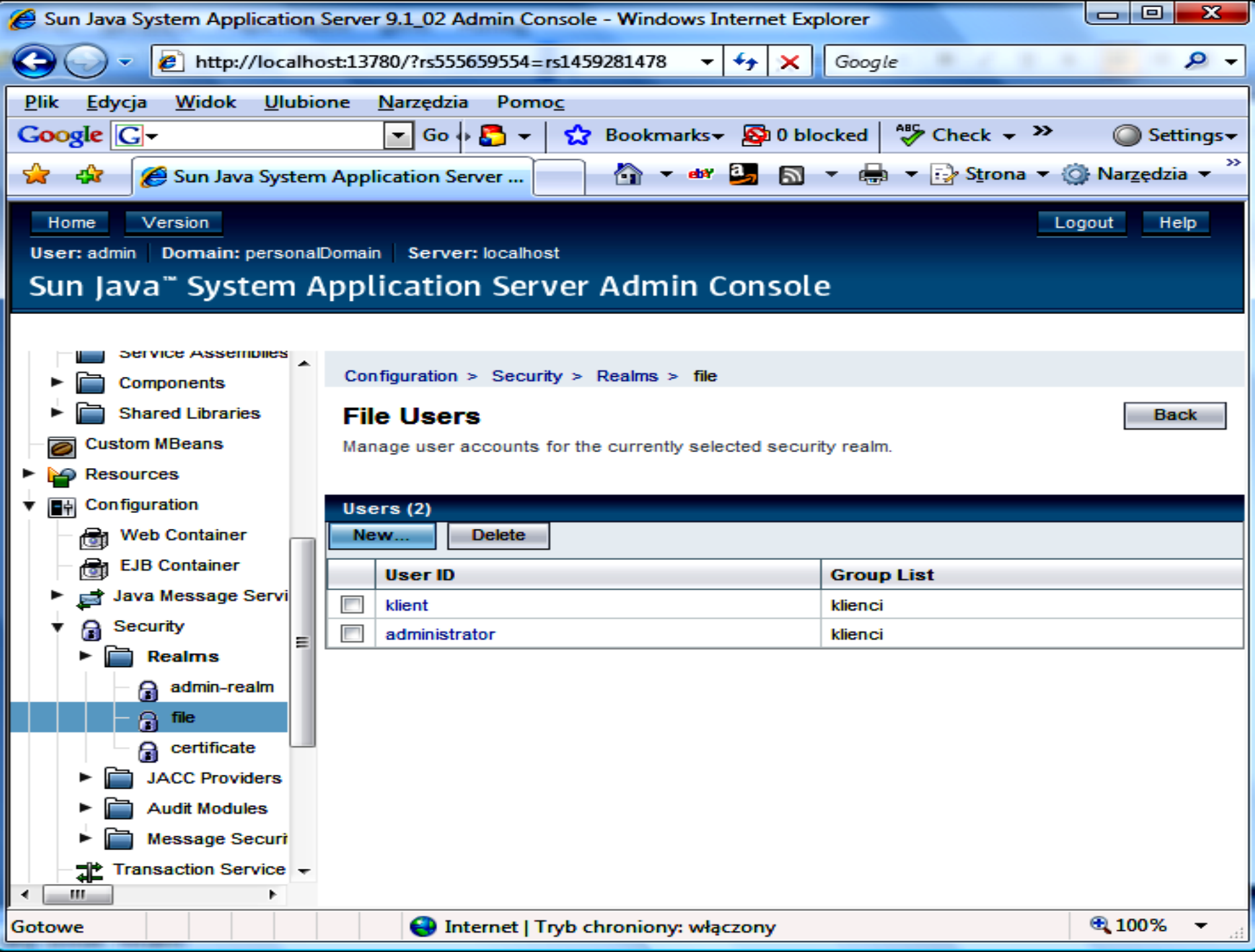
9. Uwierzytelnianie i autoryzacja oprogramowania
Drugi etap tworzenia warstwy prezentacji

(1) Dodanie użytkowników do Serwera aplikacji JavaEE

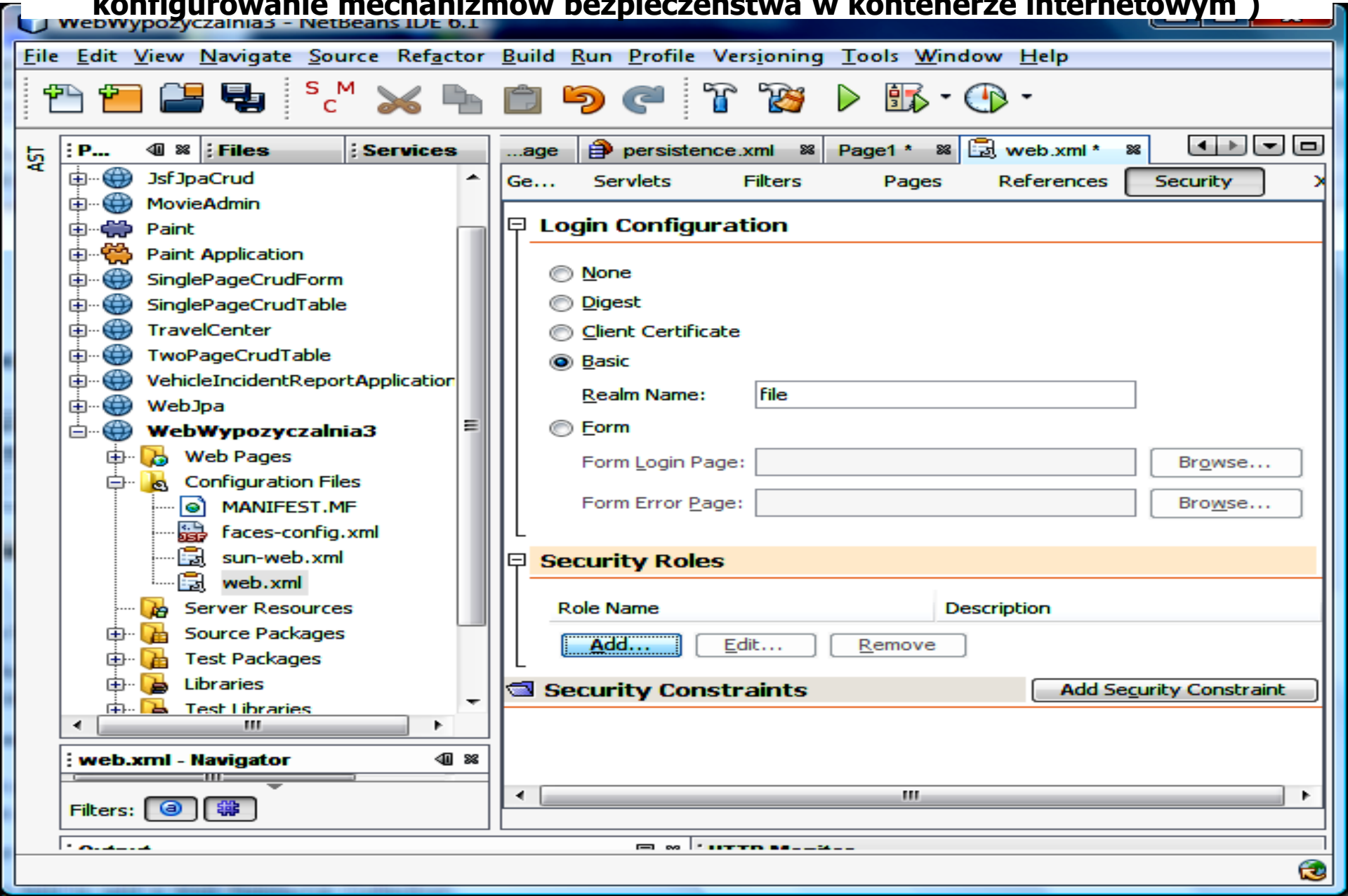


(2) Wstawienie zwykłego użytkownika z ograniczonymi uprawnieniami, które zostaną ustawione na poziomie aplikacji

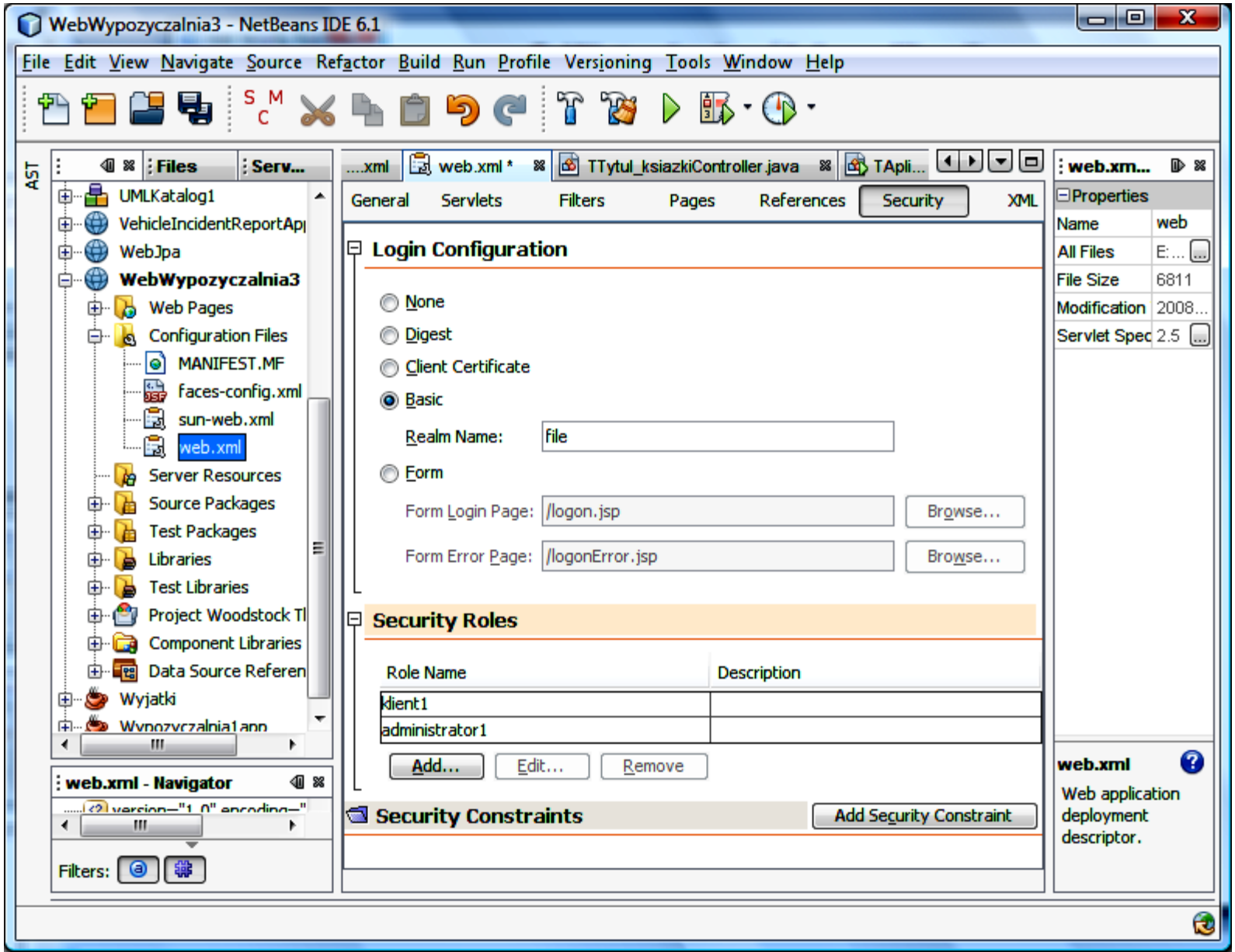



Sun Java System Application Server 9.1_02 Admin Console - Windows Internet Explorer
http://localhost:13780/?rs555659554=rs1459281478
Google
Plik Edycja Widok Ulubione Narzędzia Pomoc
Google Go Bookmarks 0 blocked Check Settings
Sun Java System Application Server ...
Strona Narzędzia
Home Version Logout Help
User: admin | Domain: personalDomain | Server: localhost
Sun Java™ System Application Server Admin Console
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
Configuration > Security > Realms > file
File Users
Back
Manage user accounts for the currently selected security realm.
Users (2)
New... Delete
User ID
Group List
klient
klienci
administrator
klienci
Gotowe
Internet | Tryb chroniony: włączony
100%

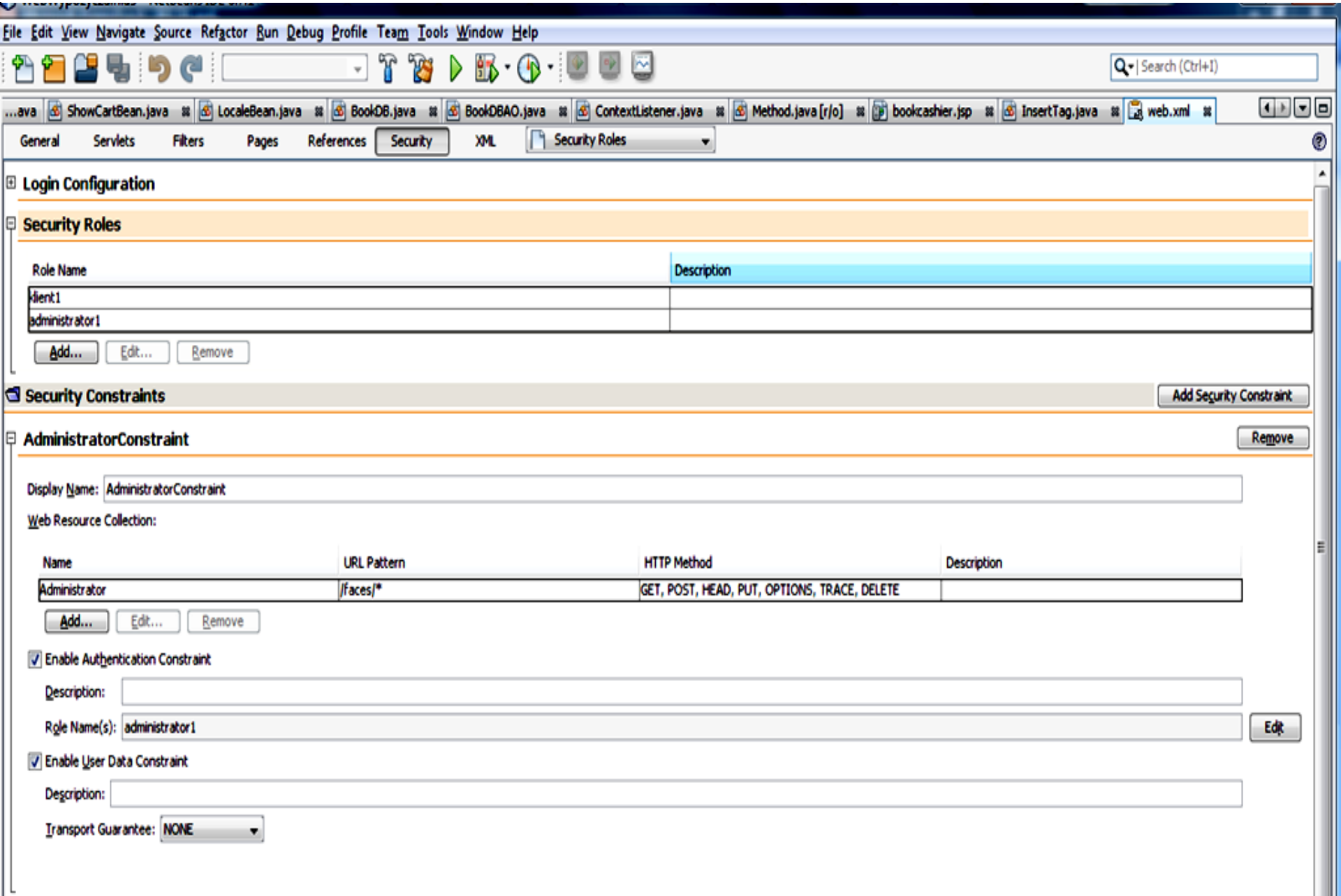
(3) Konfigurowanie logowania – deskryptor aplikacji *web.xml* po wybraniu opcji *Security -> Login Configuration* (deklaratywne konfigurowanie mechanizmów bezpieczeństwa w kontenerze internetowym)



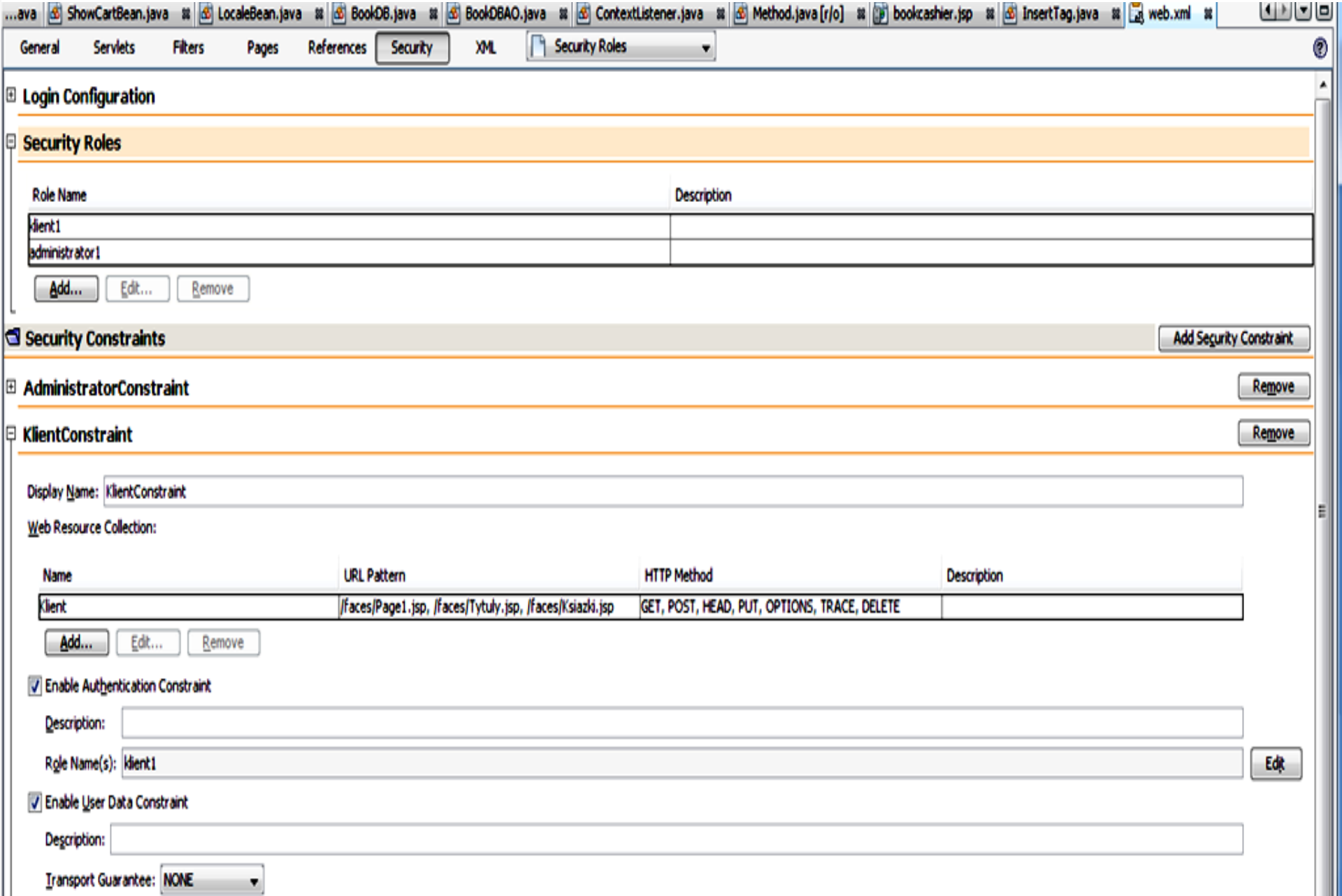
(4) Wstawianie ról użytkowników do aplikacji – deskryptor aplikacji *web.xml* po wybraniu opcji *Security-> Security Roles* (deklaratywne konfigurowanie mechanizmów bezpieczeństwa w kontenerze internetowym)



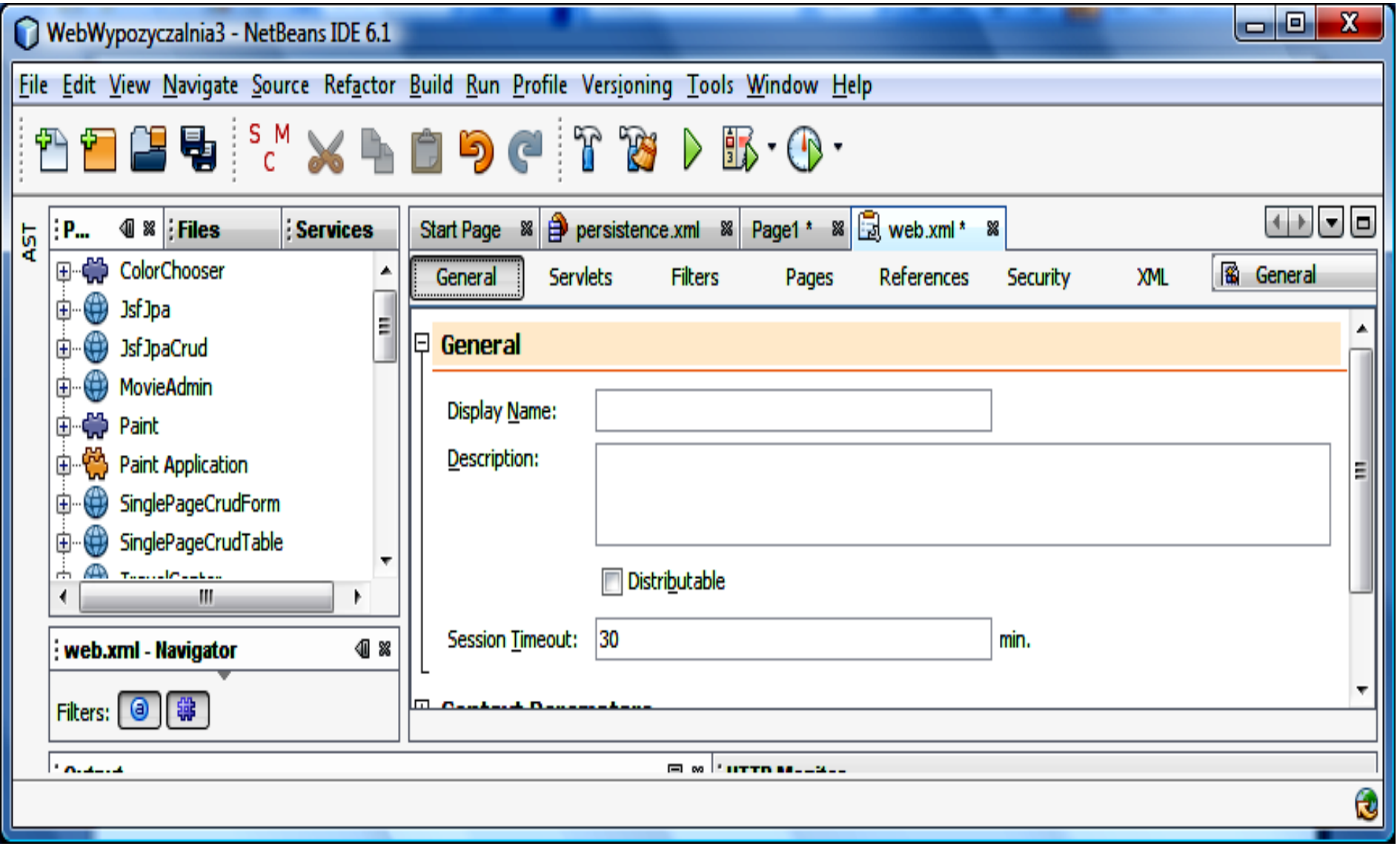
(5) Wstawianie ograniczeń dla ról użytkowników – deskryptor aplikacji *web.xml* po wybraniu opcji *Security-> Security Constraints*



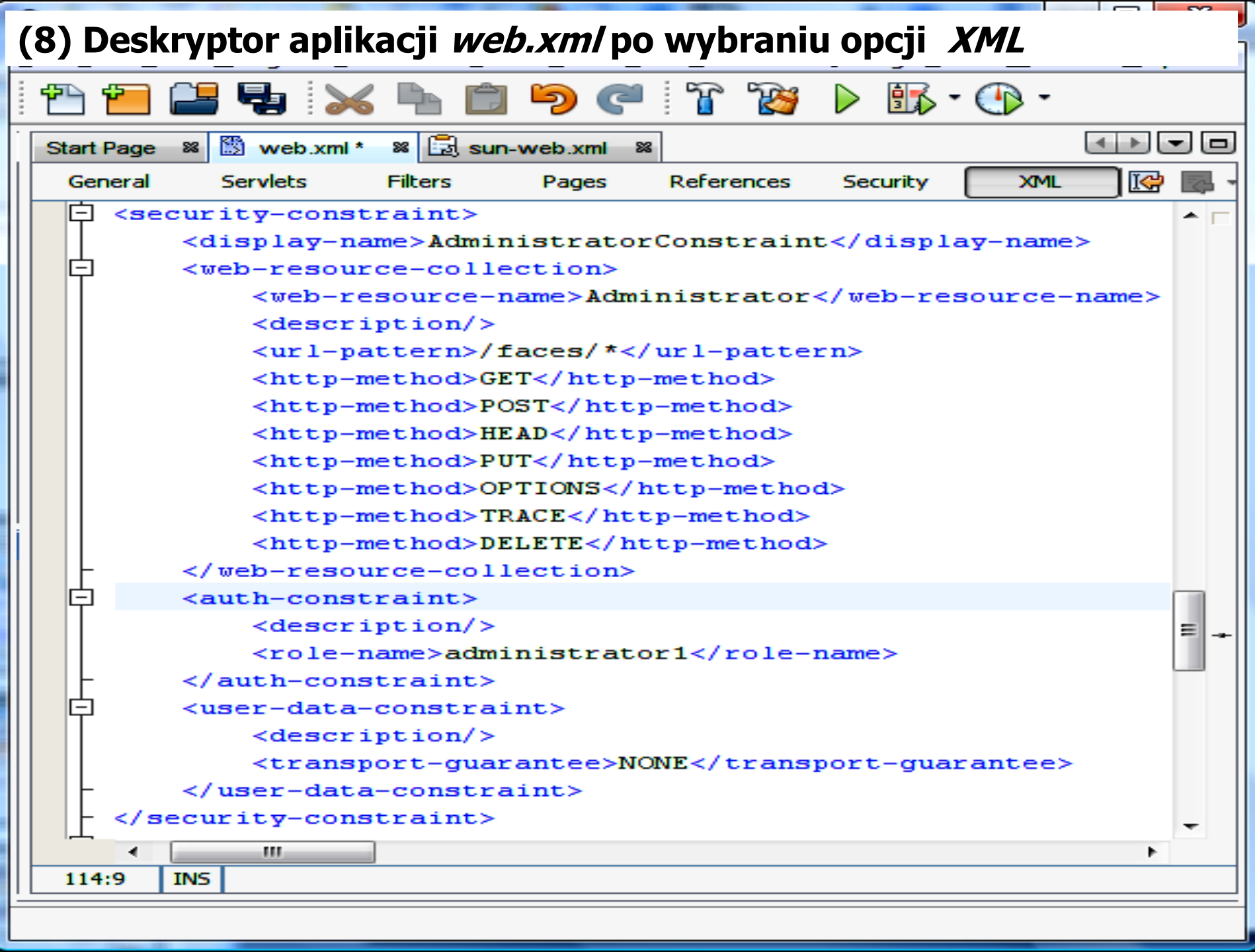
(6) Wstawianie ograniczeń dla ról użytkowników – deskryptor aplikacji *web.xml* po wybraniu opcji *Security-> Security Constraints*



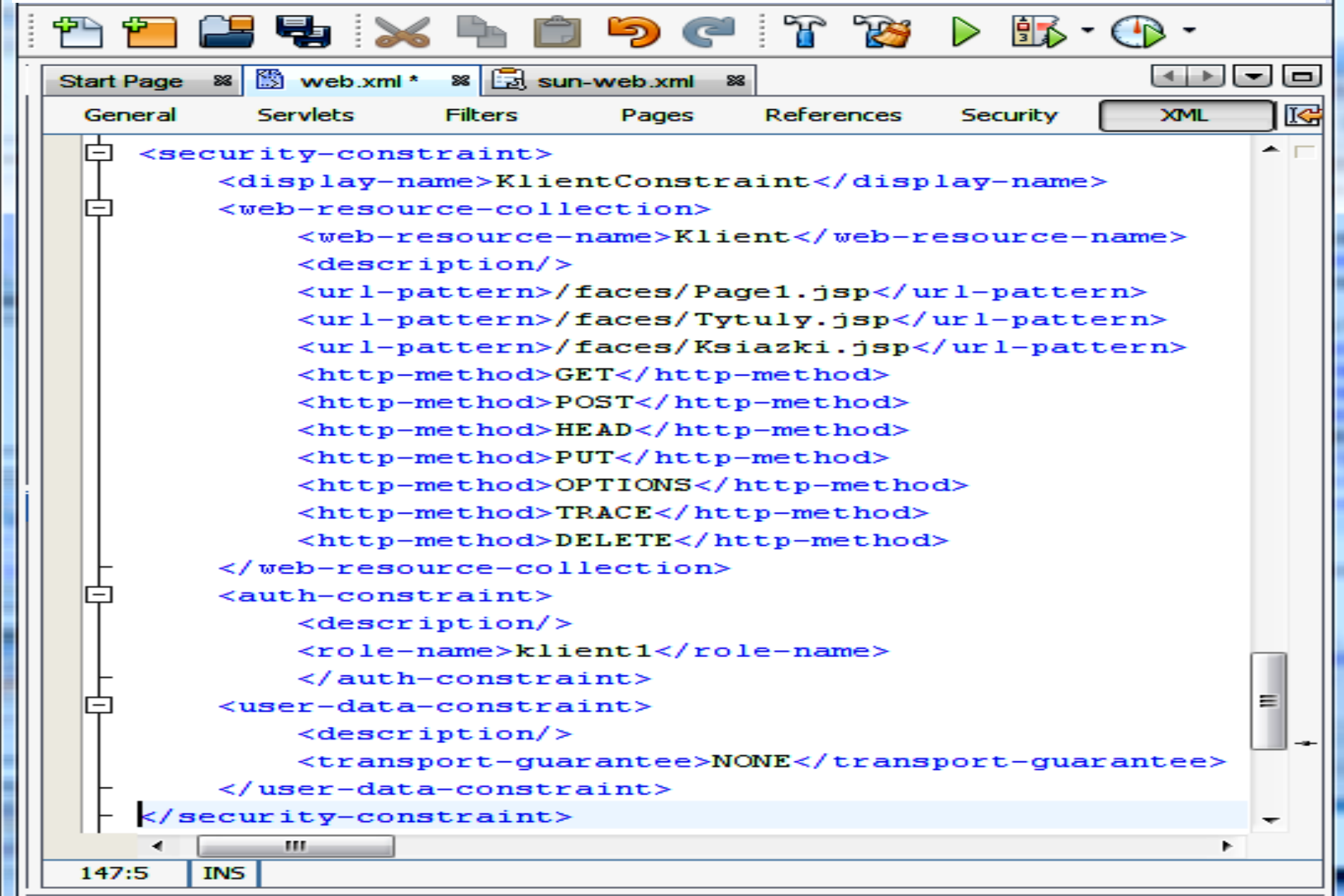
(7) Ustawienie czasu sesji



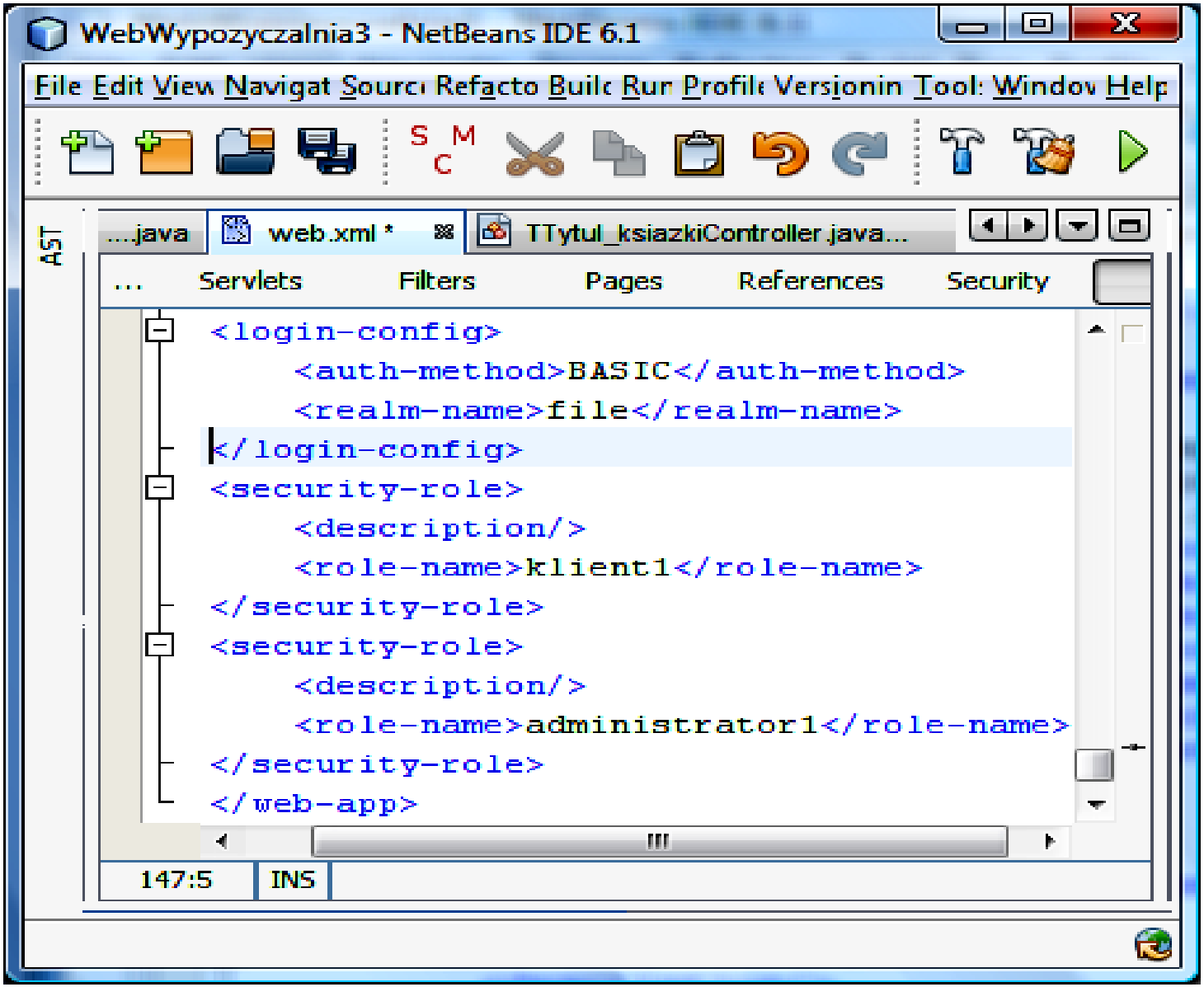
(8) Deskryptor aplikacji *web.xml* po wybraniu opcji *XML*



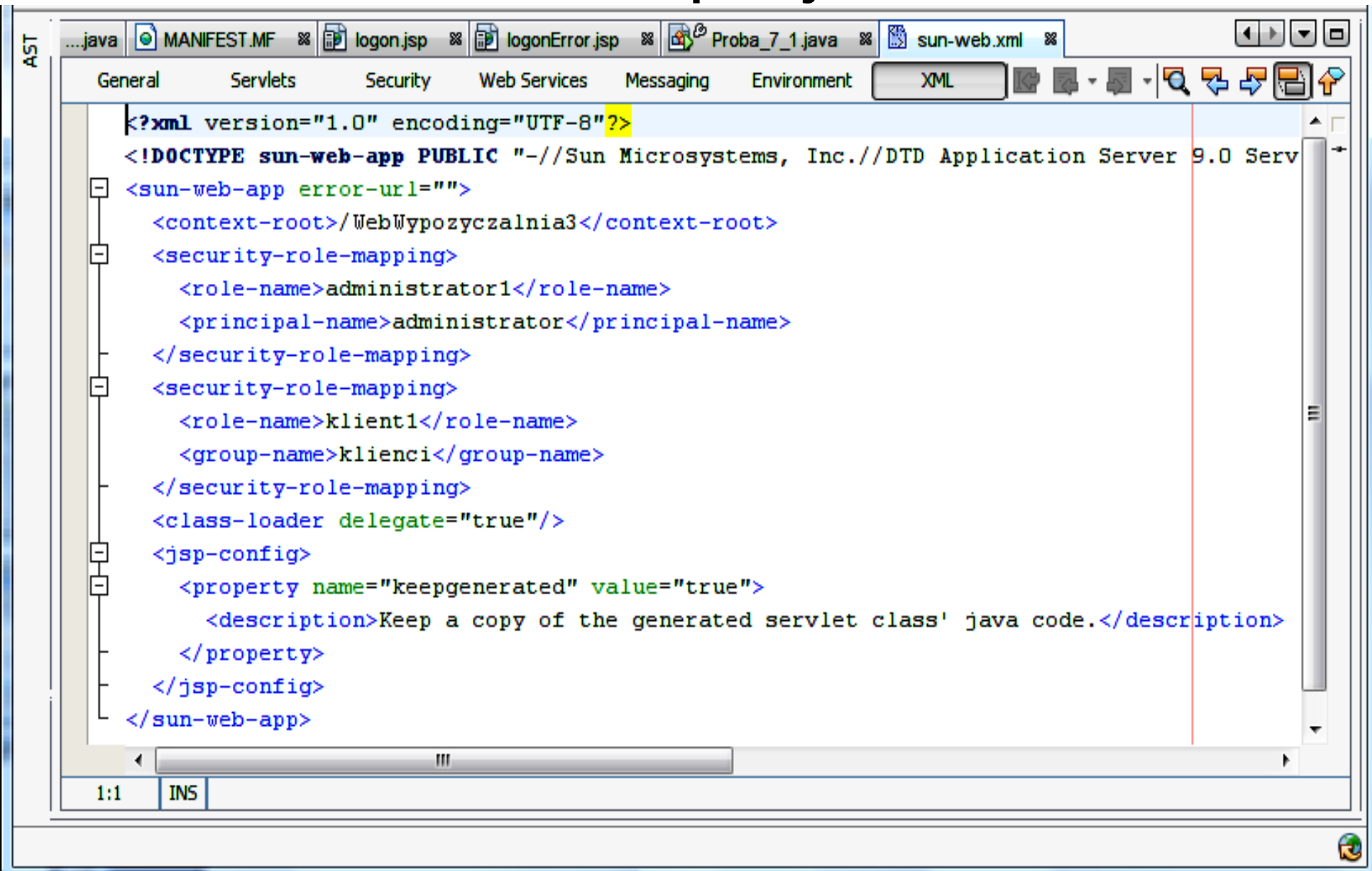
(9) Deskryptor aplikacji *web.xml* po wybraniu opcji *XML*



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

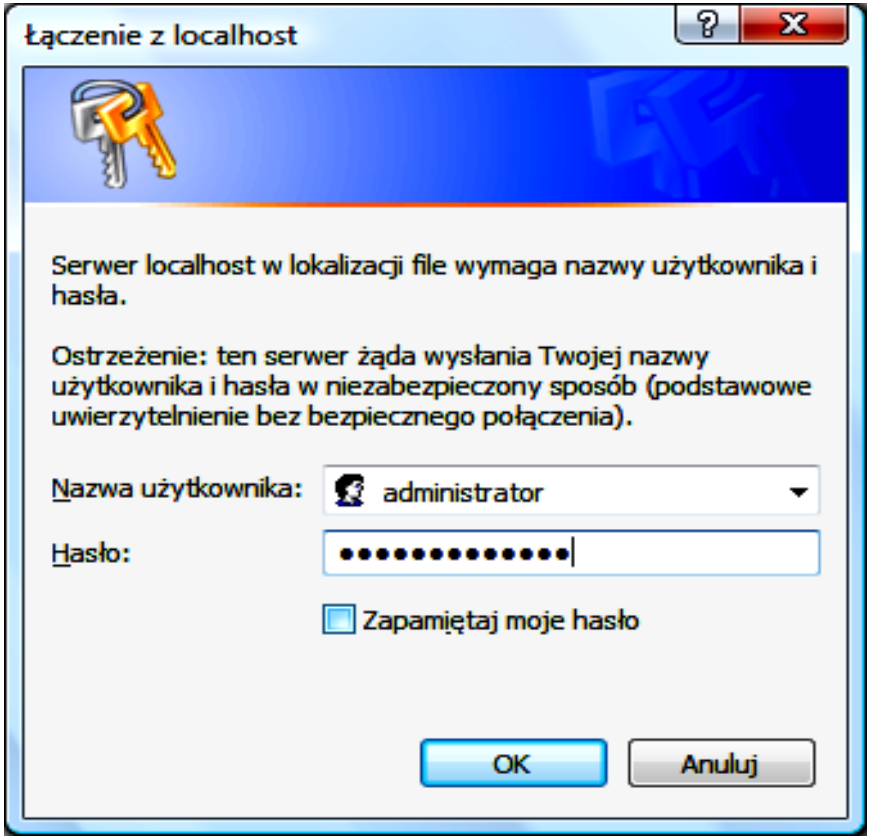


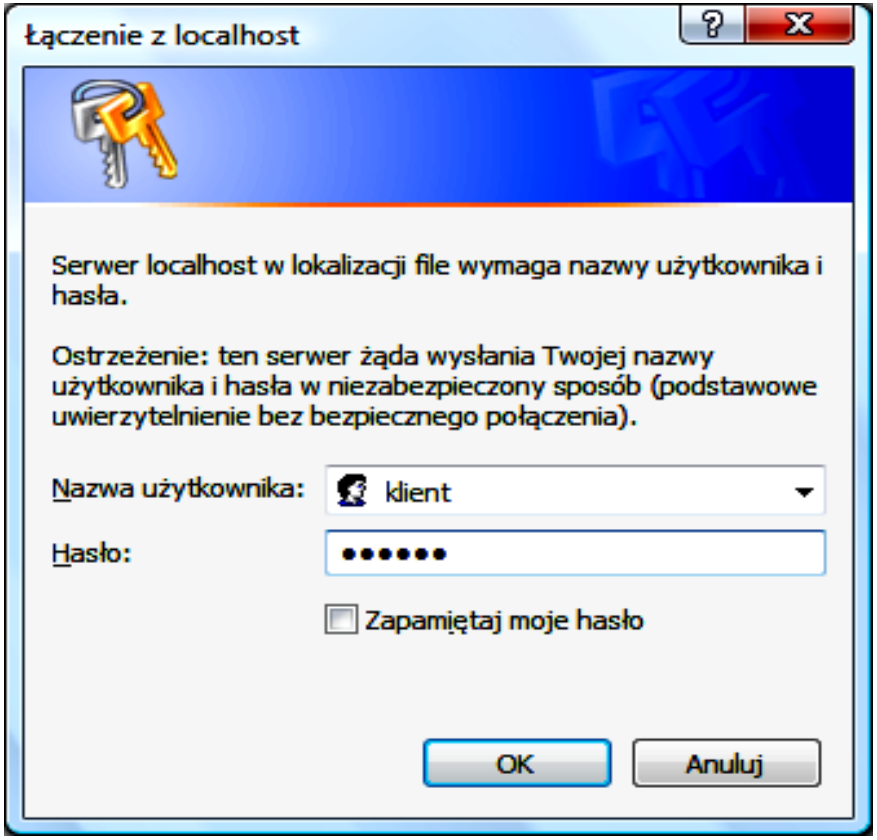
(11) Zawartość deskryptora serwera aplikacji – sun-web.xml po zmapowaniu nazwy użytkownika z serwera aplikacji do roli nadanej mu w aplikacji



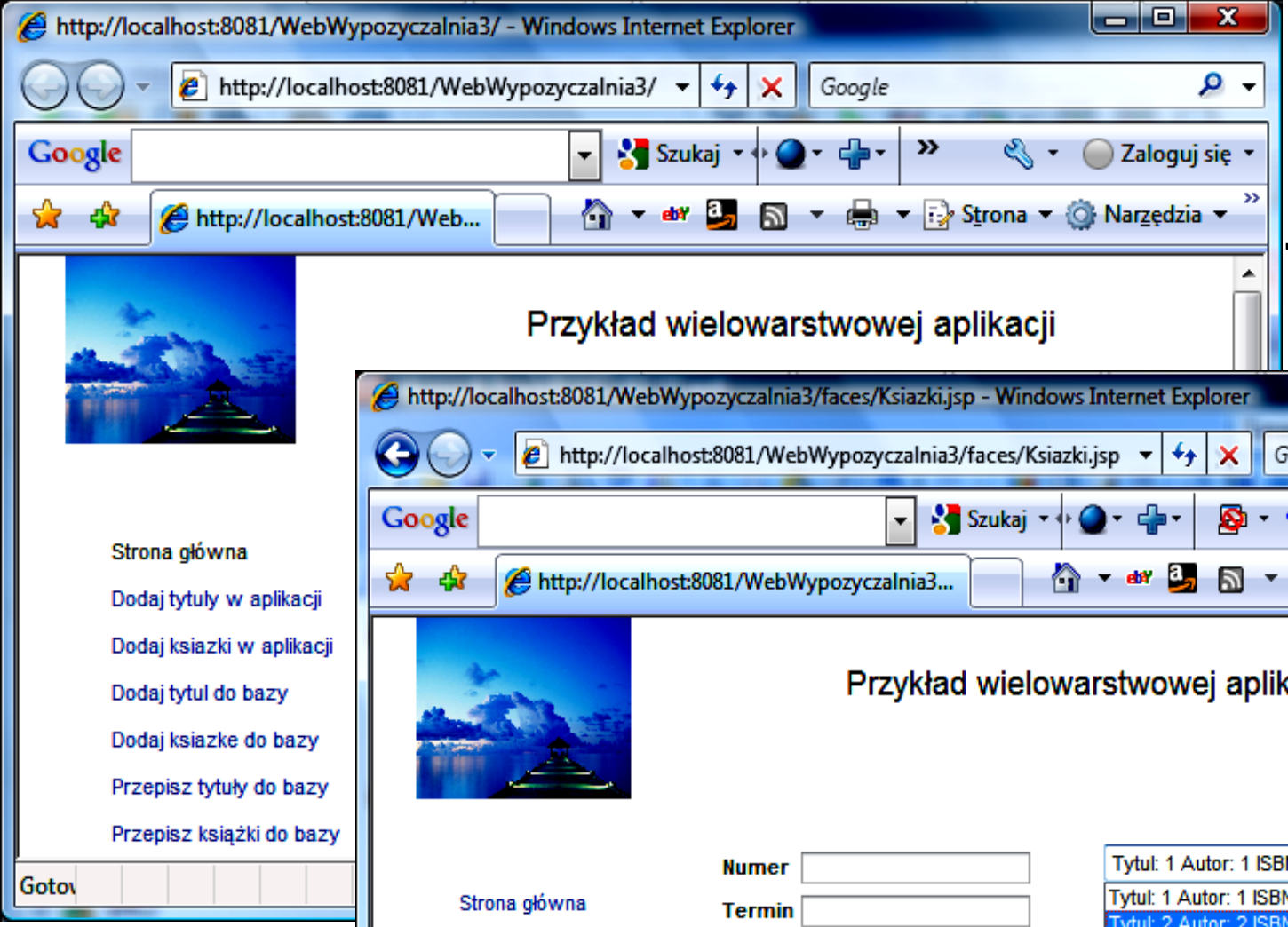
(12) Uruchomienie aplikacji w trybie uwierzytelniania

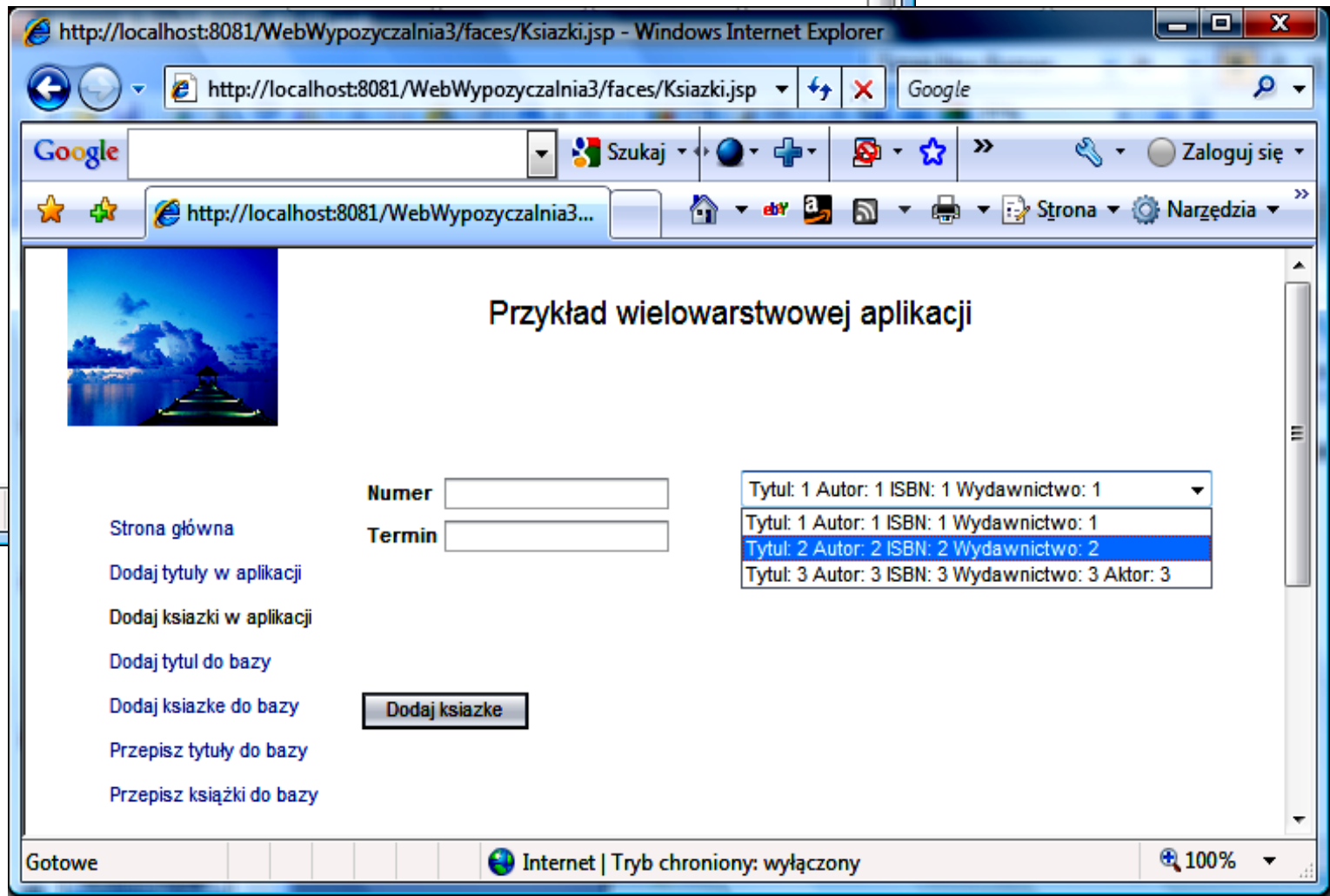
Basic-*Based Authentication* HTTP, zabezpieczenia przez role



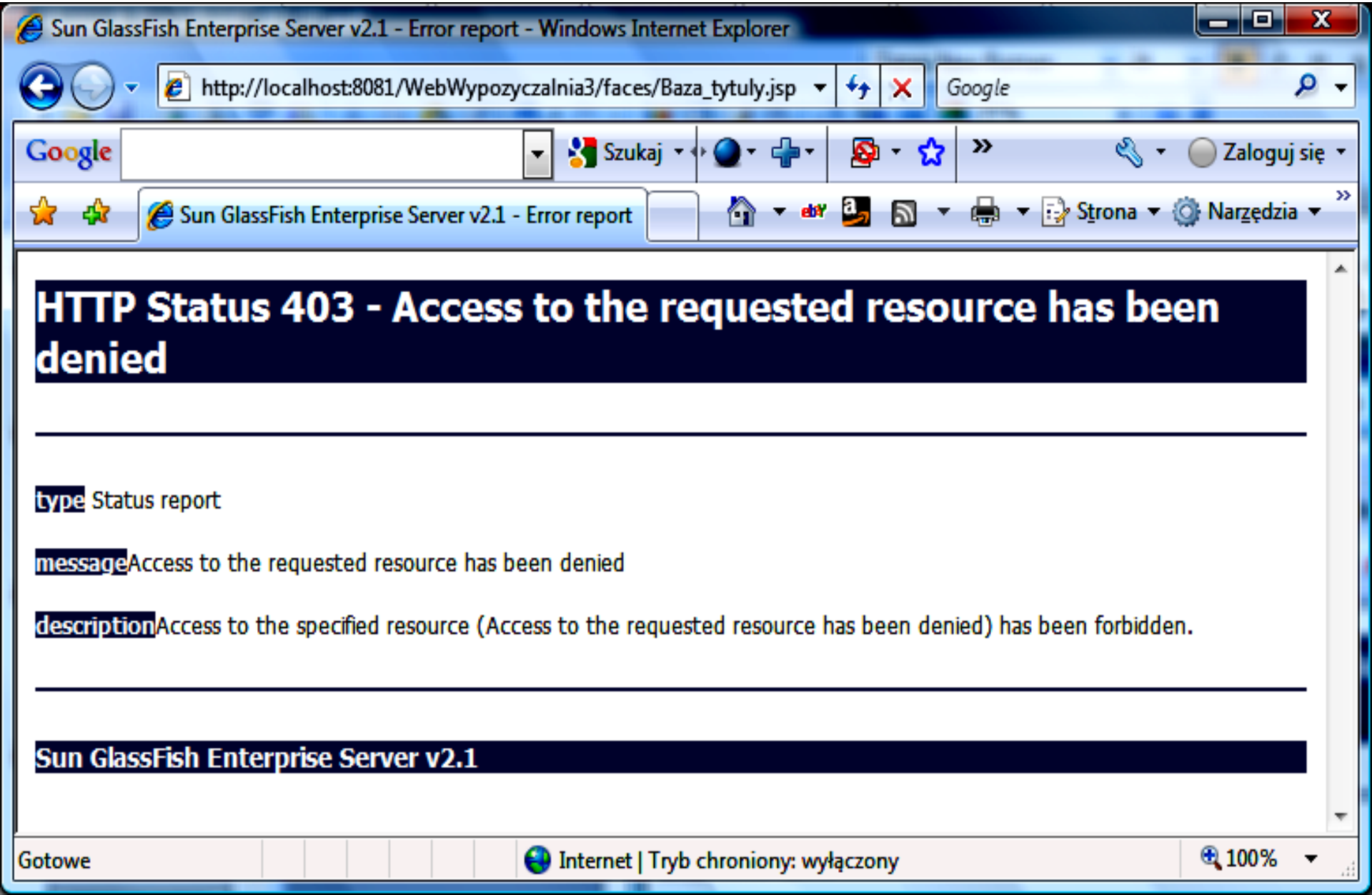


(13)Zalogowany użytkownik jako klient



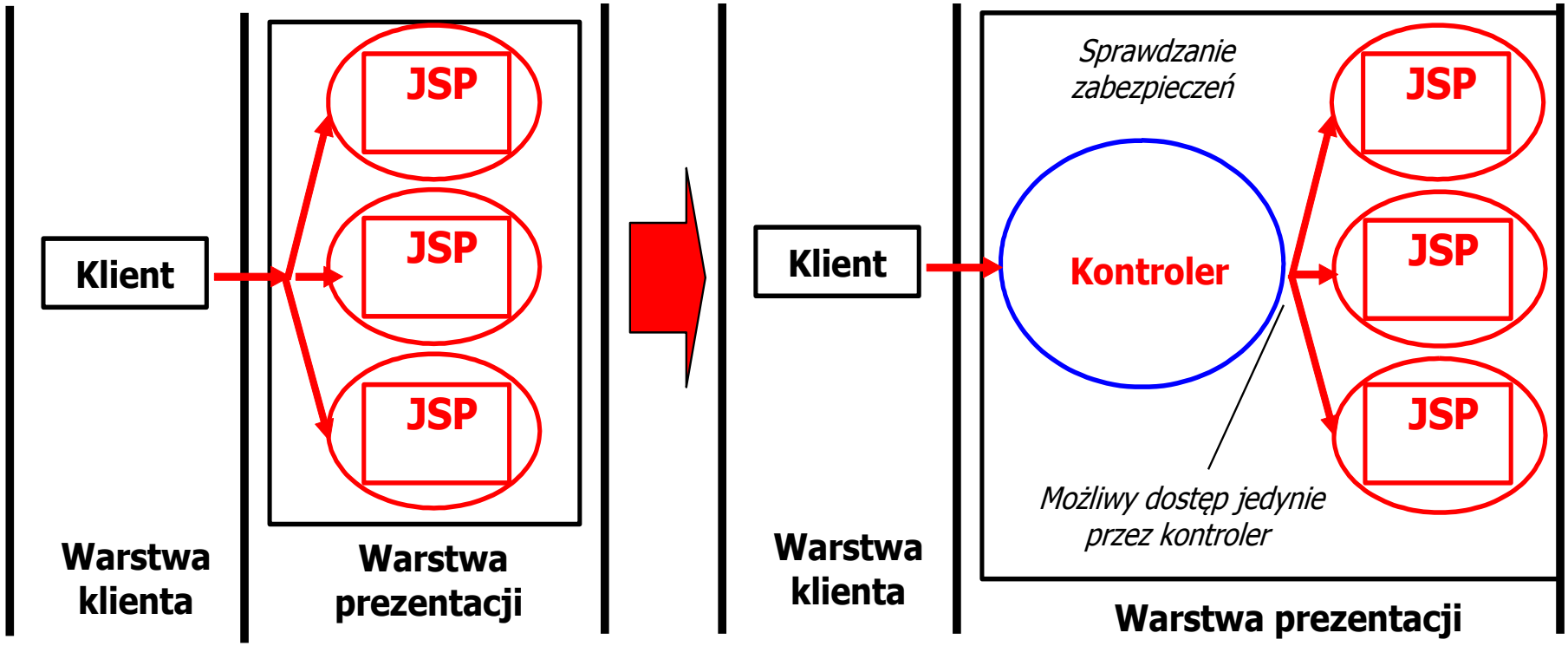


(14) Niedostępne strony dla użytkownika „klient” występującego w roli „klient1” (objęte ograniczeniem ***Web Resource Collection***)



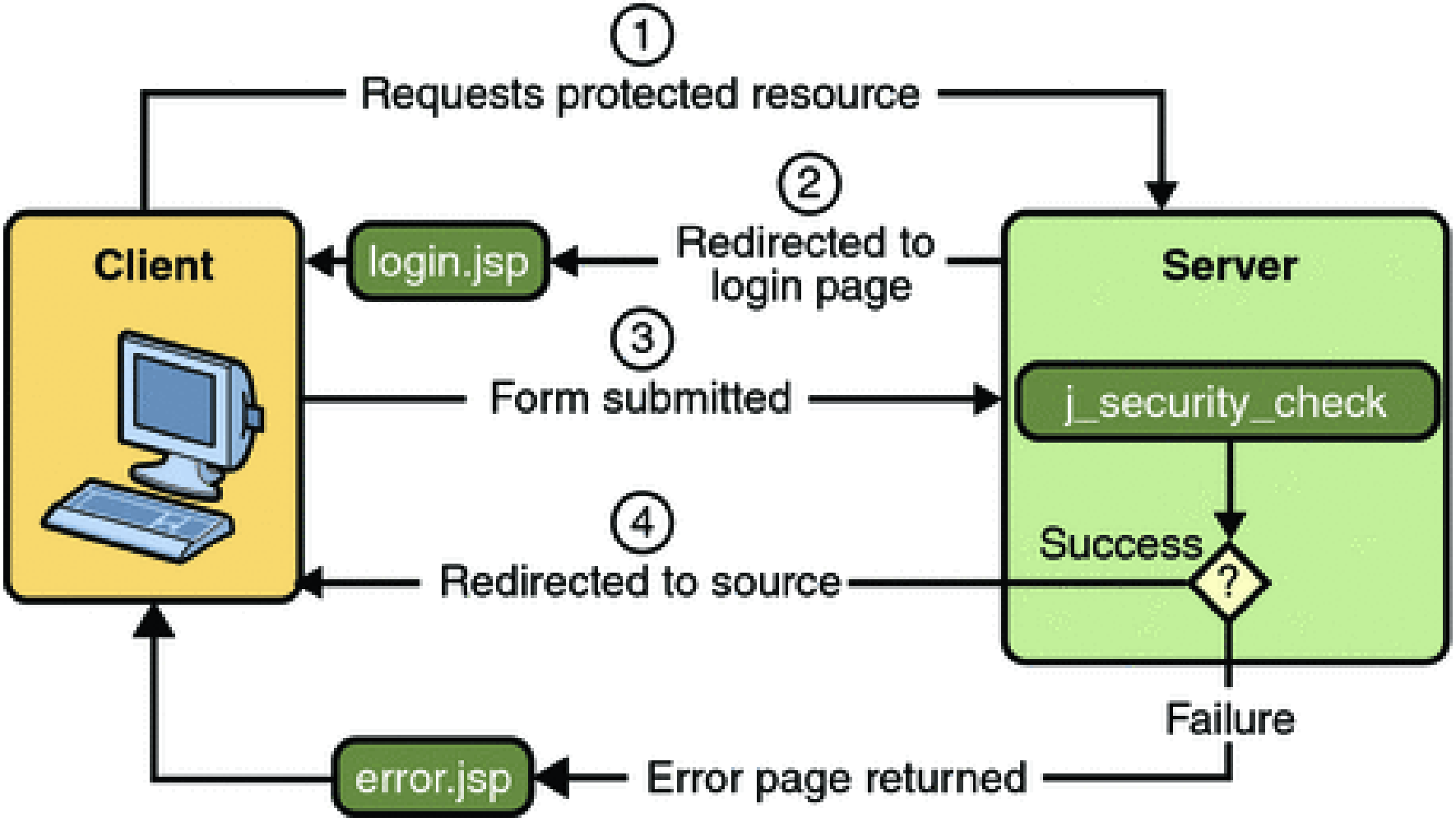
2.1. Refaktoryzacja warstwy prezentacji

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



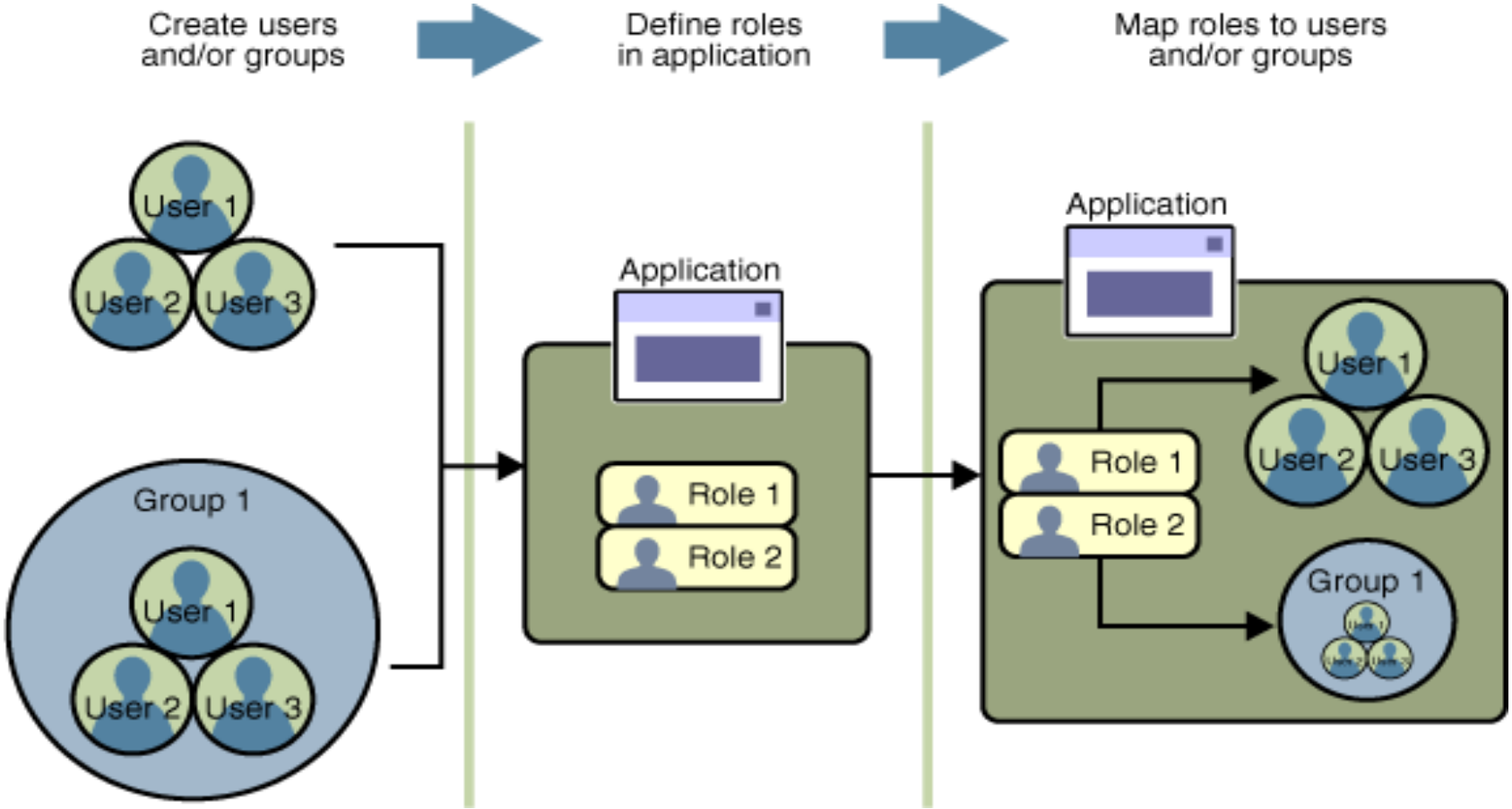
Przebieg uwierzytelniania (logowania)

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" (deskryptory aplikacji np.* ***web.xml*** dla aplikacji typu ***web)***. Deskryptory 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 deskryptor aplikacji.

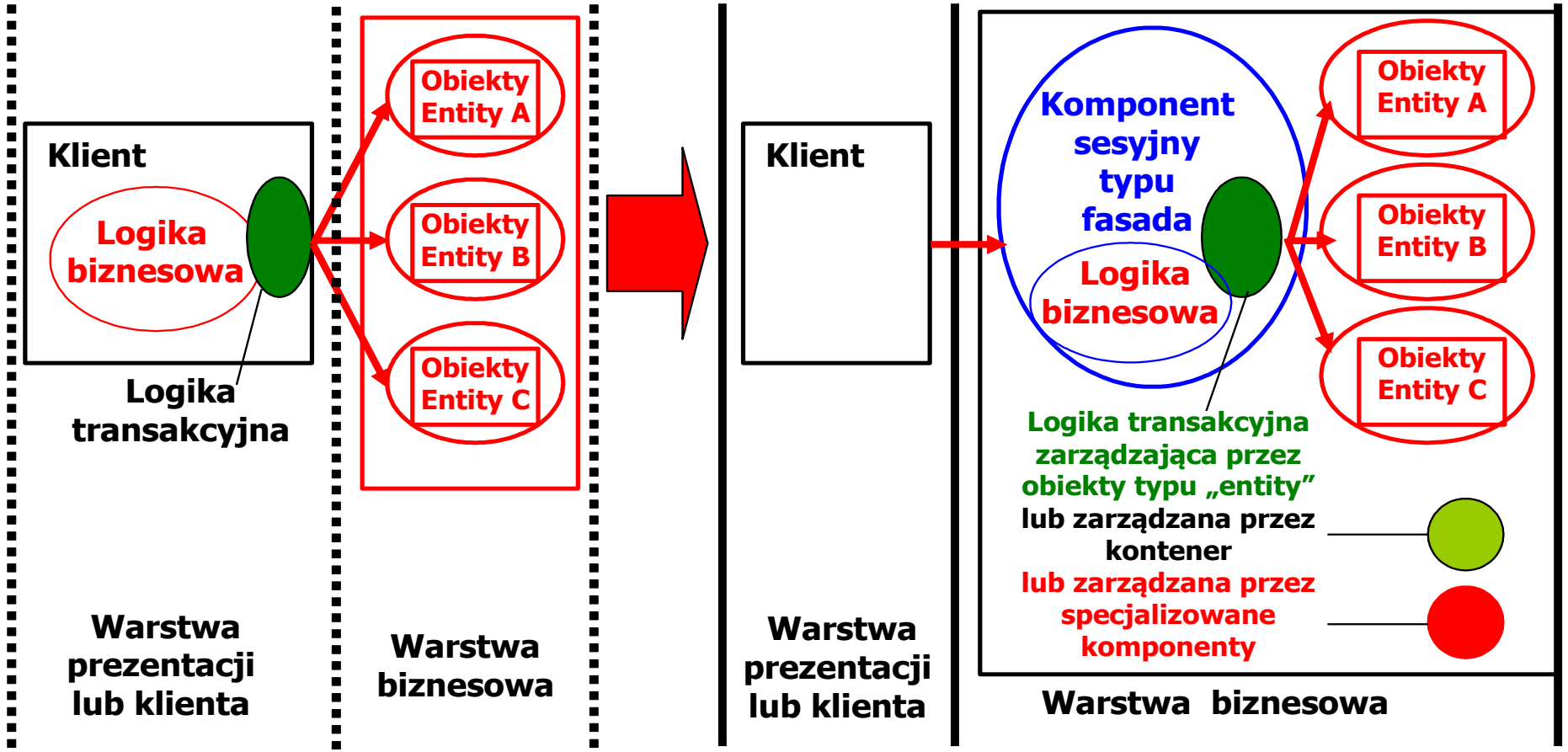
Np.

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

2.2. Refaktoryzacja warstwy biznesowej

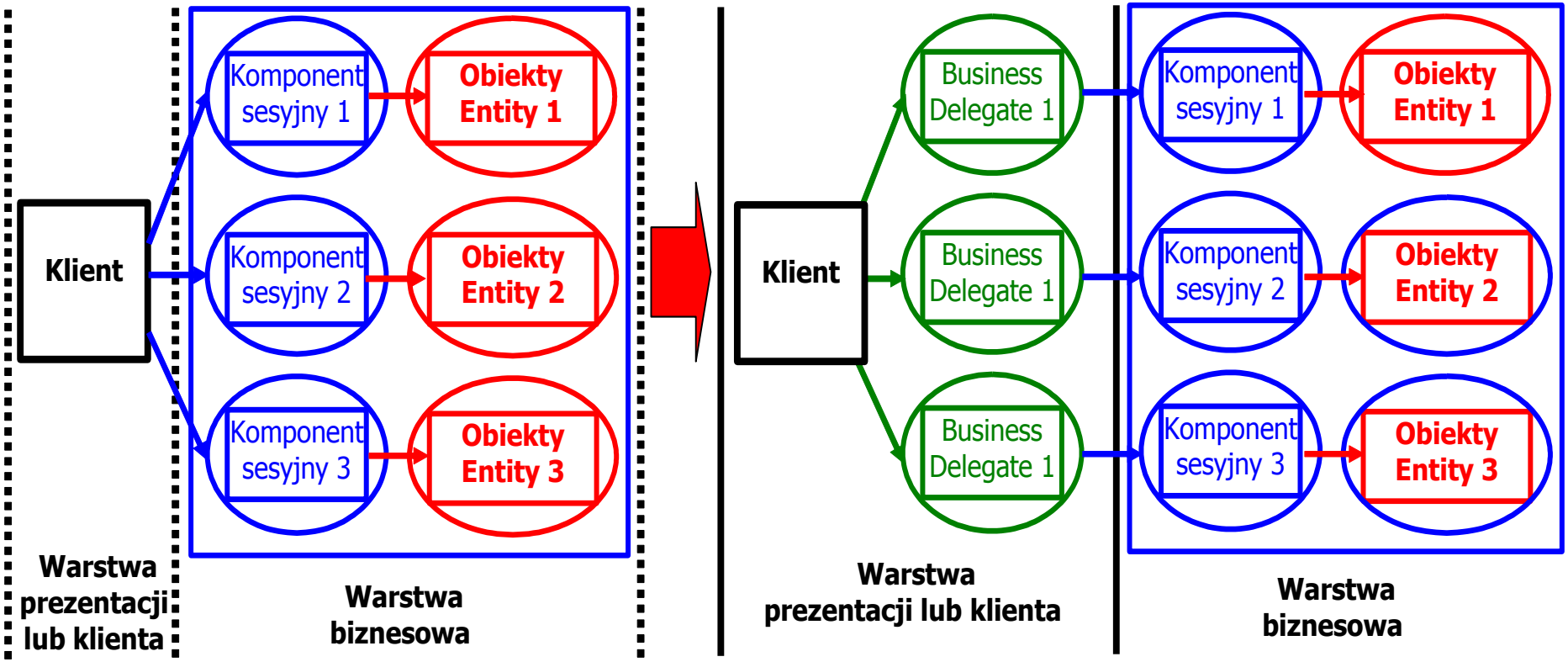
Refaktoryzacja warstwy biznesowej 1

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)**



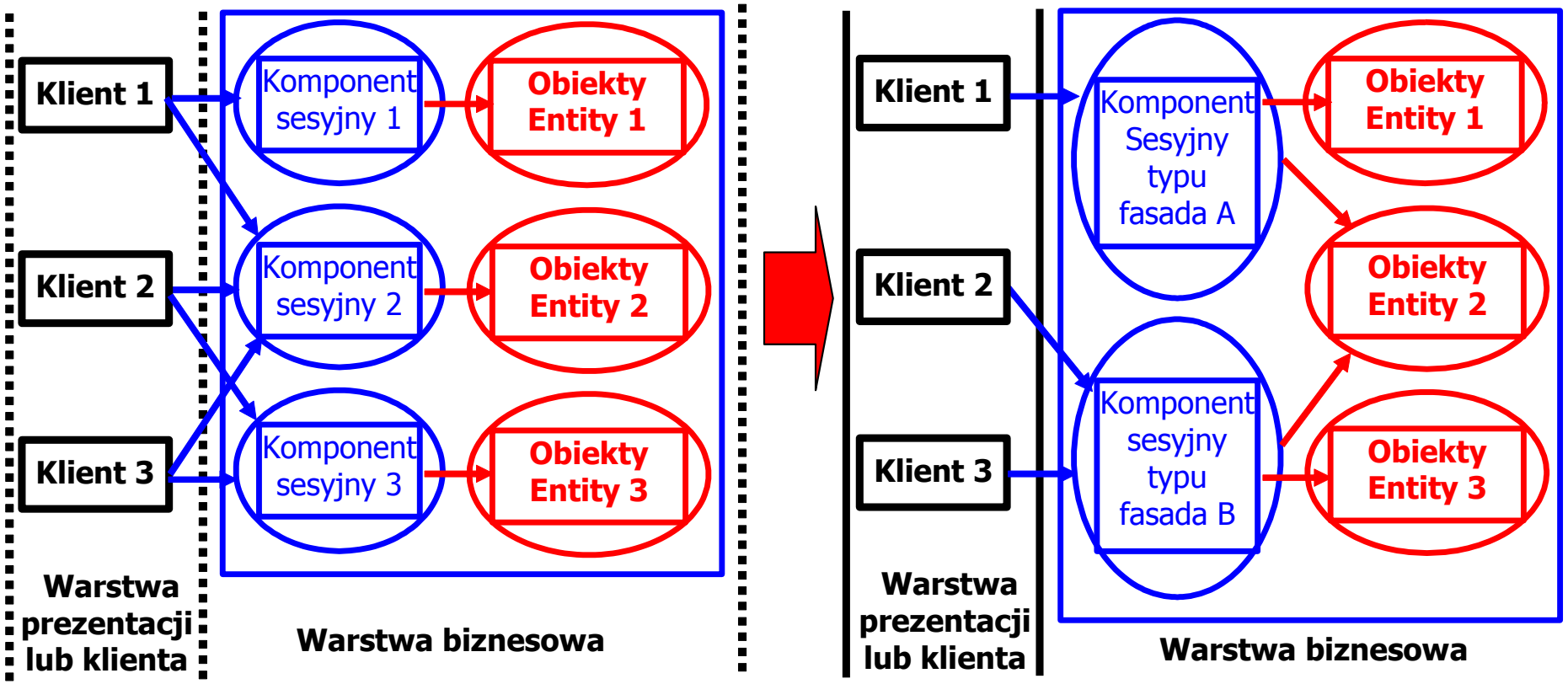
Refaktoryzacja warstwy biznesowej 2

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

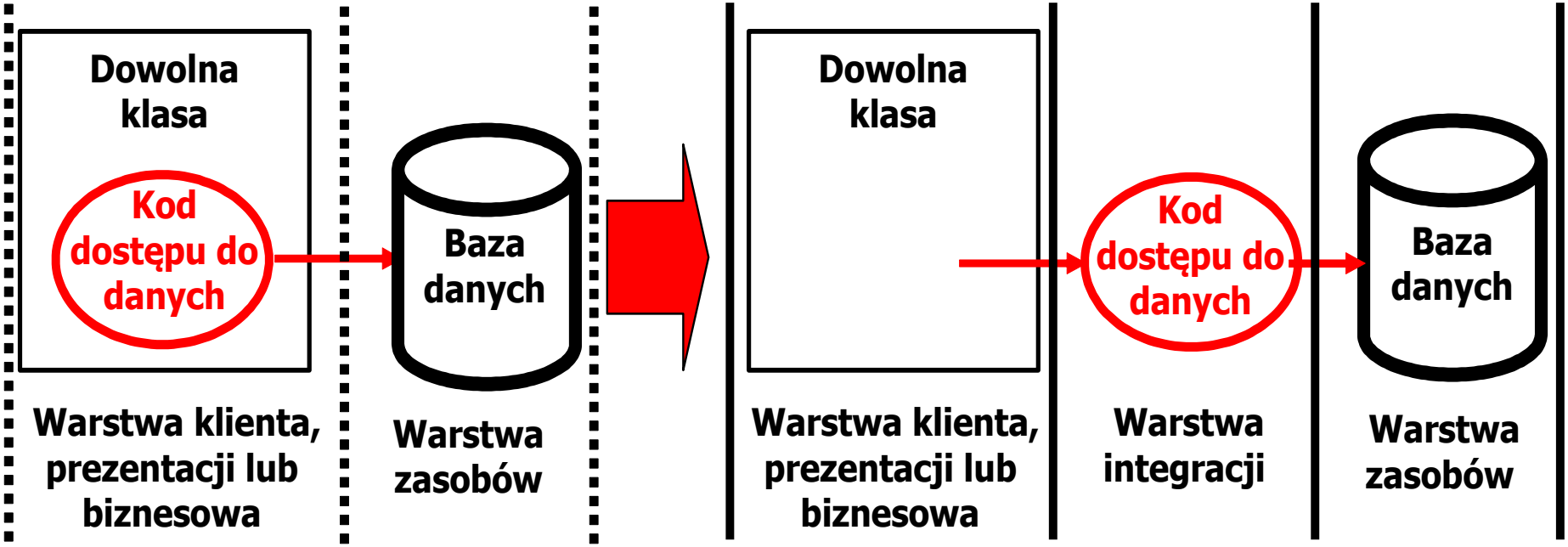
Sesyjne komponenty fasadowe typu „Control” (każdy komponent jako odrębna usługa biznesowa), hermetyzujące **obiekty danych typu „Entity”** z warstwy biznesowej są udostępniane klientom w innych warstwach. Zwykłe obiekty sesyjne są jedynie pośrednikami obiektów „**Entity**”**,** natomiast nie hermetyzują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



2.3. Tworzenie warstwy integracji

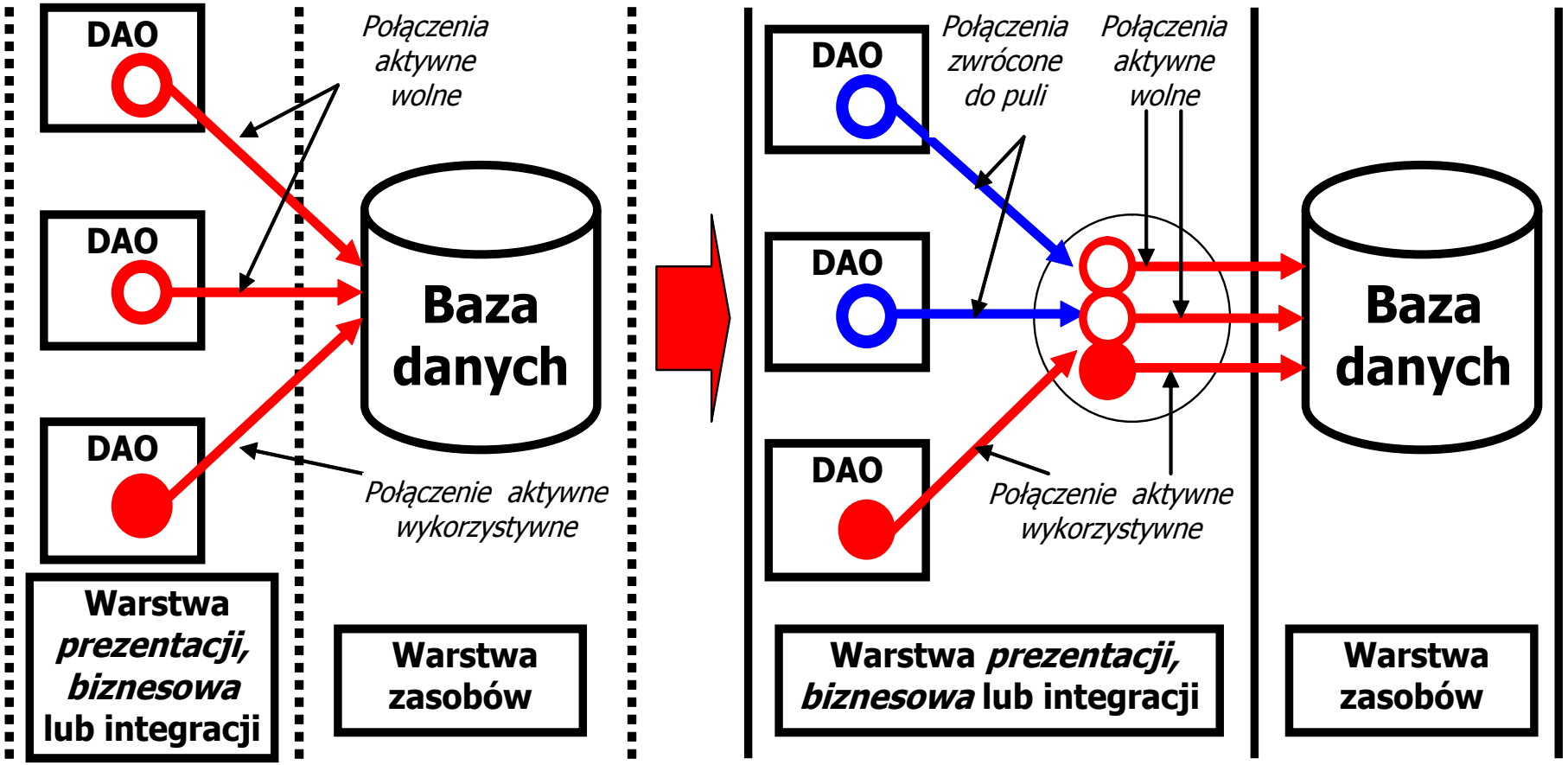
Wydzielanie kodu dostępu do danych

- **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**

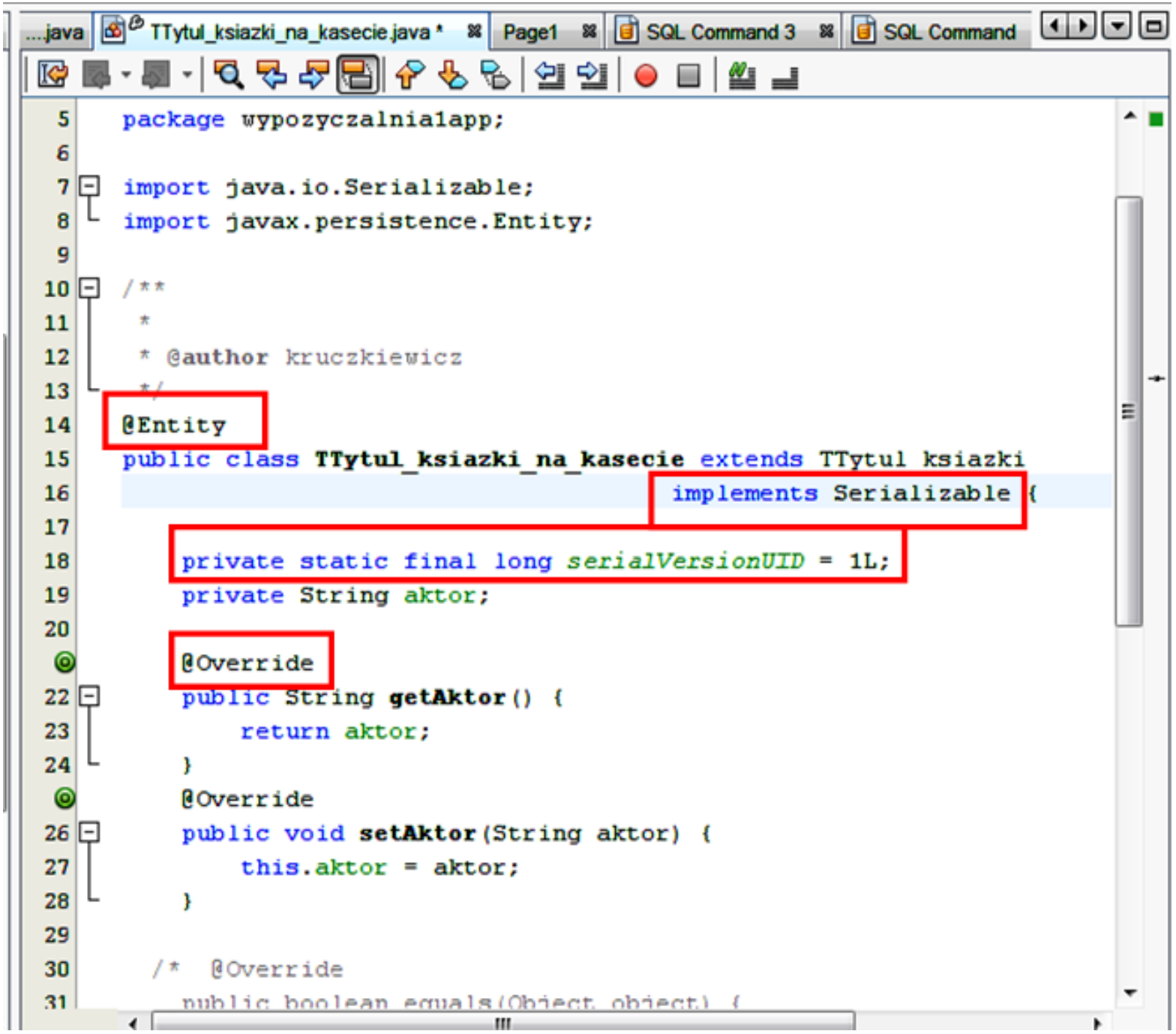


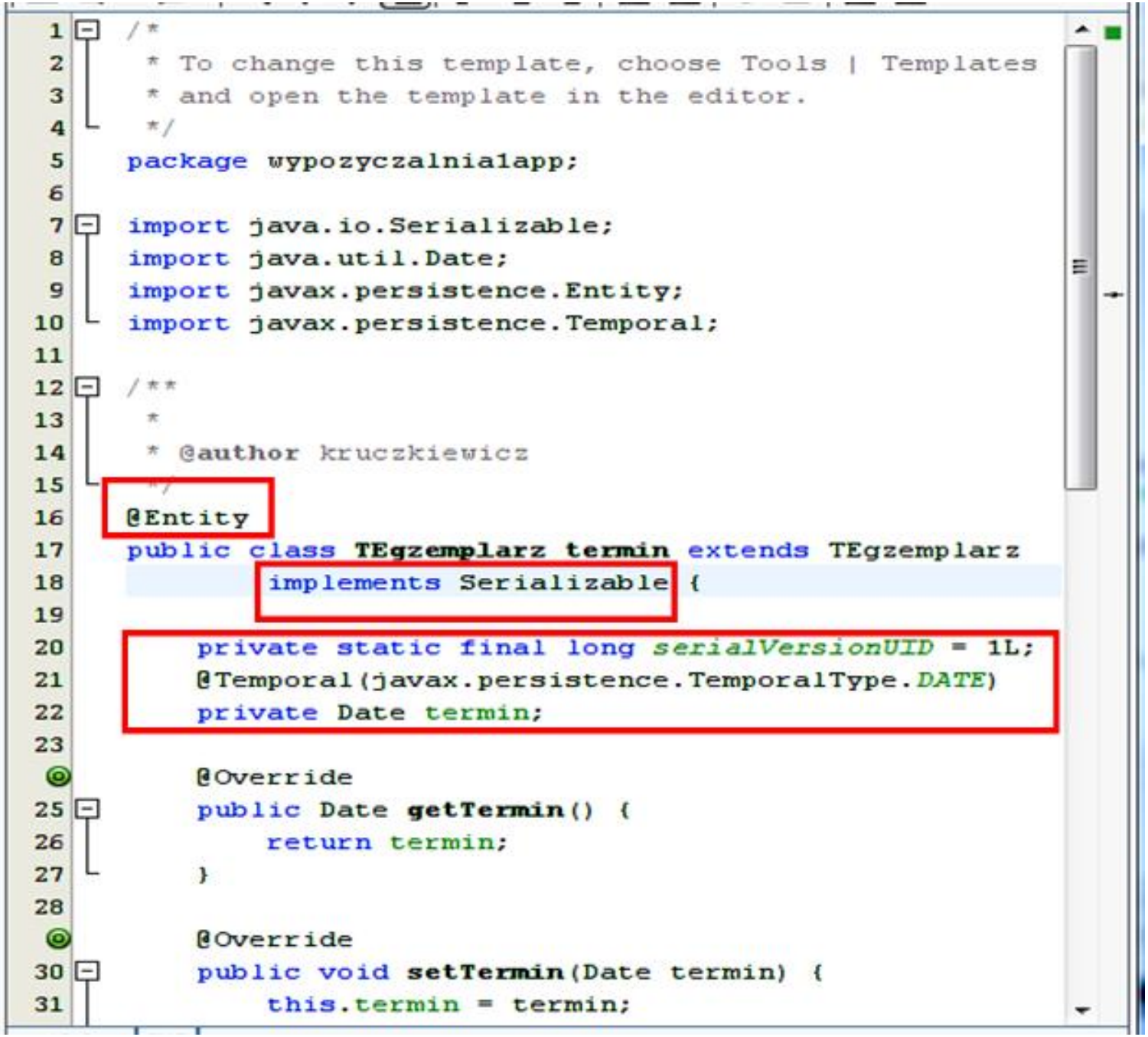
```
public boolean removeTTytul_ksiazkis (TTytul_ksiazki TTytul_ksiazki)
{
    EntityManager em = getEntityManager();
    try {
        em.getTransaction().begin();
        TTytul_ksiazki TTytul_ksiazkix =
            em.find(TTytul_ksiazki.class, TTytul_ksiazki.getId());
        em.remove(TTytul_ksiazkix);
        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_ksiazki_na_kasecie.java *
Page1
SQL Command 3
SQL Command
5 package wypozyczalnia1app;
6
7 import java.io.Serializable;
8 import javax.persistence.Entity;
9
10 /**
11 *
12 * @author kruczkiewicz
13 */
14 @Entity
15 public class TTytul_ksiazki_na_kasecie extends TTytul_ksiazki
16 implements Serializable {
17
18 private static final long serialVersionUID = 1L;
19 private String aktor;
20
@Override
22 public String getAktor() {
23 return aktor;
24 }
@Override
26 public void setAktor(String aktor) {
27 this.aktor = aktor;
28 }
29
30 /* @Override
31 public boolean equals(Object object) {



```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package wypozyczalnia1app;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.Entity;
import javax.persistence.Temporal;

/**
 *
 * @author kruczkiewicz
 */
@Entity
public class TEgzemplarz termin extends TEgzemplarz
        implements Serializable {

    private static final long serialVersionUID = 1L;
    @Temporal(javax.persistence.TemporalType.DATE)
    private Date termin;

    @Override
    public Date getTermin() {
        return termin;
    }

    @Override
    public void setTermin(Date termin) {
        this.termin = termin;
```

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

- **do pobierana 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  {    tytulyaplikacjaPanel.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 initialiation code here
          Managed Component Initialization
          // *after* managed components are initialized
          // TODO - add your own initialization code here
          updateTytuls();
          updateKsiazkis();
          updateAplikacja();
          przygotujtytuly();
      }
    private TAplikacja aplikacja = new TAplikacja();
    public TAplikacja getAplikacja() {
          return aplikacja;
    }
    public void setAplikacja(TAplikacja aplikacja) {
          this.aplikacja = aplikacja;
    }
    public void updateAplikacja() {
     for (int i = 0; i < tytuly.length; i++)
          aplikacja.getTytul_ksiazki().add(tytuly[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]); }
        } }   }
```

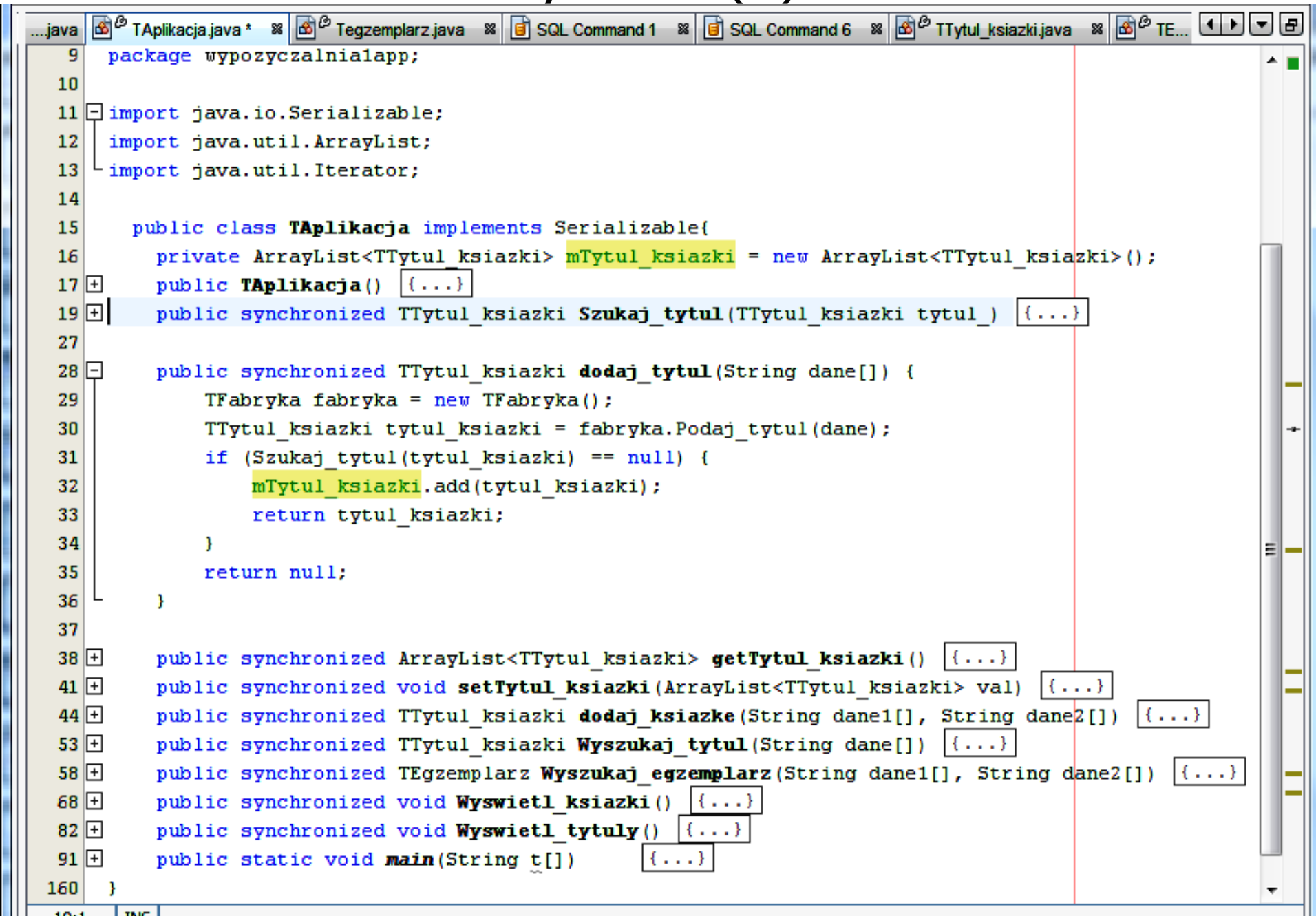
Nazwa metody po zestandaryzowaniu nazw dla atrybutu mTytul_ksiazki w klasie TEgzemplarz

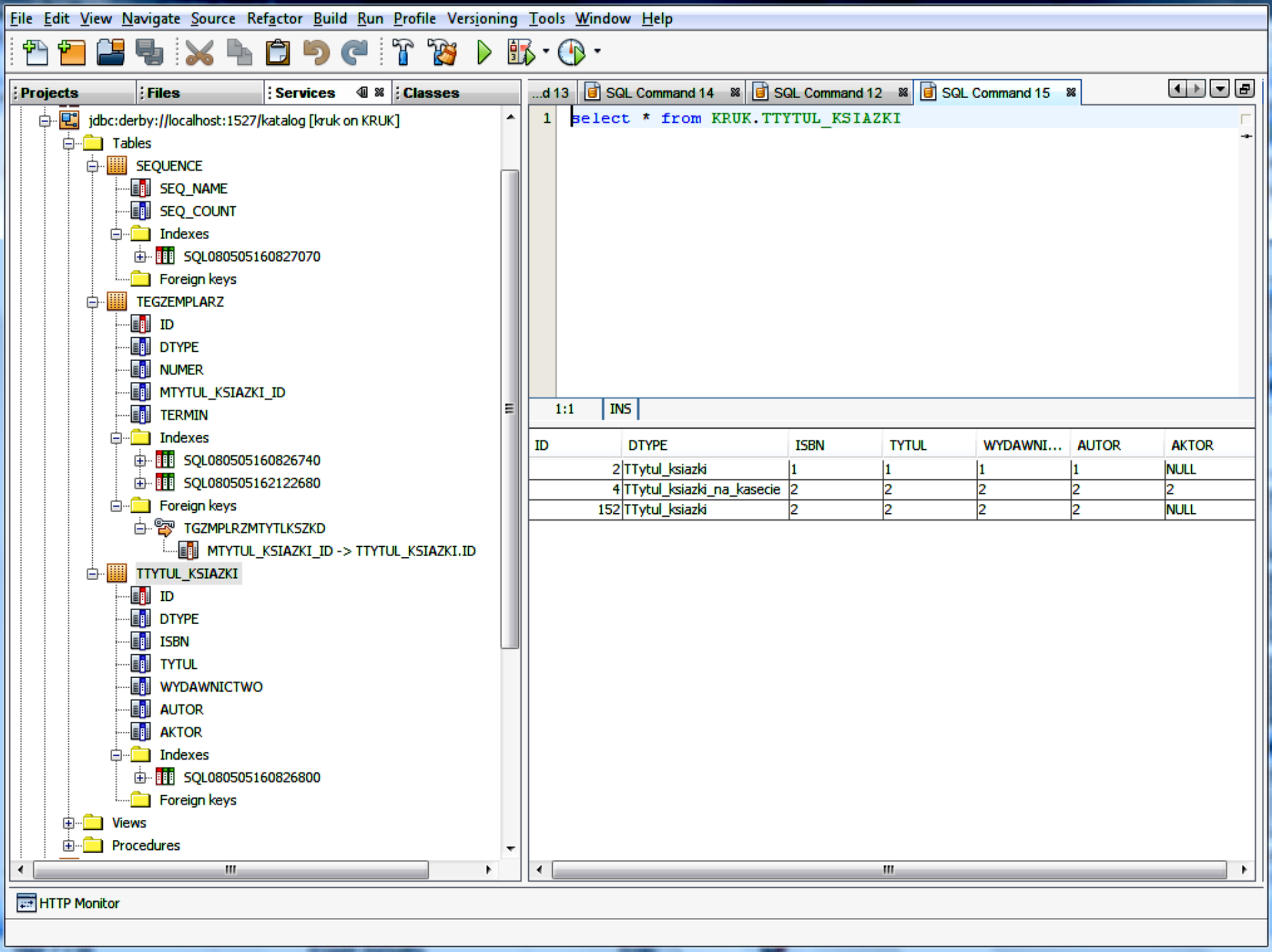
Stąd:

getMTytul_ksiazki oraz

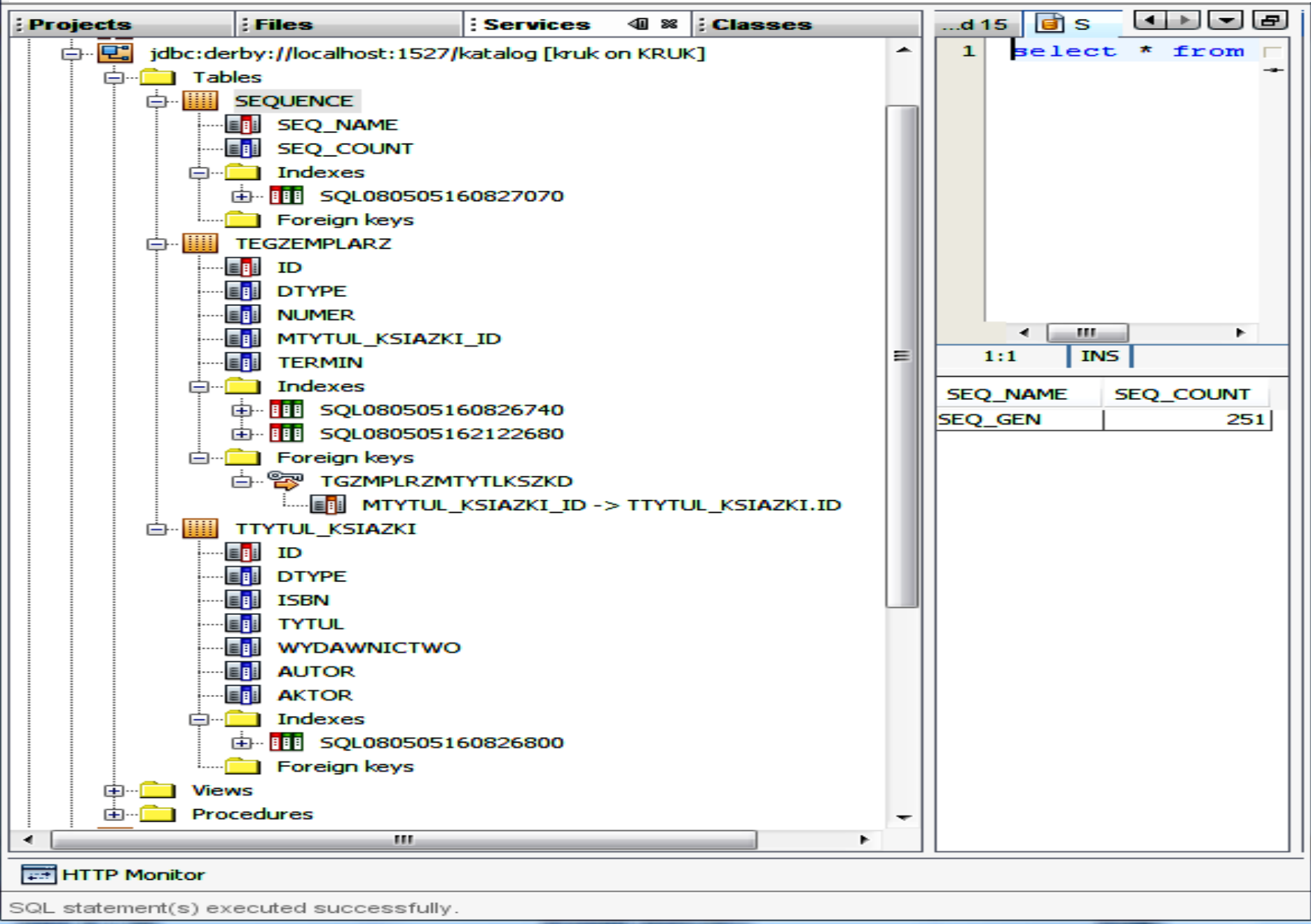
setMTytul_ksiazki

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

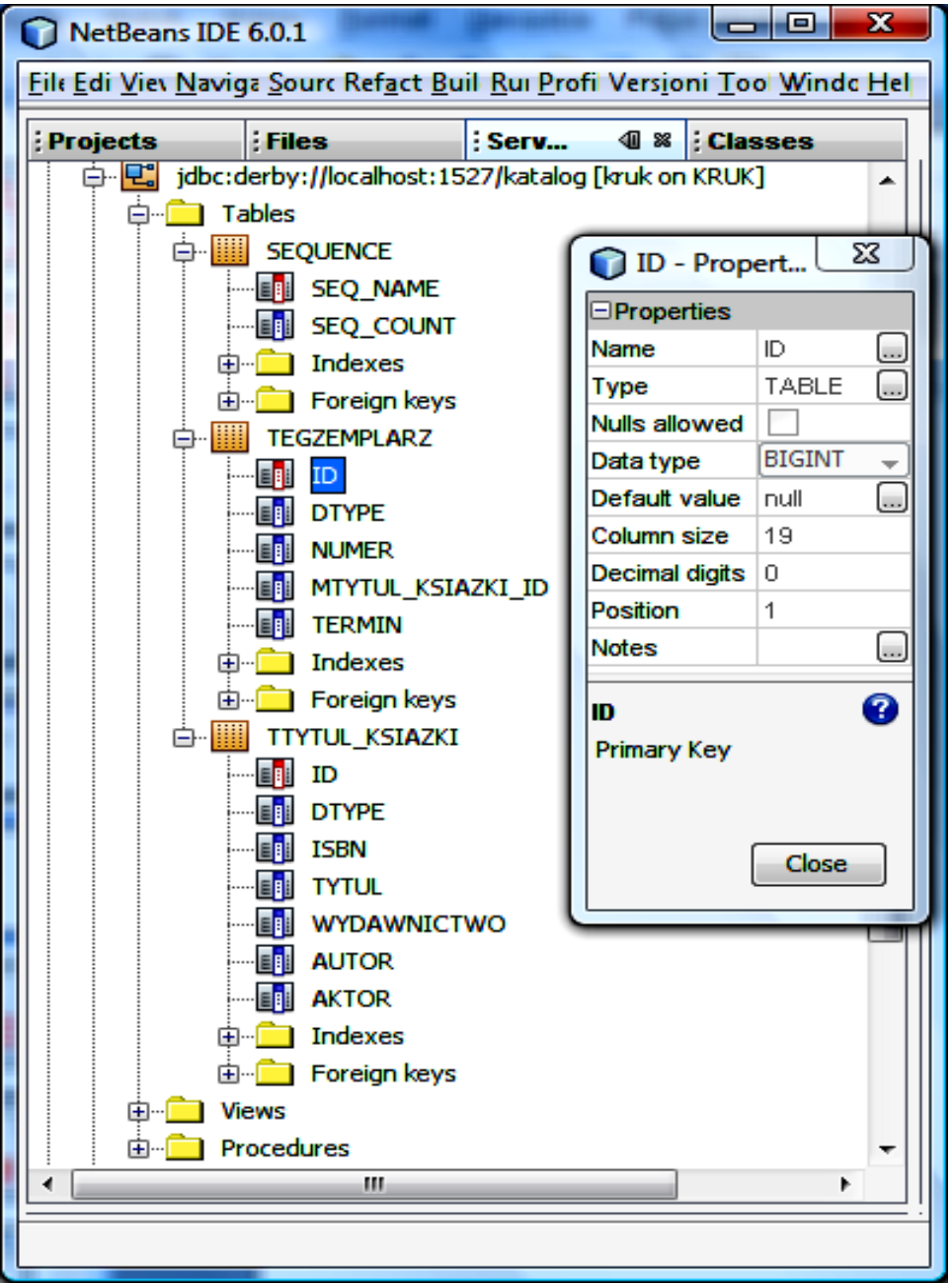


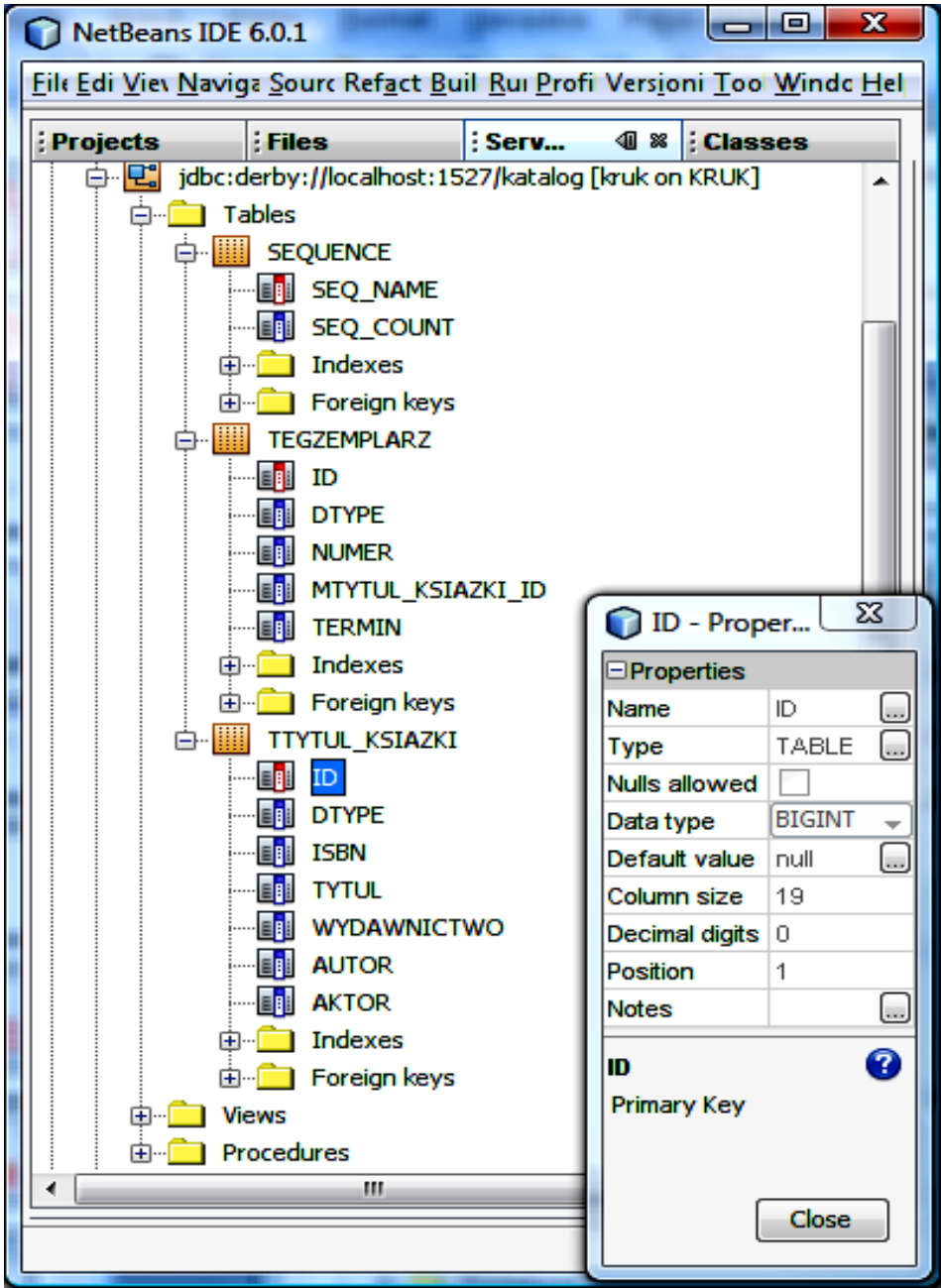

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Projects
Files
Services
Classes
jdbc:derby://localhost:1527/katalog [kruk on KRUK]
Tables
SEQUENCE
SEQ_NAME
SEQ_COUNT
Indexes
SQL080505160827070
Foreign keys
TEGZEMPLARZ
ID
DTYPE
NUMER
MTYTUL_KSIAZKI_ID
TERMIN
Indexes
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
HTTP Monitor

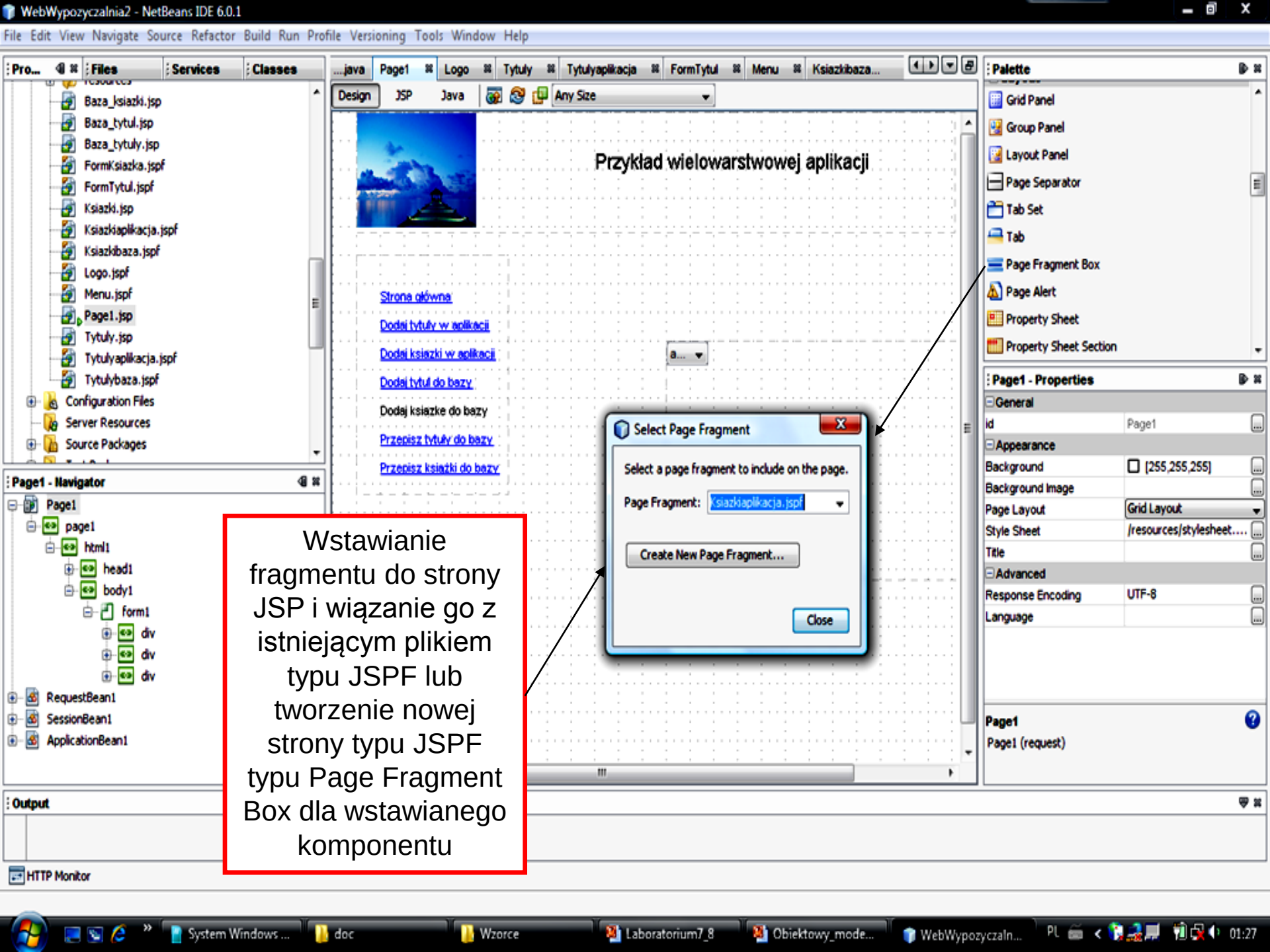
Pomocnicza tabela **Sequence** do generowania kluczy głównych



Właściwości kluczy głównych przy zastosowanej strategii AUTO






WebWypozyczalnia2 - NetBeans IDE 6.0.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Files
Services
Classes
Baza_ksiazki.jsp
Baza_tytul.jsp
Baza_tytuly.jsp
FormKsiazka.jspf
FormTytul.jspf
Ksiazki.jsp
Ksiazkiaplikacja.jspf
Ksiazkibaza.jspf
Logo.jspf
Menu.jspf
Page1.jsp
Tytuly.jsp
Tytulyaplikacja.jspf
Tytulybaza.jspf
Configuration Files
Server Resources
Source Packages
Page1 - Navigator
Page1
page1
html1
head1
body1
form1
div
div
div
RequestBean1
SessionBean1
ApplicationBean1
Page1
Logo
Tytuly
Tytulyaplikacja
FormTytul
Menu
Ksiazkibaza...
Design
JSP
Java
Any Size
Przykład wielowarstwowej aplikacji
Strona główna
Dodaj tytuły w aplikacji
Dodaj ksiązki w aplikacji
Dodaj tytuł do bazy
Dodaj ksiazke do bazy
Przepisz tytuły do bazy
Przepisz ksiązki do bazy
Select Page Fragment
Select a page fragment to include on the page.
Page Fragment:
ksiazkiaplikacja.jspf
Create New Page Fragment...
Close
Palette
Grid Panel
Group Panel
Layout Panel
Page Separator
Tab Set
Tab
Page Fragment Box
Page Alert
Property Sheet
Property Sheet Section
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)
Output
HTTP Monitor
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
System Windows ...
doc
Wzorce
Laboratorium7_8
Obiektowy_mode...
WebWypozyczaln...
PL
01:27