

Tworzenie systemów informatycznych

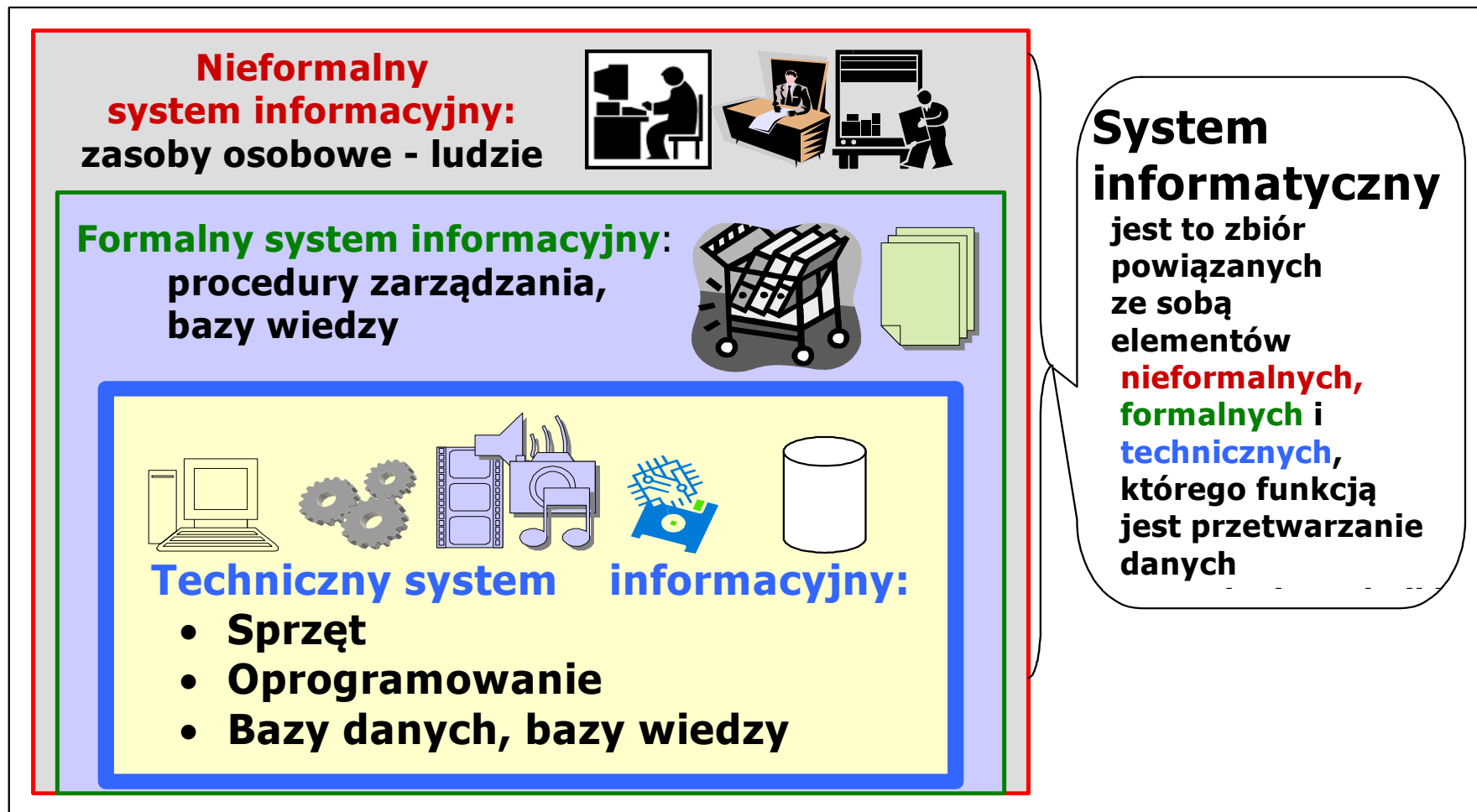
**Inżynieria oprogramowania
Zofia Kruczkiewicz**

Zagadnienia

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

1. Wielowarstwowa architektura systemu informatycznego

Definicja systemu informatycznego



Techniczny system informacyjny

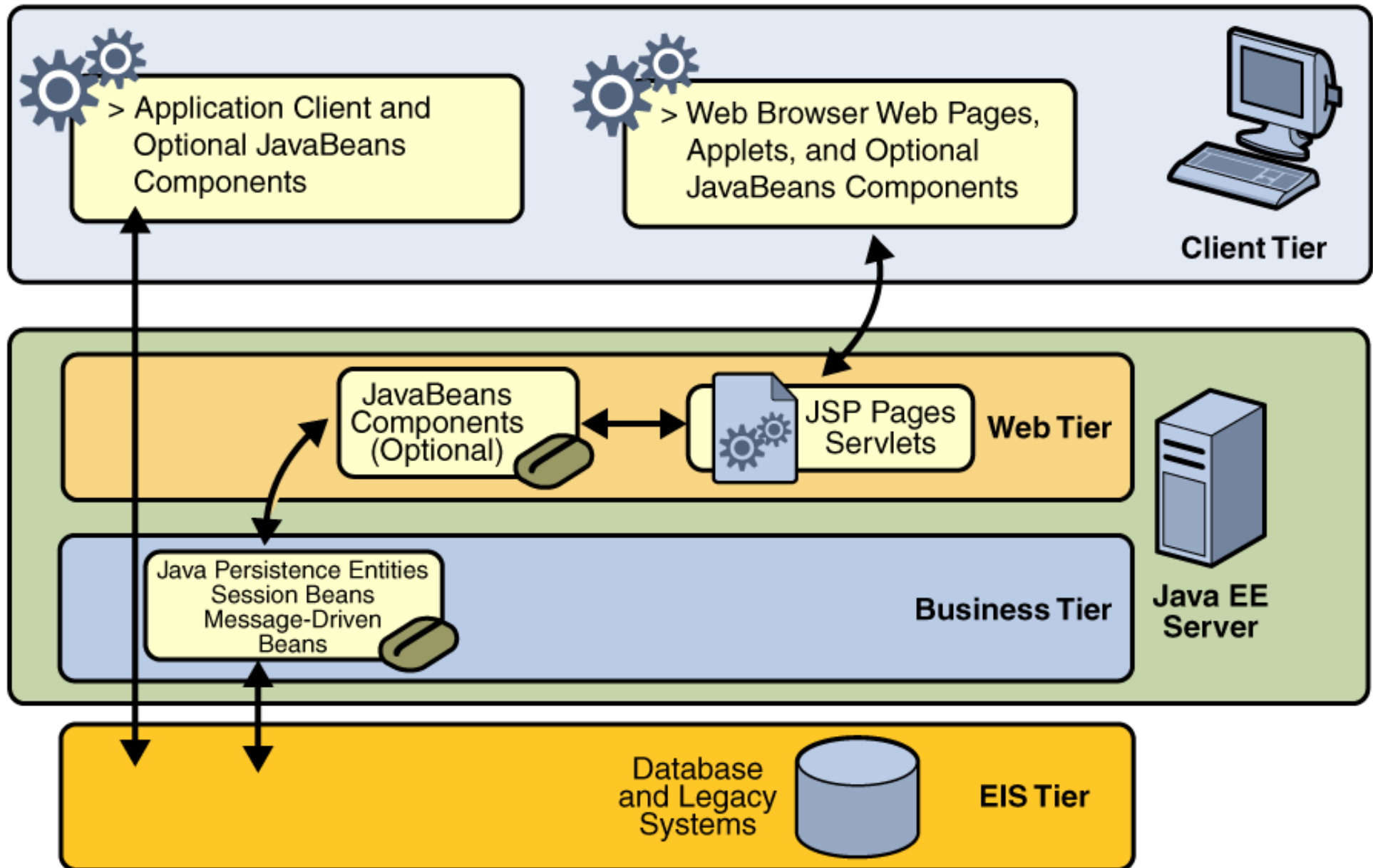
- zorganizowany zespół środków technicznych (komputerów, oprogramowania, urządzeń teletransmisyjnych itp.)
- służący do gromadzenia, przetwarzania i przesyłania informacji

Pięciowarstwowy model logicznego rozdzielania zadań

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



Warstwy aplikacji (Java EE)

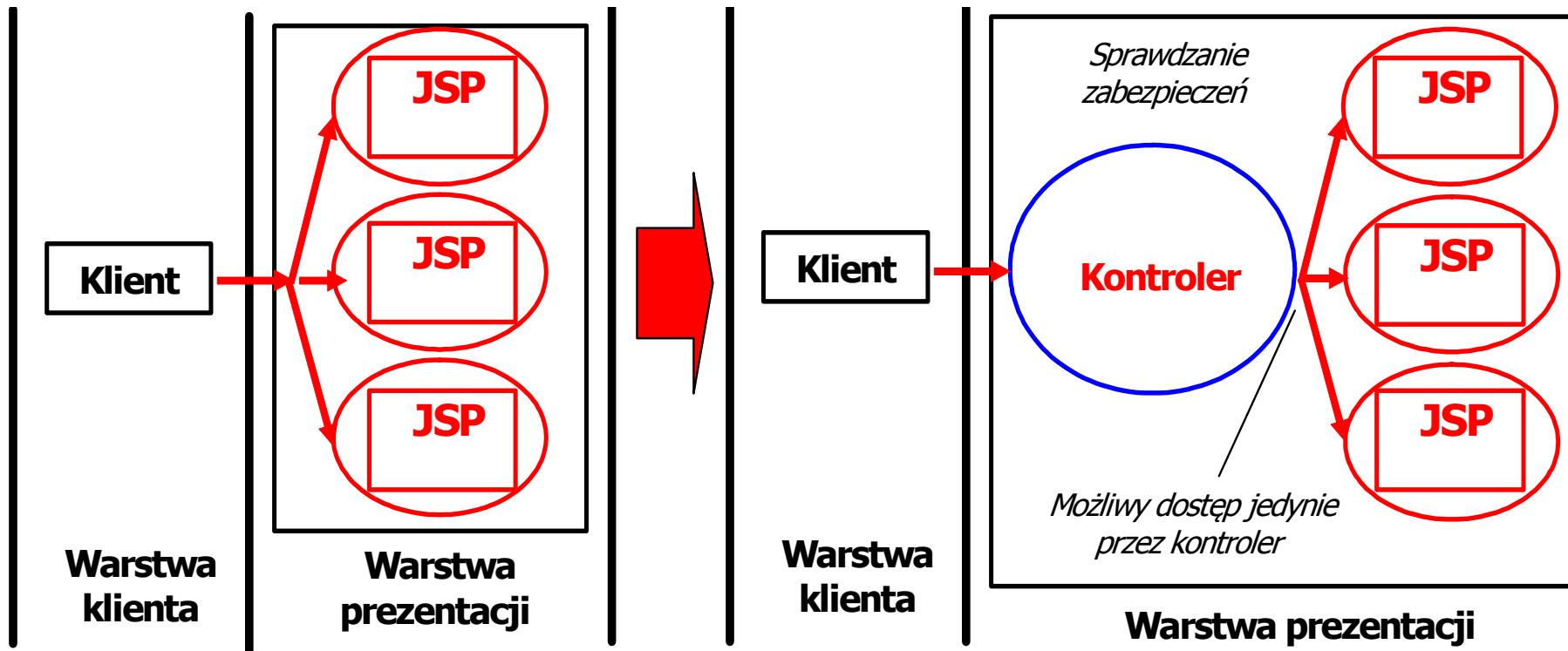


2. Refaktoryzacja architektury wielowarstwowej systemu informatycznego

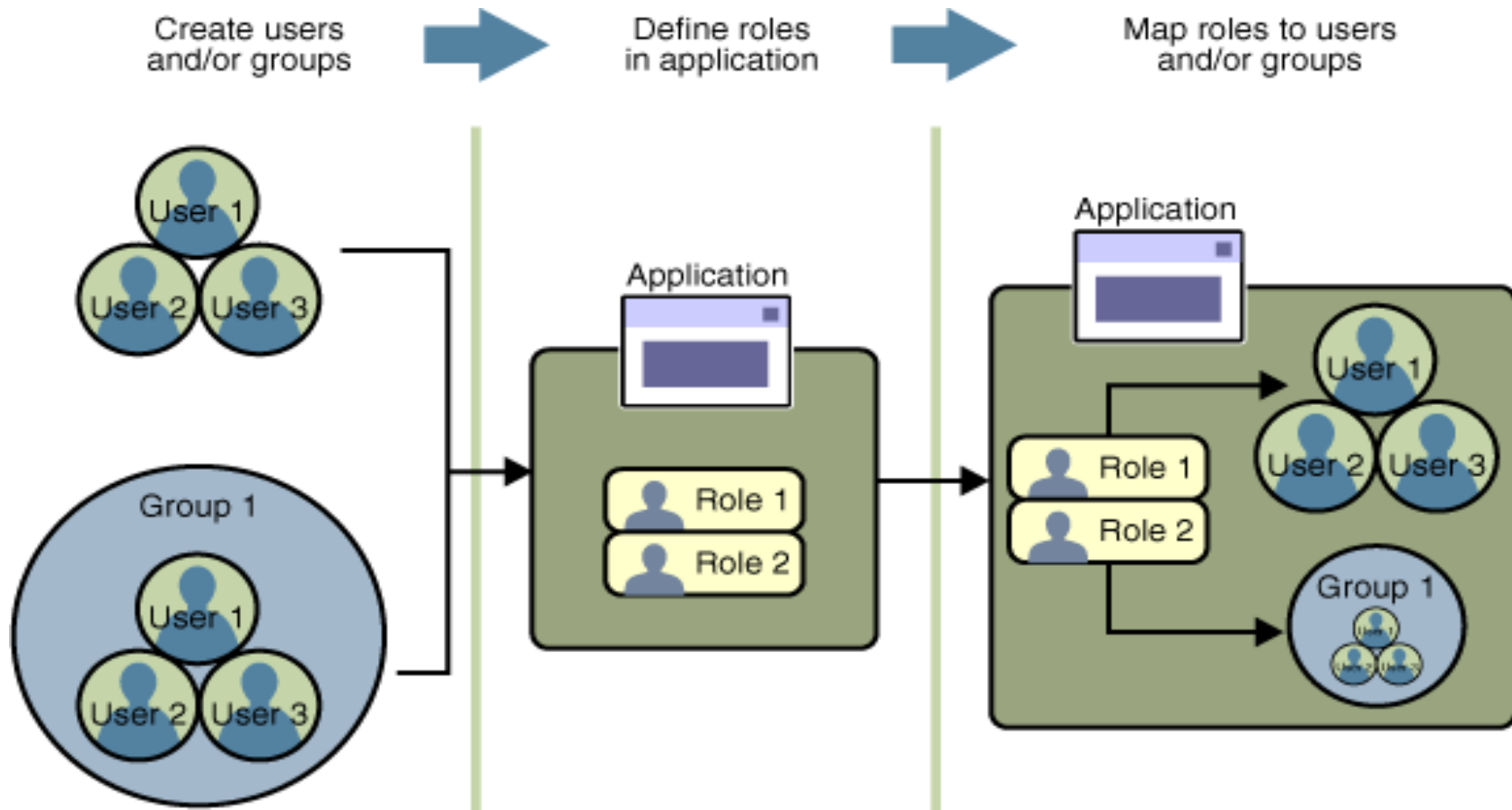
(wg. D.Alur, J.Crupi, D. Malks, Core J2EE. Wzorce projektowe.)

2.1. Refaktoryzacja warstwy prezentacji

Ukrywanie zasobów przed klientem za pomocą konfiguracji kontenera



Bazy użytkowników i grup, Użytkownik, Grupa, Rola



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.

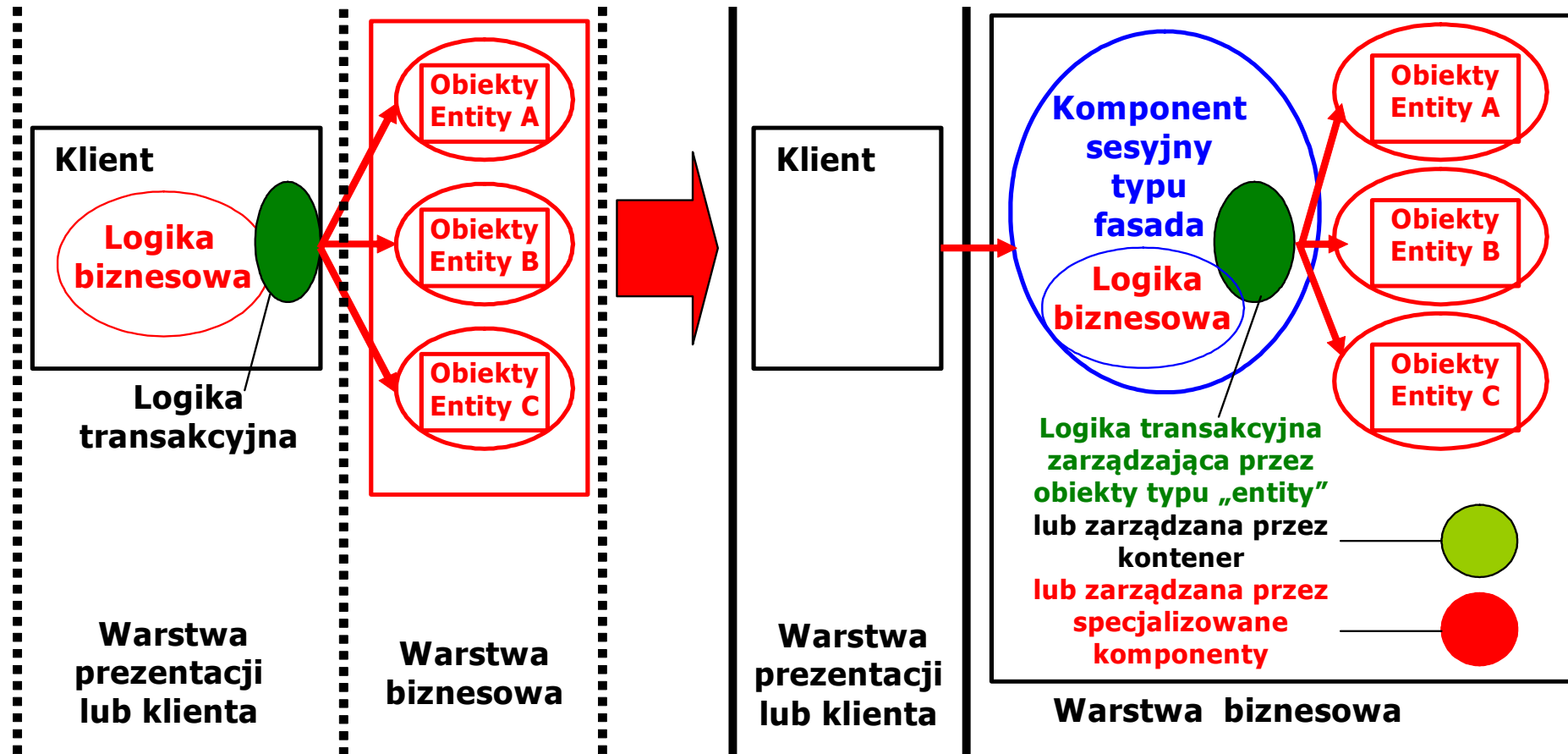
Np.

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

2.2. Refaktoryzacja warstwy biznesowej

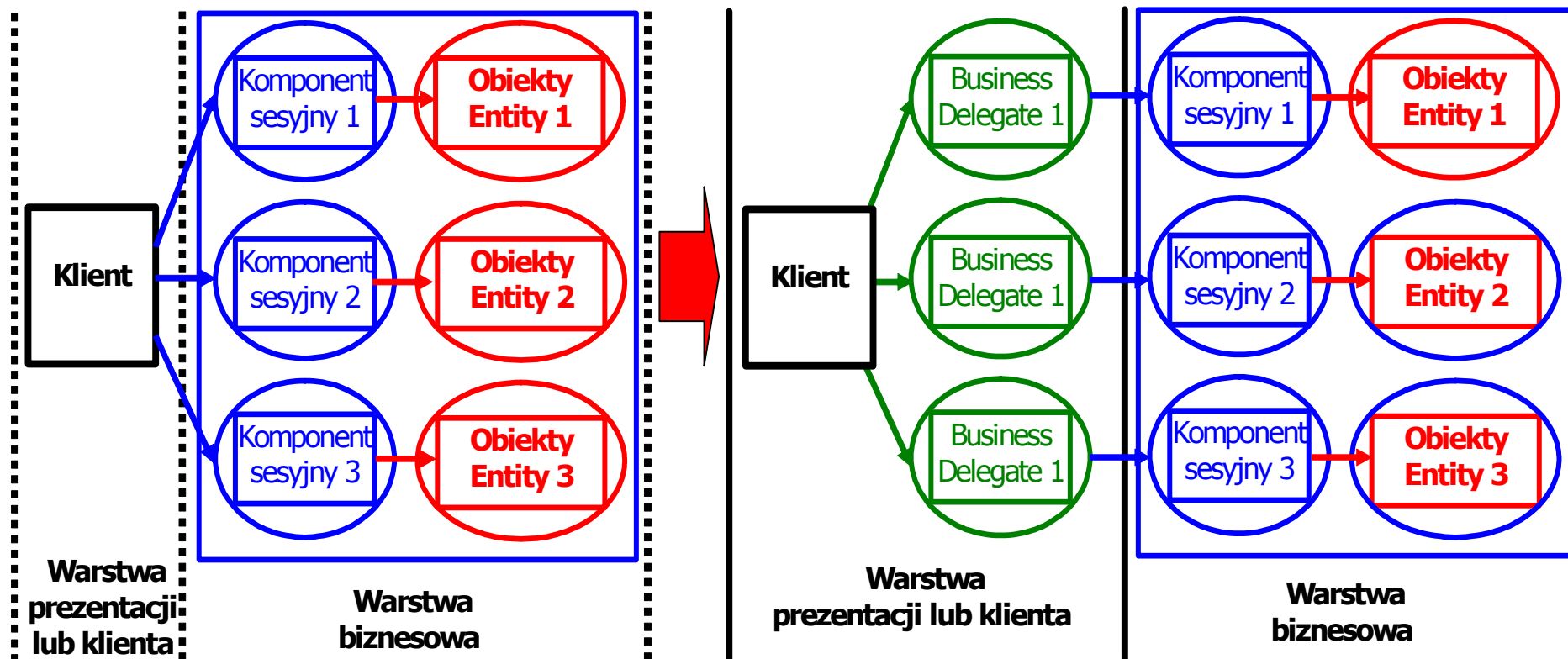
Refaktoryzacja warstwy biznesowej 1

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)



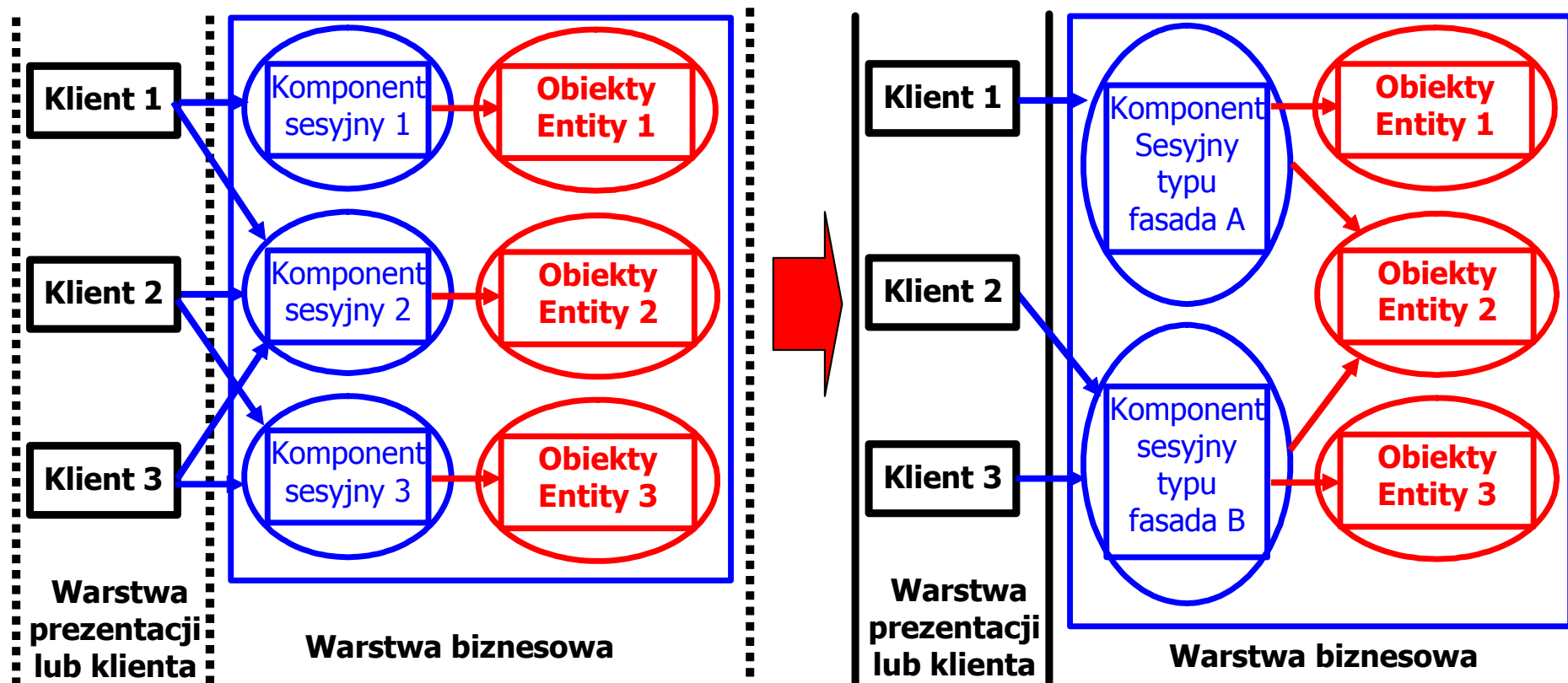
Refaktoryzacja warstwy biznesowej 2

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

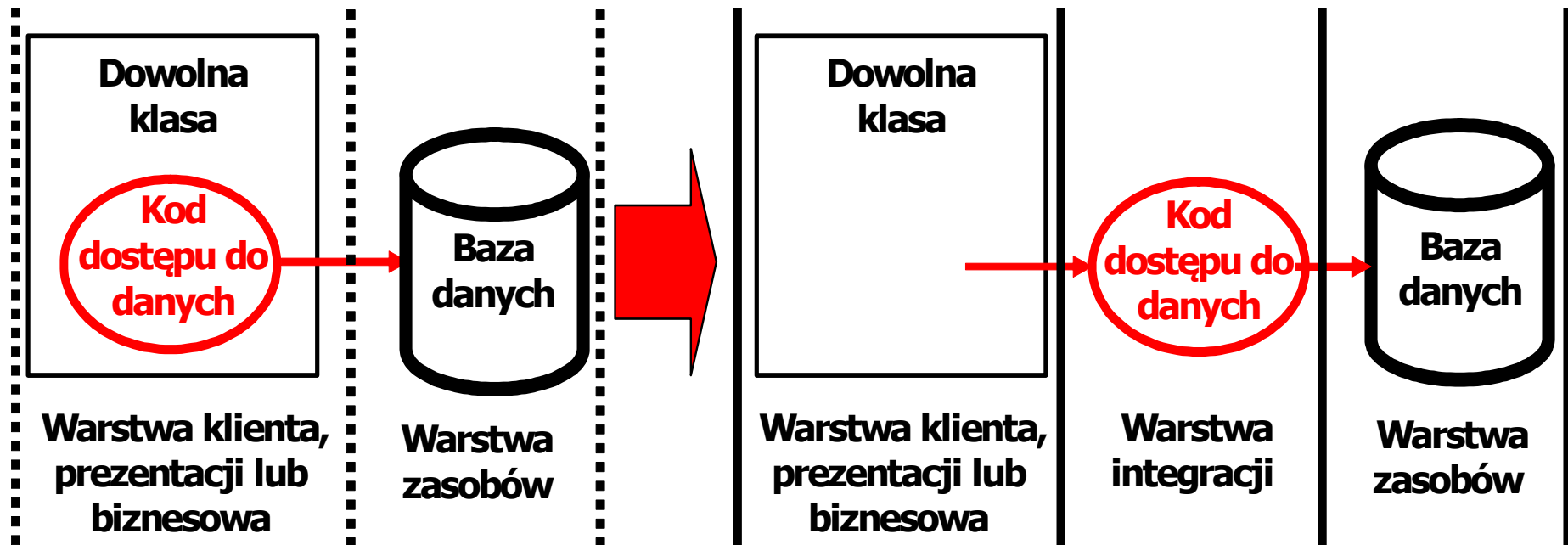
Sesyjne komponenty fasadowe typu „Control” (każdy komponent jako odrębna usługa biznesowa), hermetyzujące **obiekty danych typu „Entity”** z warstwy biznesowej są udostępniane klientom w innych warstwach. Zwykle obiekty sesyjne są jedynie pośrednikami obiektów „Entity”, natomiast nie hermetyzują całych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



2.3. Tworzenie warstwy integracji

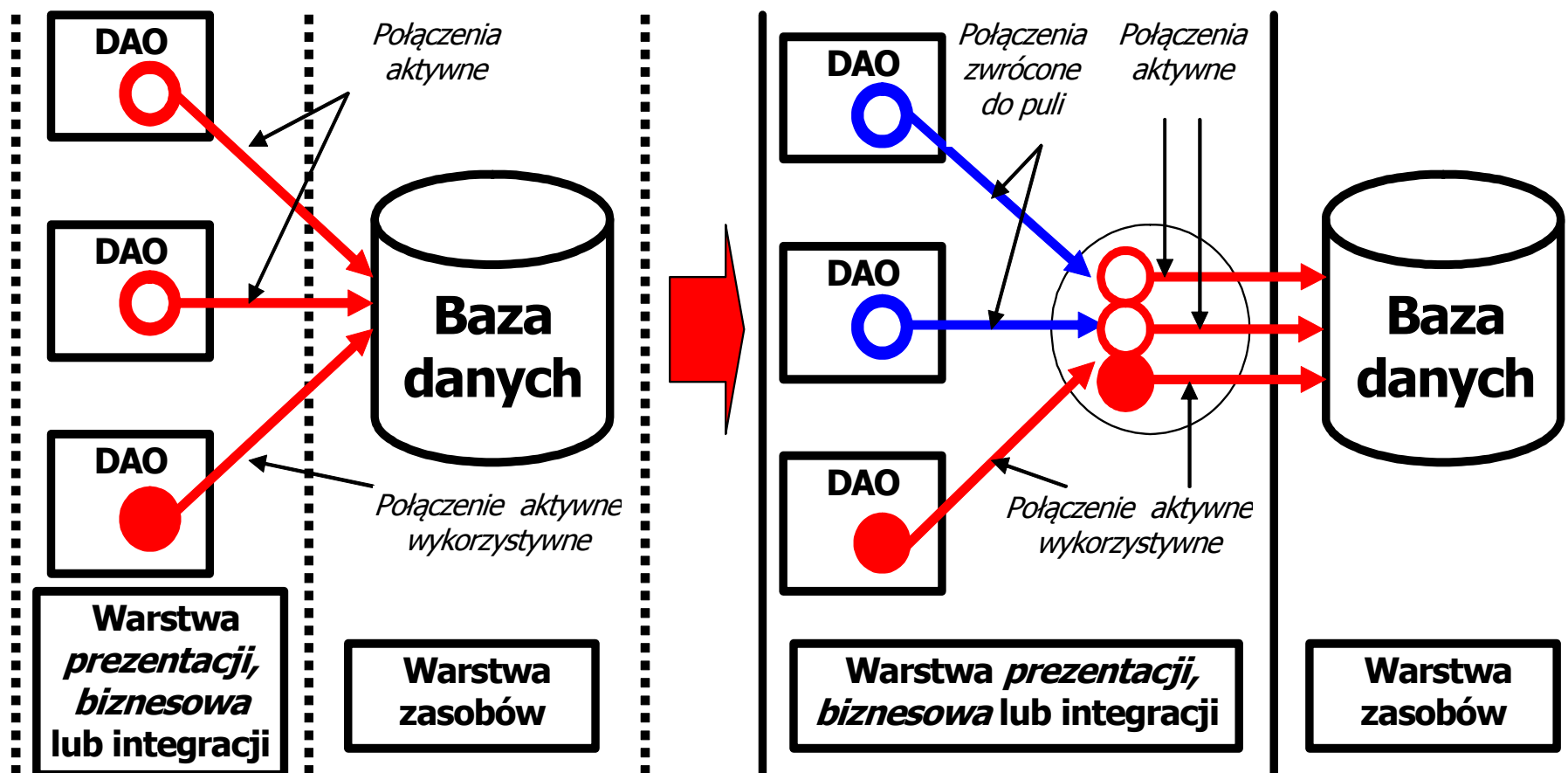
Wydzielanie kodu dostępu do danych

- 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

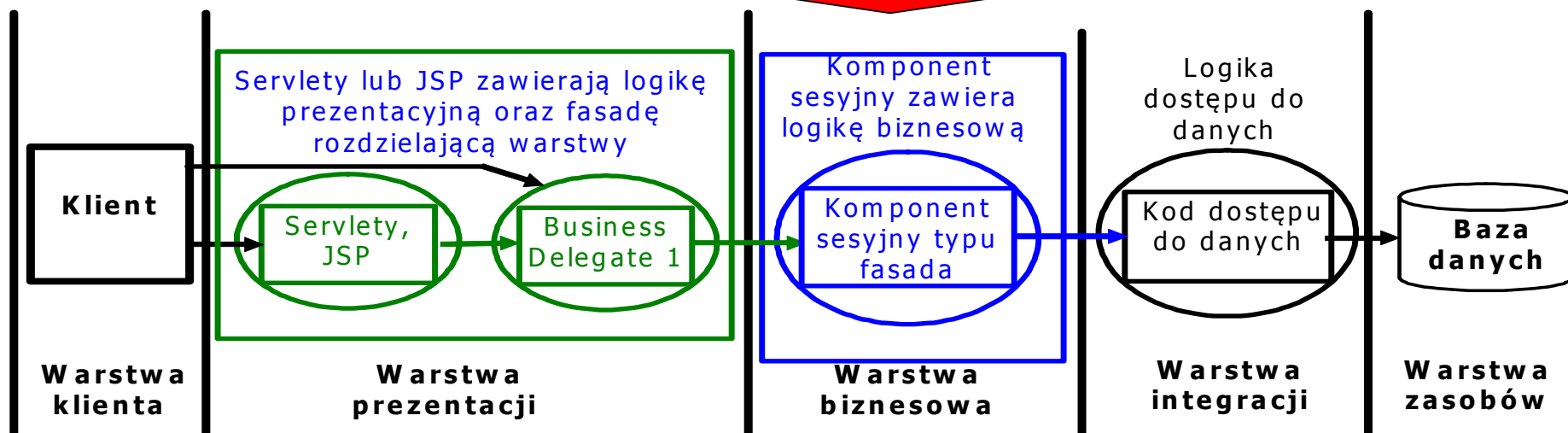


3. Poprawa struktury tworzonego systemu informatycznego – przykłady pięciowarstwowych systemów informatycznych

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

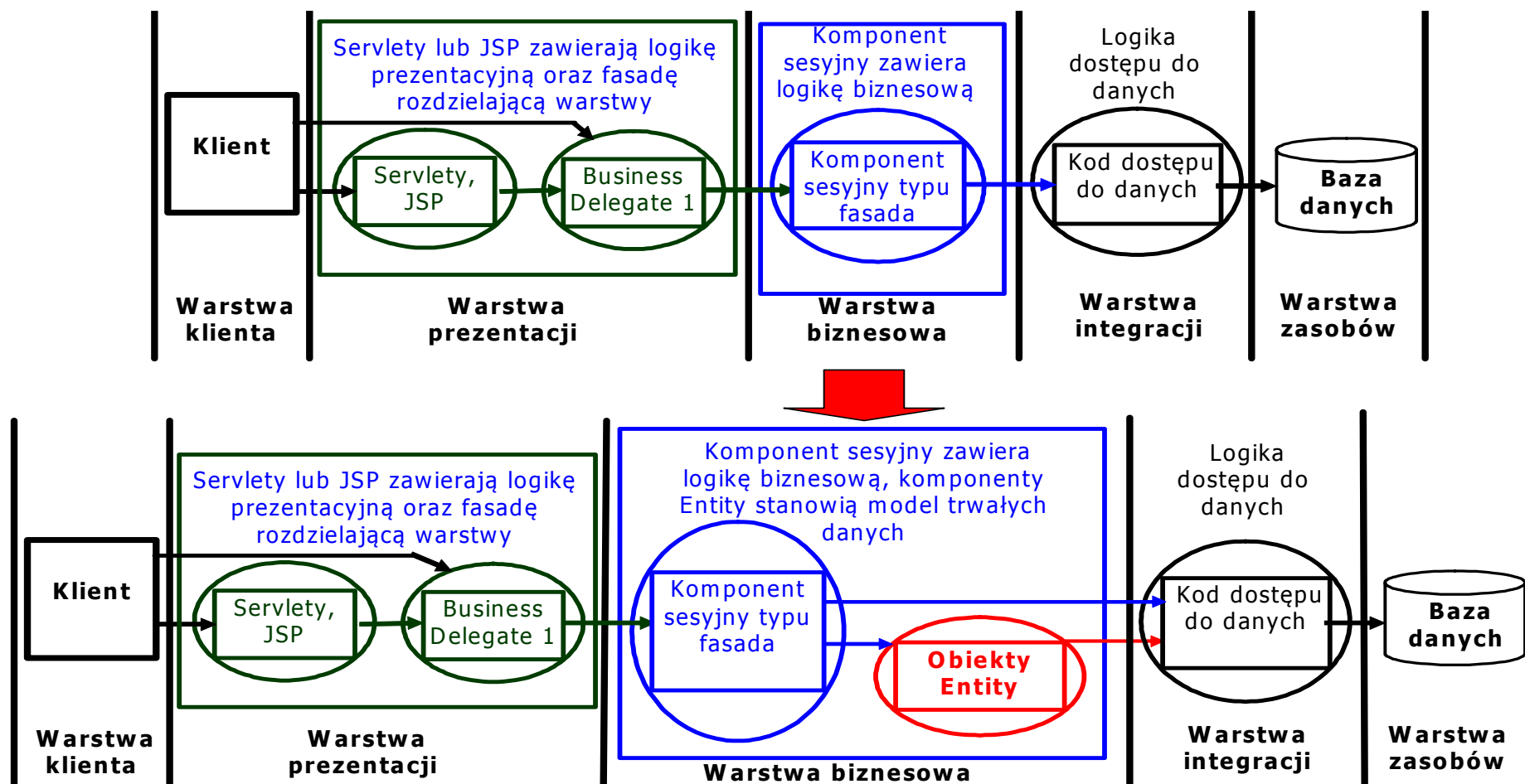
Refaktoryzacja architektury wielowarstwowej 1

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.



4. 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)

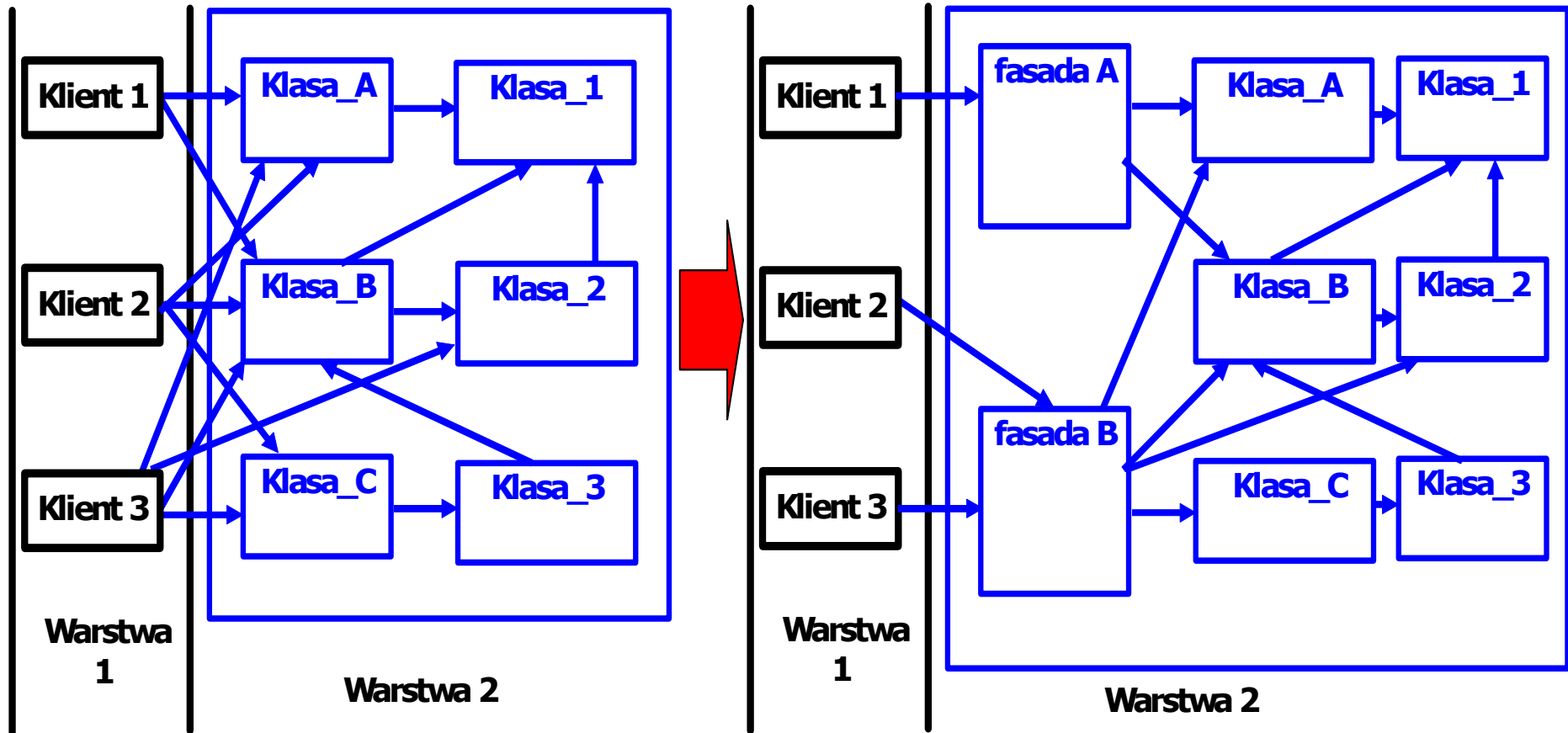
- Wzorzec to pomysł, który okazał się użyteczny w jednym rzeczywistym kontekście i prawdopodobnie będzie użyteczny w innym. **(Martin Fowler)**

Identyfikacja wzorców projektowych

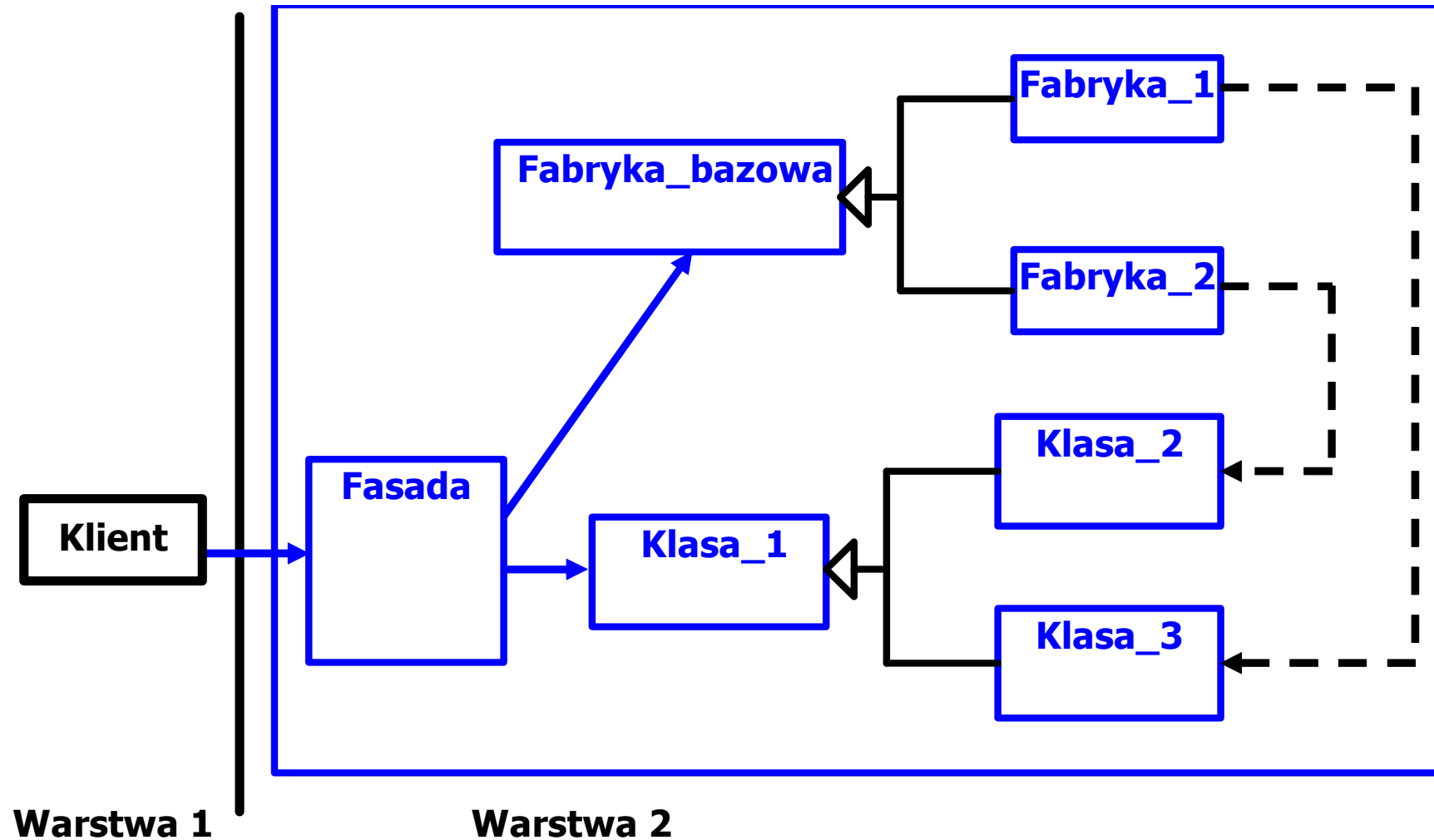
- Dobrze zbudowany system obiektowy jest pełen wzorców obiektowych
- Wzorzec to zwyczajowo przyjęte rozwiązanie typowego problemu w danym kontekście
- Strukturę wzorca przedstawia się w postaci diagramu klas
- Zachowanie się wzorca przedstawia się za pomocą diagramu sekwencji
- Wzorce projektowe: Wzorzec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

- Każdy wzorzec składa się z trzech części, które wyrażają związek między konkretnym kontekstem, problemem i rozwiązaniem (**Christopher Aleksander**)
- Każdy wzorzec to trzyczęściowa reguła, która wyraża związek między konkretnym kontekstem, rozkładem sił powtarzającym się w tym kontekście i konfiguracją oprogramowania pozwalającą na wzajemne zrównoważenie się tych sił w celu rozwiązania zadania. (**Richard Gabriel**)
- Wzorzec to pomysł, który okazał się użyteczny w jednym rzeczywistym kontekście i prawdopodobnie będzie użyteczny w innym. (**Martin Fowler**)

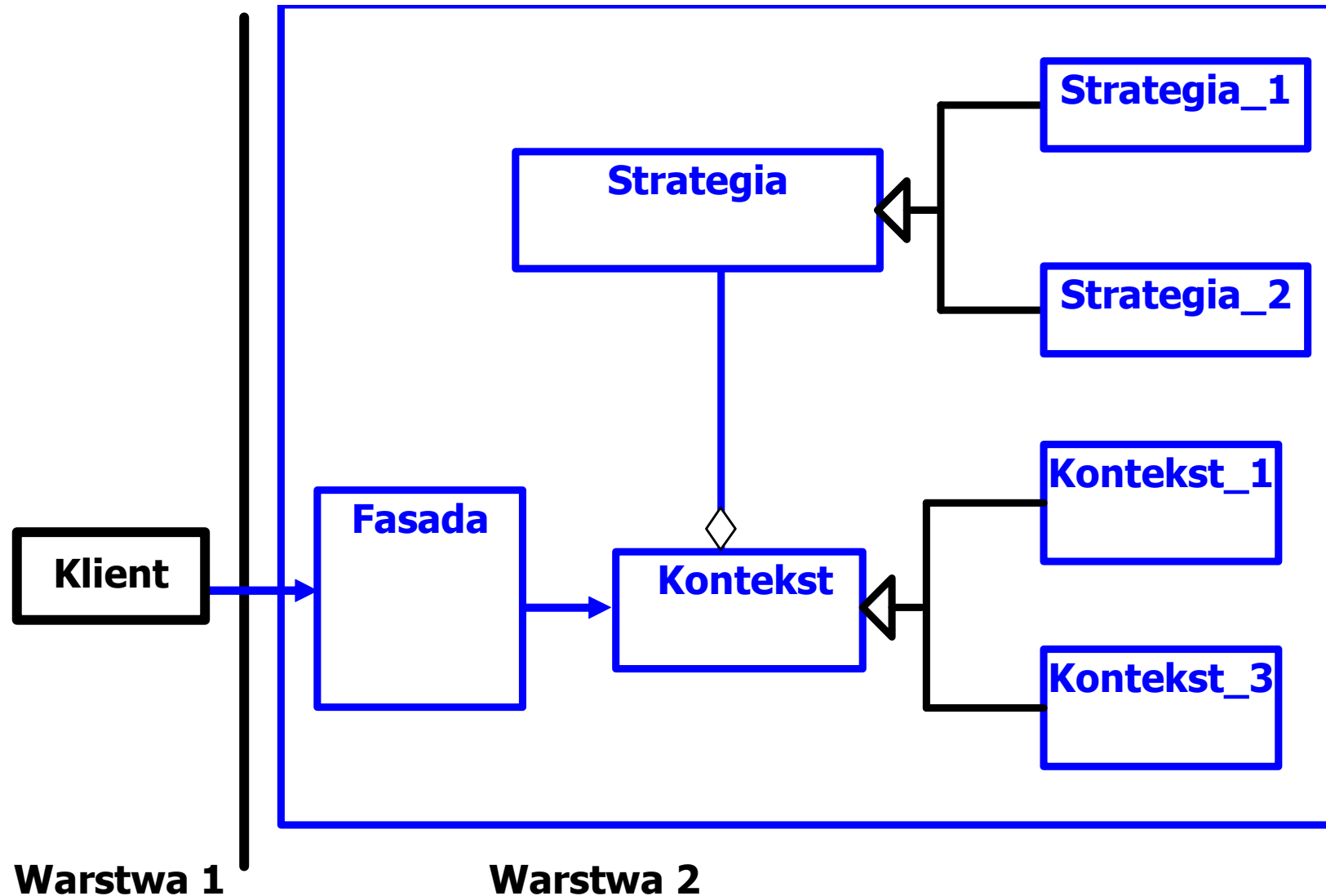
4.1. Wzorzec fasady (wzorzec strukturalny)



4.2. Wzorzec fabryki obiektów (wzorzec kreacyjny)

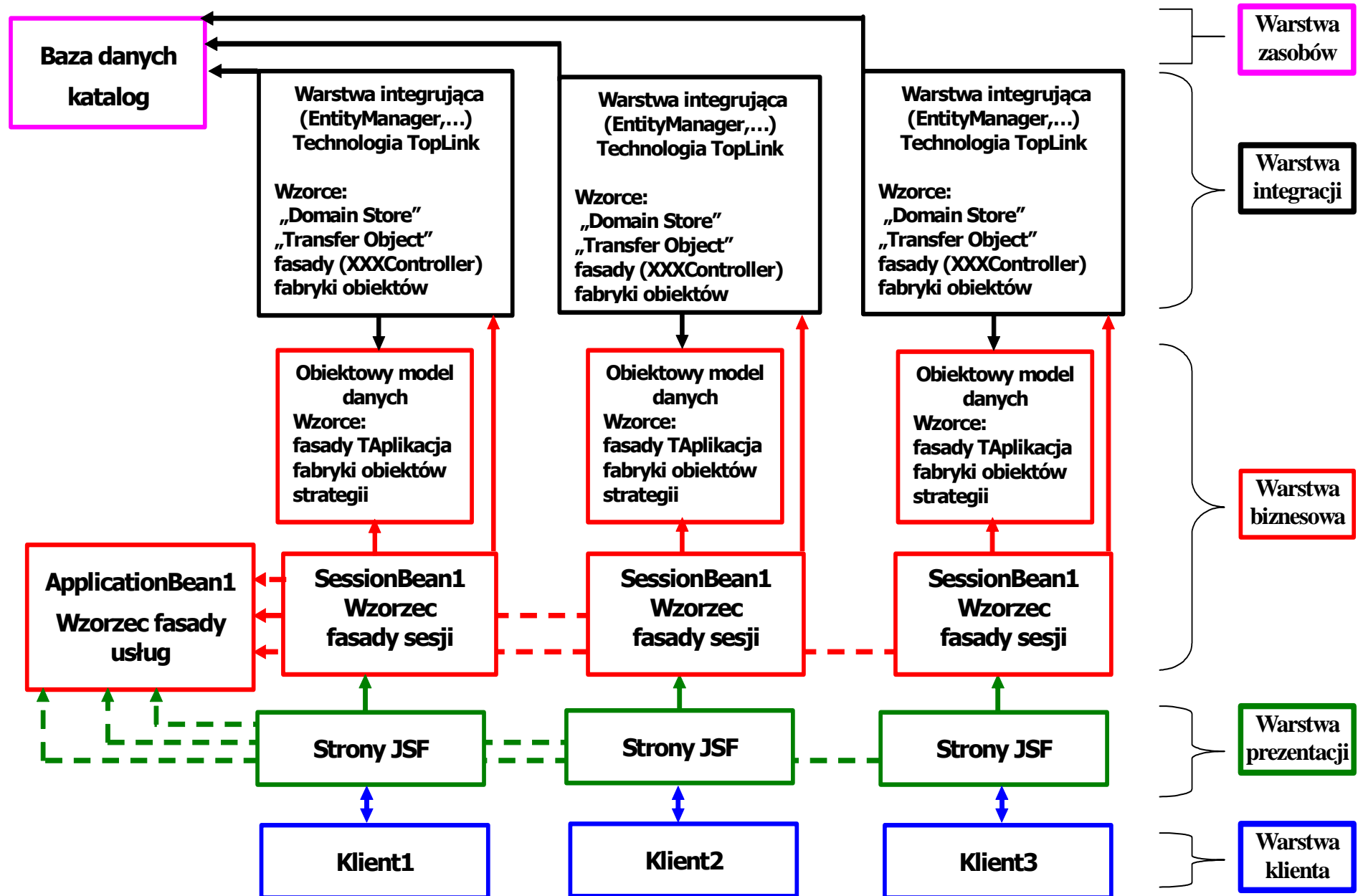


4.3. Wzorzec strategii (wzorzec czynnościowy)

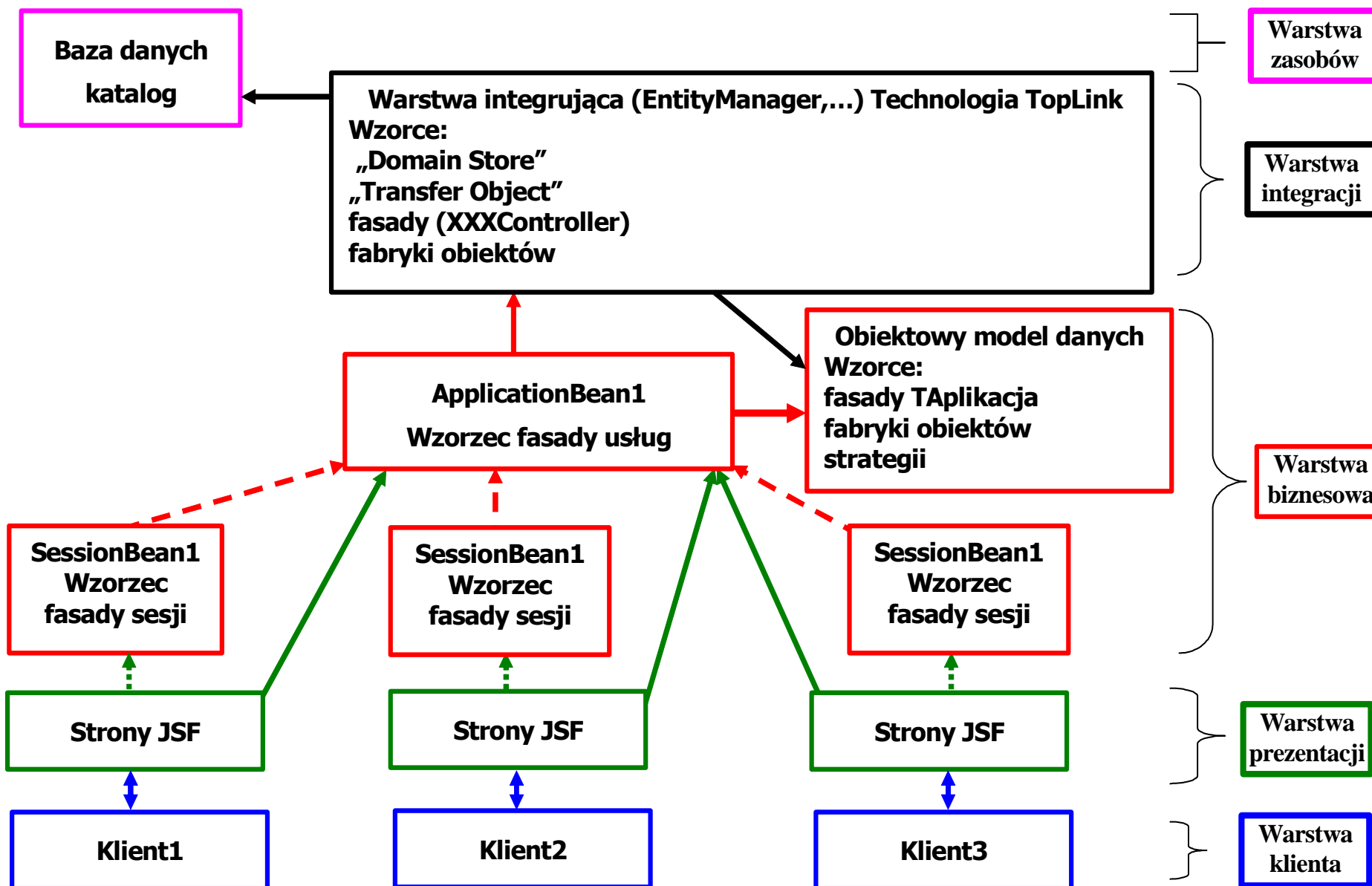


5. Przykłady architektury wielowarstwowej w środowisku Visual Web Java Server Faces

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



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



6. Przykład warstwy biznesowej stosującej wzorce obiektowe

System informacyjny wypożyczalni książek

- 1. Opis biznesowy** systemu
- Sformułowanie **wymagań funkcjonalnych i niefunkcjonalnych** systemu
- 3. Model analizy całego systemu** 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

Opis biznesowy aplikacji w języku klienta

Opis biznesowy aplikacji typu 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.

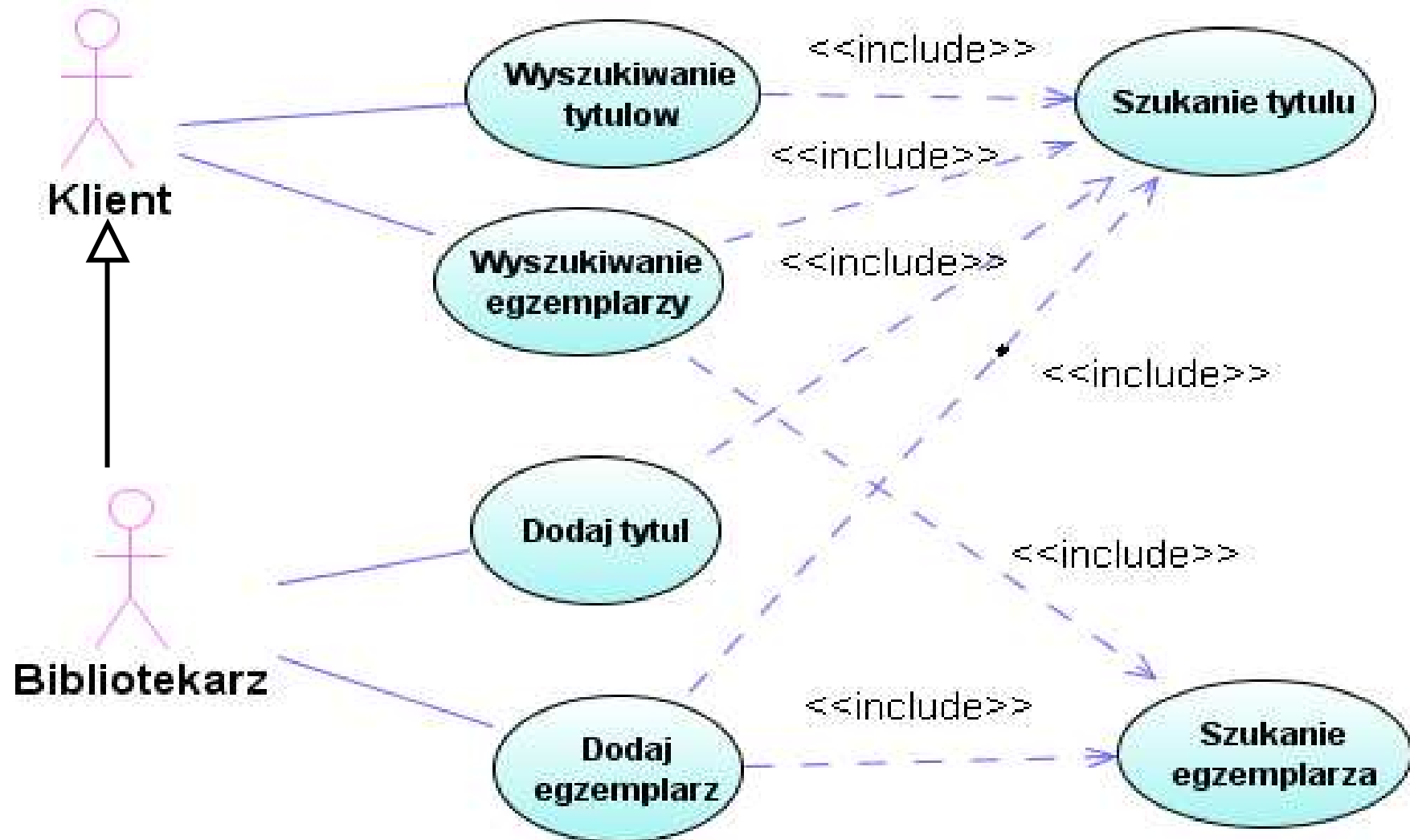
Lista wymagań funkcjonalnych

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 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

Wypożyczalnia



AKTOR	OPIS	PRZYPADKI UŻYCIA
Bibliotekarz	<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>	<ul style="list-style-type: none"> • Dodaj tytuł • Dodaj egzemplarz • Wyszukiwanie tytułów • Wyszukiwanie egzemplarzy
Klient	<i>Klient może jedynie przeszukiwać zasoby katalog tytułów i egzemplarzy książek</i>	<ul style="list-style-type: none"> • Wyszukiwanie tytułów • Wyszukiwanie egzemplarzy

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 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 egzemplarza
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)

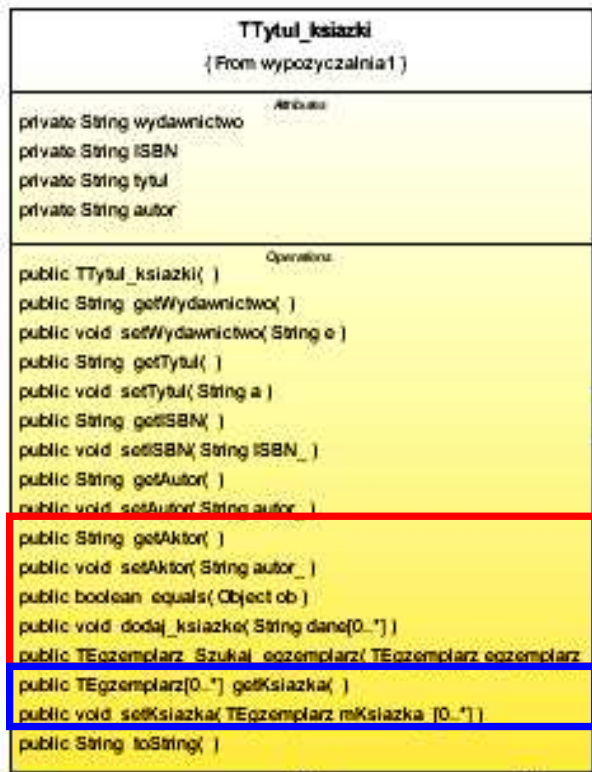
WK (warunki końcowe): wstawienie egzemplarza o tytule zgodnym z podanymi atrybutami obowiązkowymi ISBN lub ISBN i aktor w przypadku nagrania dźwiękowego oraz podanym numerze i ewentualnie atrybucie do określania terminu zwrotu, jeśli należy wstawić egzemplarz z wyznaczonym terminem zwrotu lub podanie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

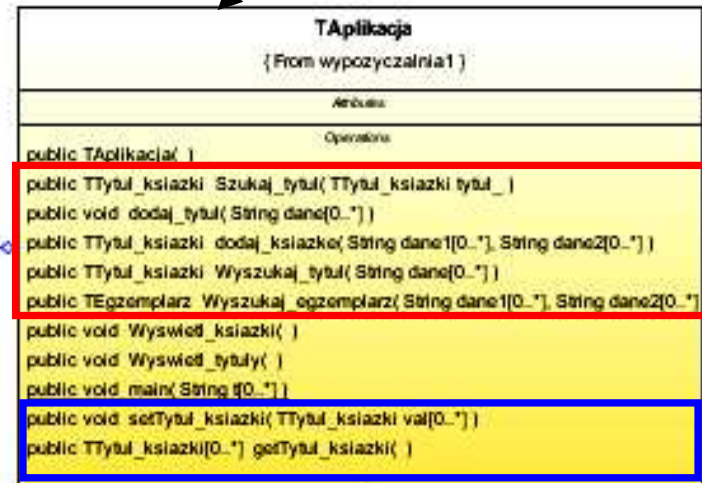
Analiza wspólności i zmienności

- Wykryto **dwie główne klasy typu „Entity”** ze względu na odpowiedzialność: **TTytul_książki** (zawiera atrybuty tytułu, posiada książki – wstawia i wyszukuje je), oraz **TEgzemplarz** (posiada numer). Pojęcia **książki i egzemplarza** są równoważne.
- Wykryto dziedziczenie w właściwościach tytułów, które mogą wystąpić jako zwykłe książki lub jako nagrania dźwiękowe (**klasa TTytul_książki_na_kasecie** typu „Entity”, która dziedziczy od klasy **TTytul_książki**). Określono strategię przechowywania danych o tytule na wielu egzemplarzach książek lub kaset. Wyróżniono egzemplarze zwykłe typu **TEgzemplarz**, rozróżniane w ramach danego tytułu książki zwykłej lub nagranej w postaci dźwiękowej numerem oraz egzemplarze **TEgzemplarz_termin** z dodatkowo oznaczonym terminem oddania.
- 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.
- Wykryto związki silnej agregacji między tytułem i egzemplarzem – egzemplarz nie może istnieć bez tytułu. Wybrano **worzec strategii** do implementacji obiektów typu **TEgzemplarz**
- Zastosowano klasę **TAplikacja** typu „Control” jako **worzec fasady** do oddzielenia obiektów typu „Entity” od pozostałej części systemu oraz **klasę** typu „Control” jako **worzec fabryki obiektów** (**TFabryka**) do tworzenia różnych typów tytułów oraz egzemplarzy.



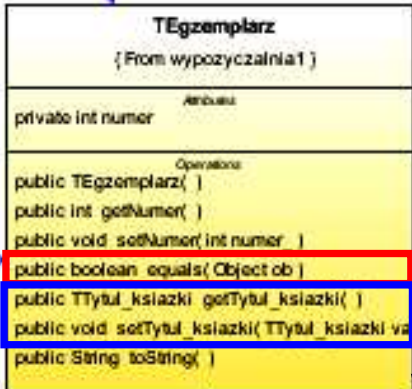
- Implementacja powiązań
- Metody przypadków użycia
- Decyzja projektowa

Wzorzec fasady



Wzorzec strategii

Wzorzec fabryki obiektów



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

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

Projekt przypadku użycia

„ **Szukanie tytułu** ”

za pomocą diagramu sekwencji i diagramu klas.

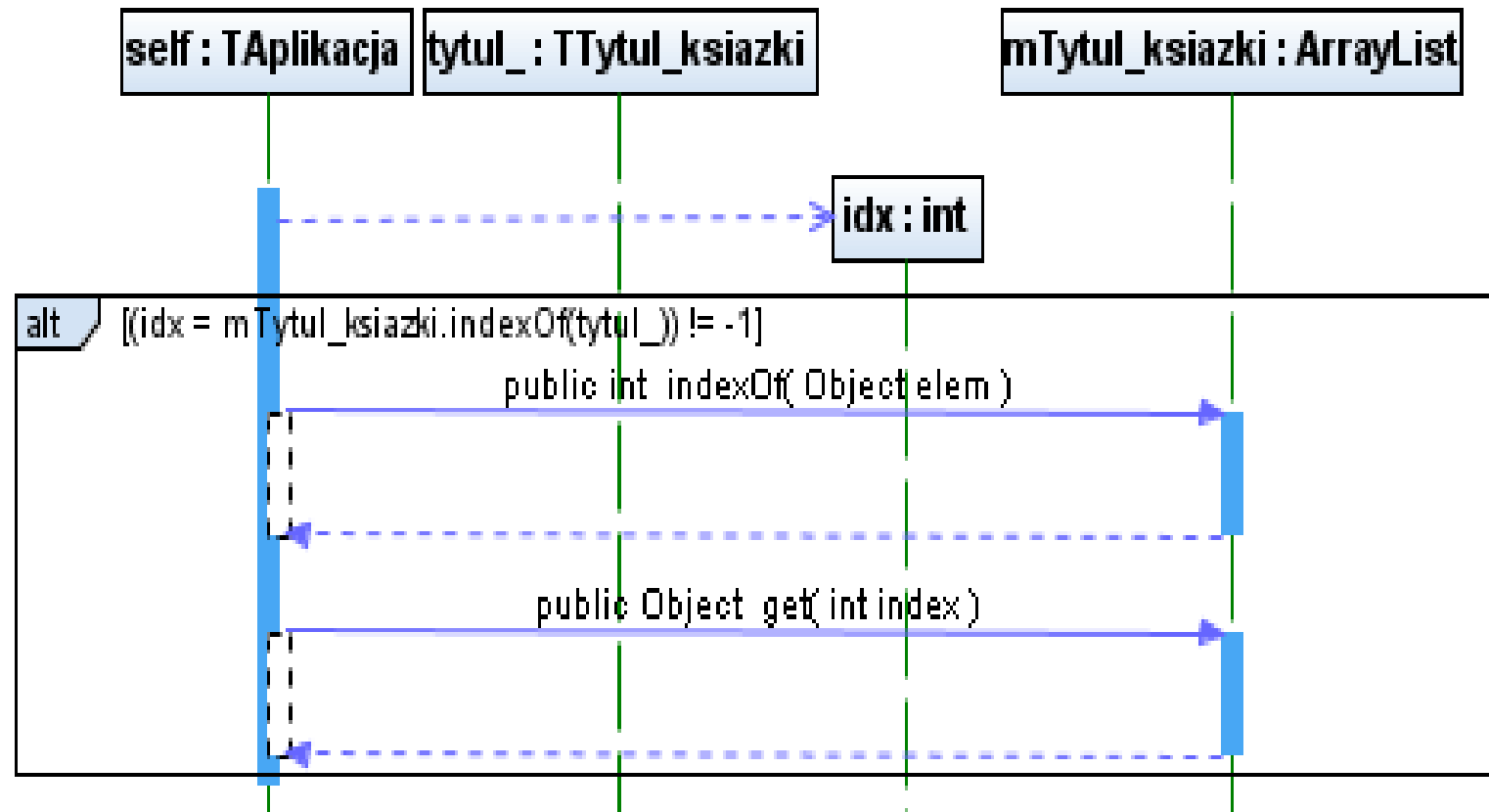
Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia

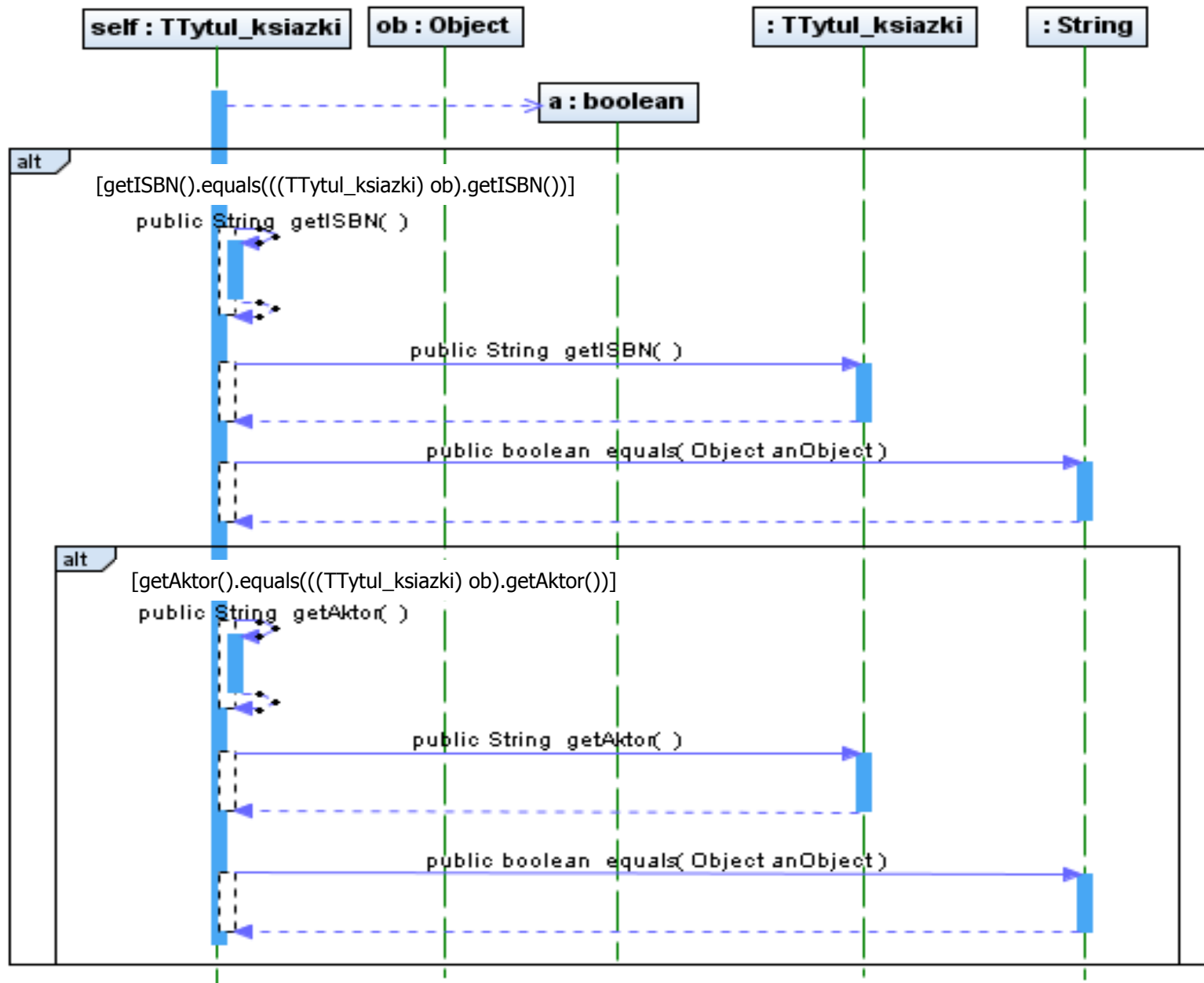
na podstawie diagramów sekwencji

(1) Szukanie tytułu

(TTytul_książki TAplikacja::Szukaj_tytul(TTytul_książki tytul_))



boolean TTytul_książki::equals(Object ob)



Projekt przypadku użycia

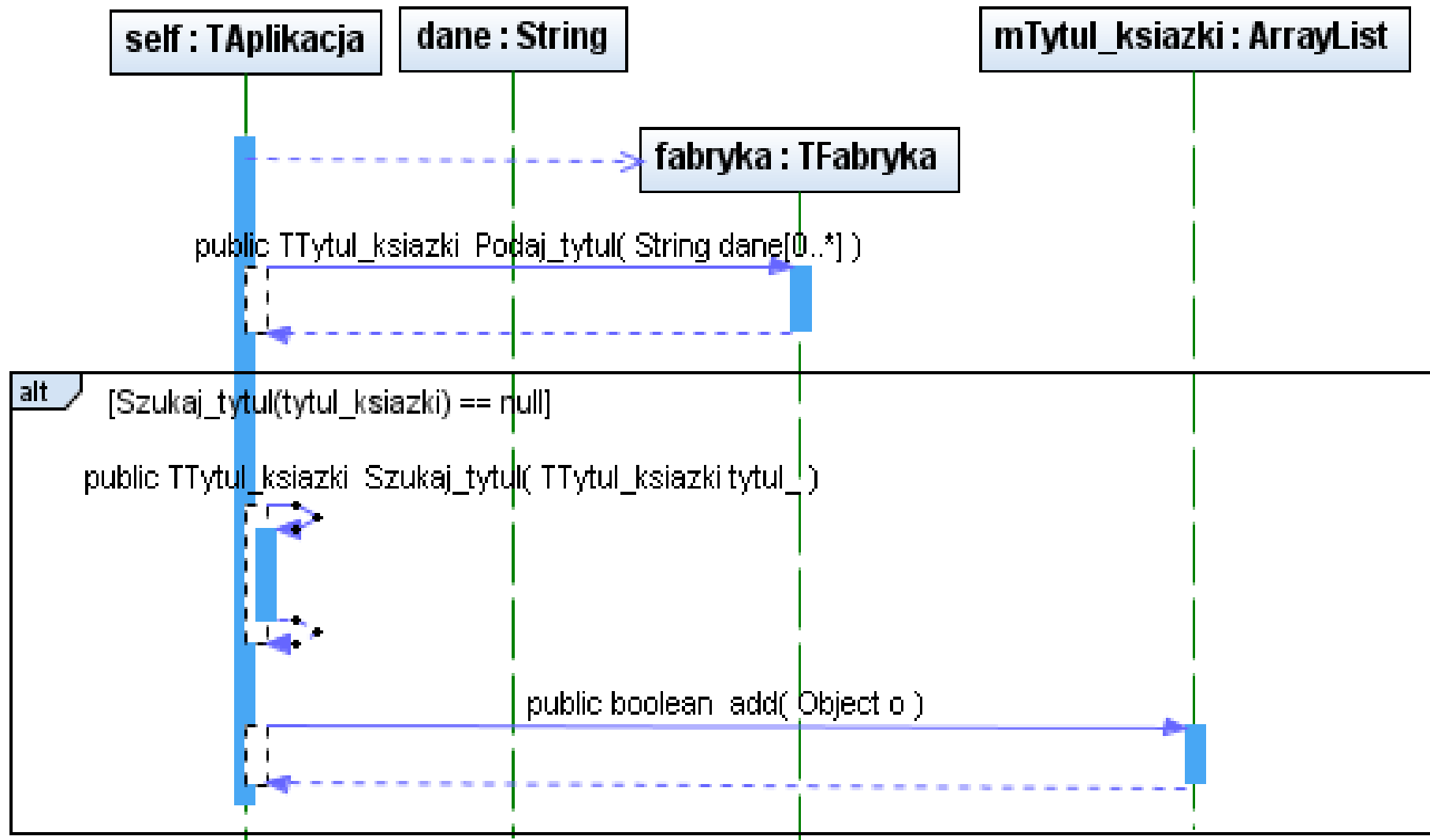
„ **Dodaj tytuł** ”

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

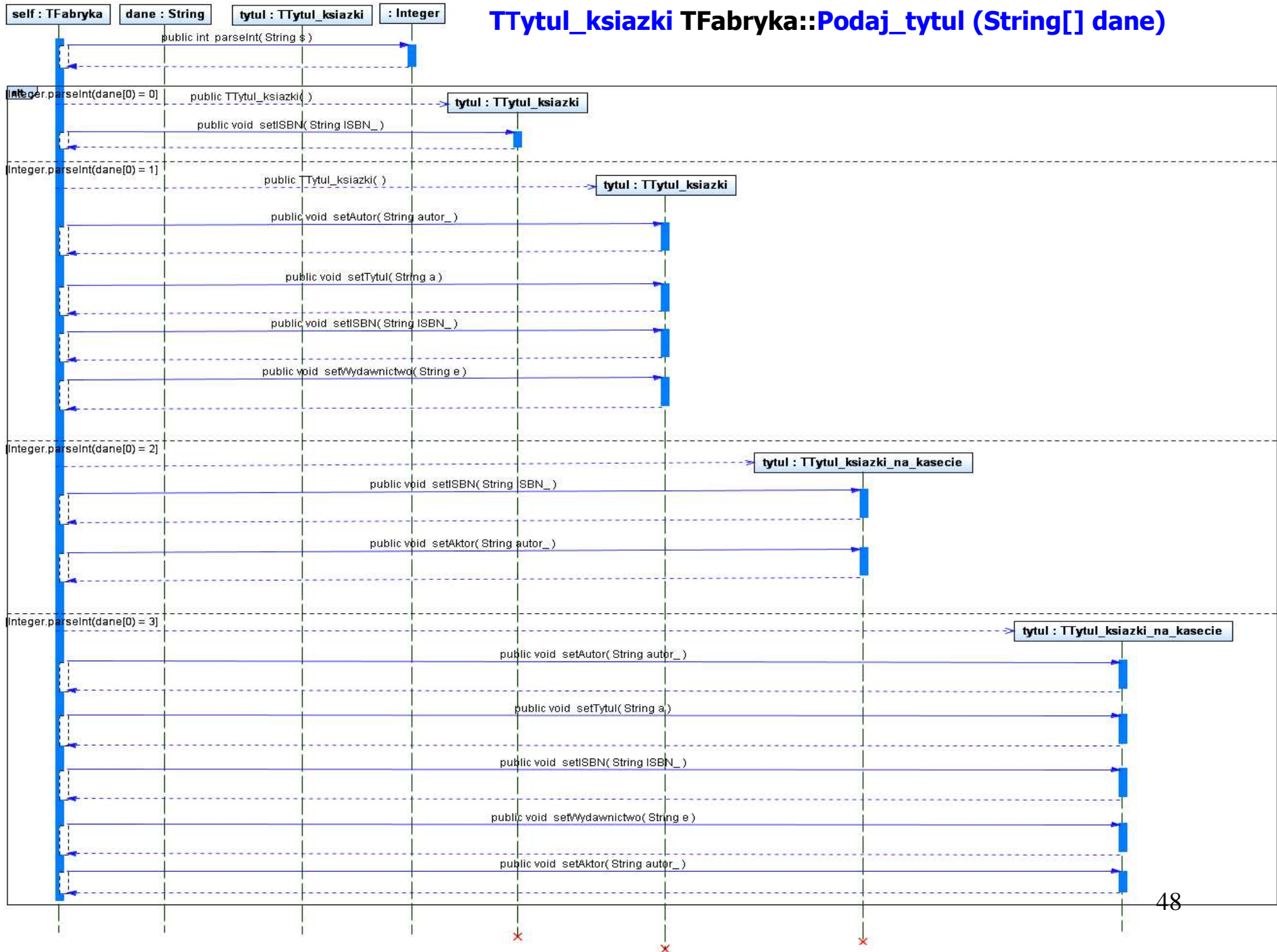
Definiowanie kodu metod realizujących przypadek użycia na podstawie diagramów sekwencji

(2) Dodaj tytuł

void TAplikacja::dodaj_tytul(String dane[])



TTytul_ksiazki TFabryka::Podaj_tytul (String[] dane)




```
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]
```

// 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_książki** oraz **TTytul_książki_na_kasecie**) i **getTytul_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 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);
```

```
}
```

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

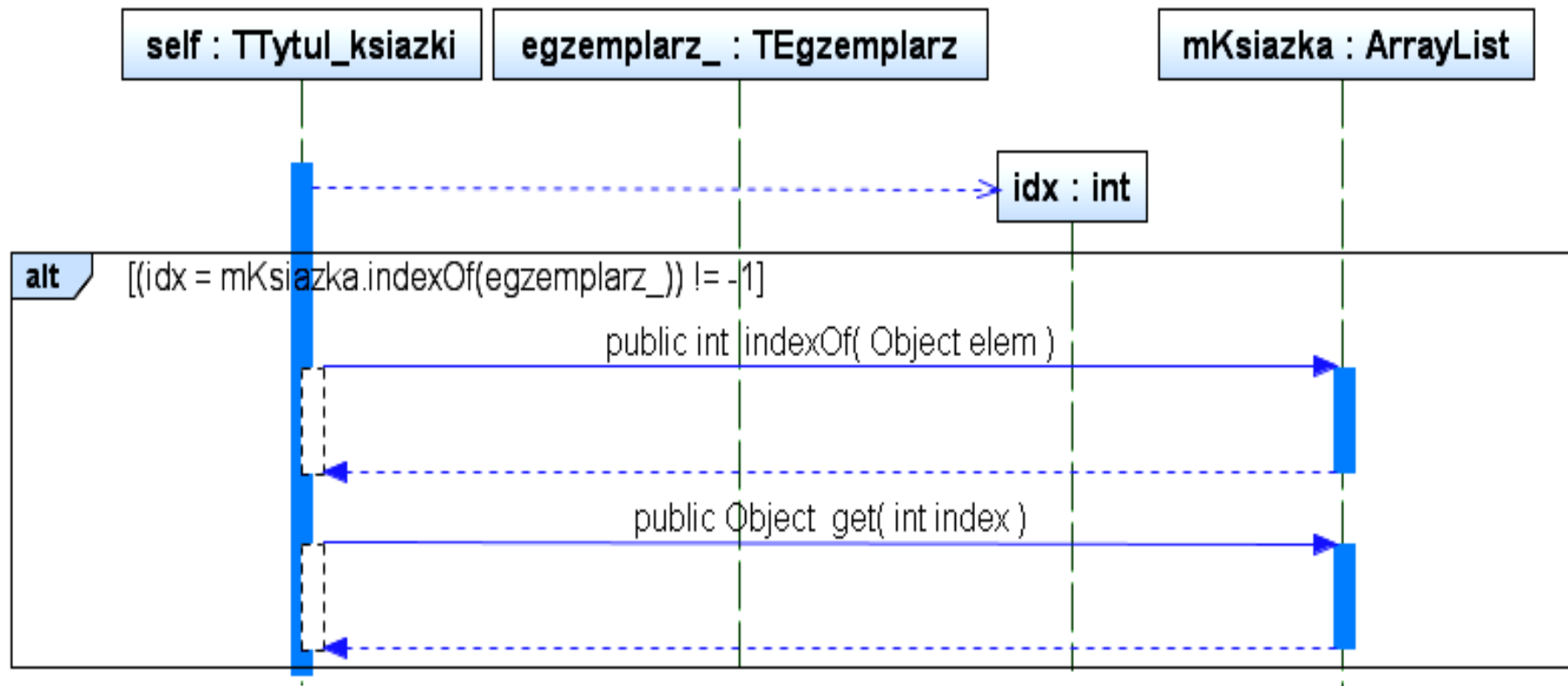
Projekt przypadku użycia
„Szukaj egzemplarz”

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

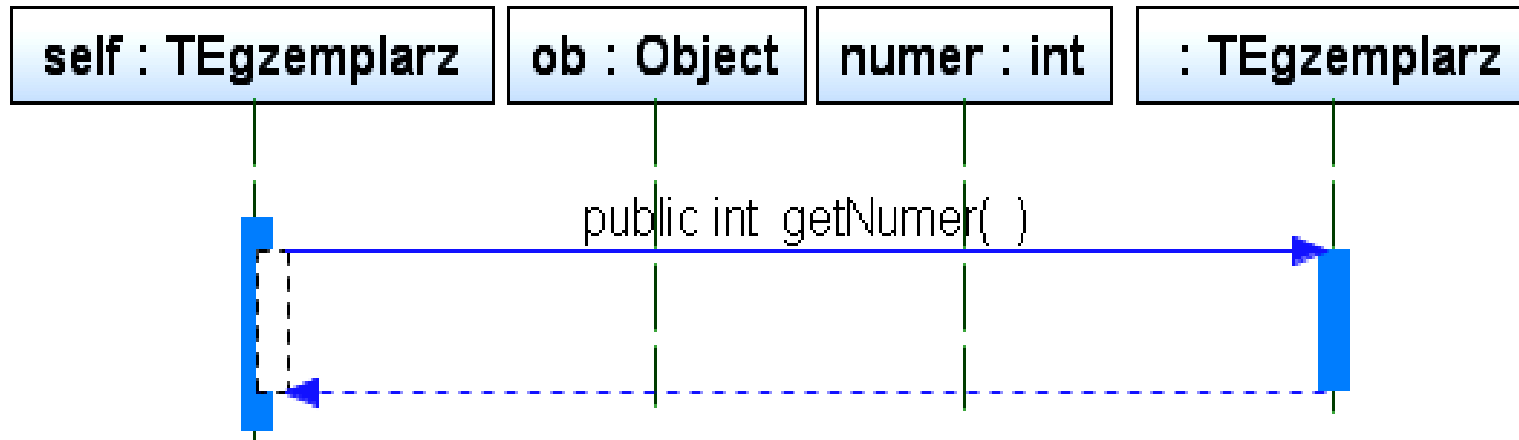
Definiowanie kodu metod realizujących przypadek użycia na podstawie diagramów sekwencji

(3) Szukaj egzemplarz

TEgzemplarz TTytul_ksiazki::Szukaj_egzemplarz(TEgzemplarz egzemplarz_)



boolean TEgzemplarz::equals(Object ob)



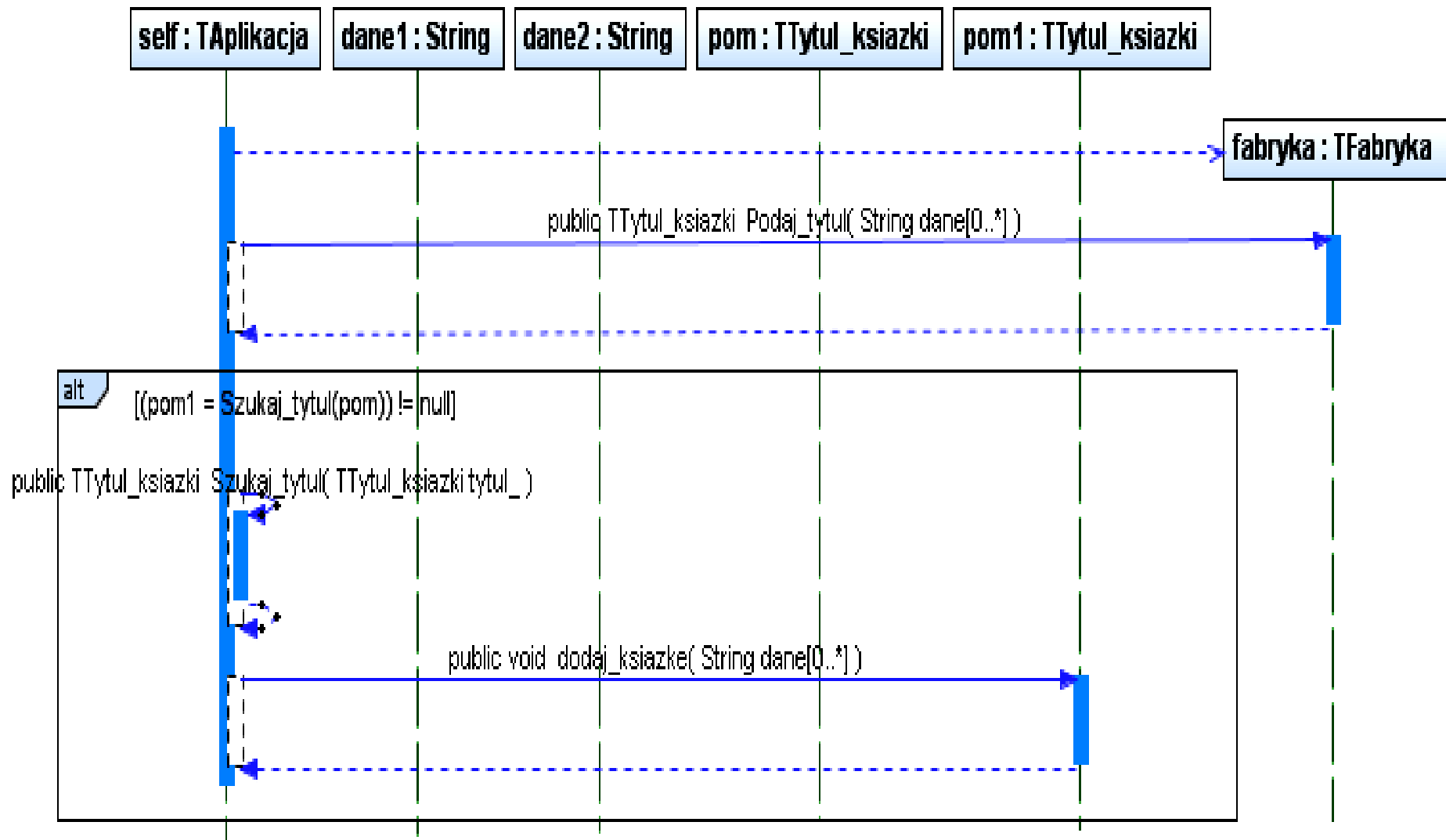
Projekt przypadku użycia
„ **Dodaj egzemplarz** ”

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

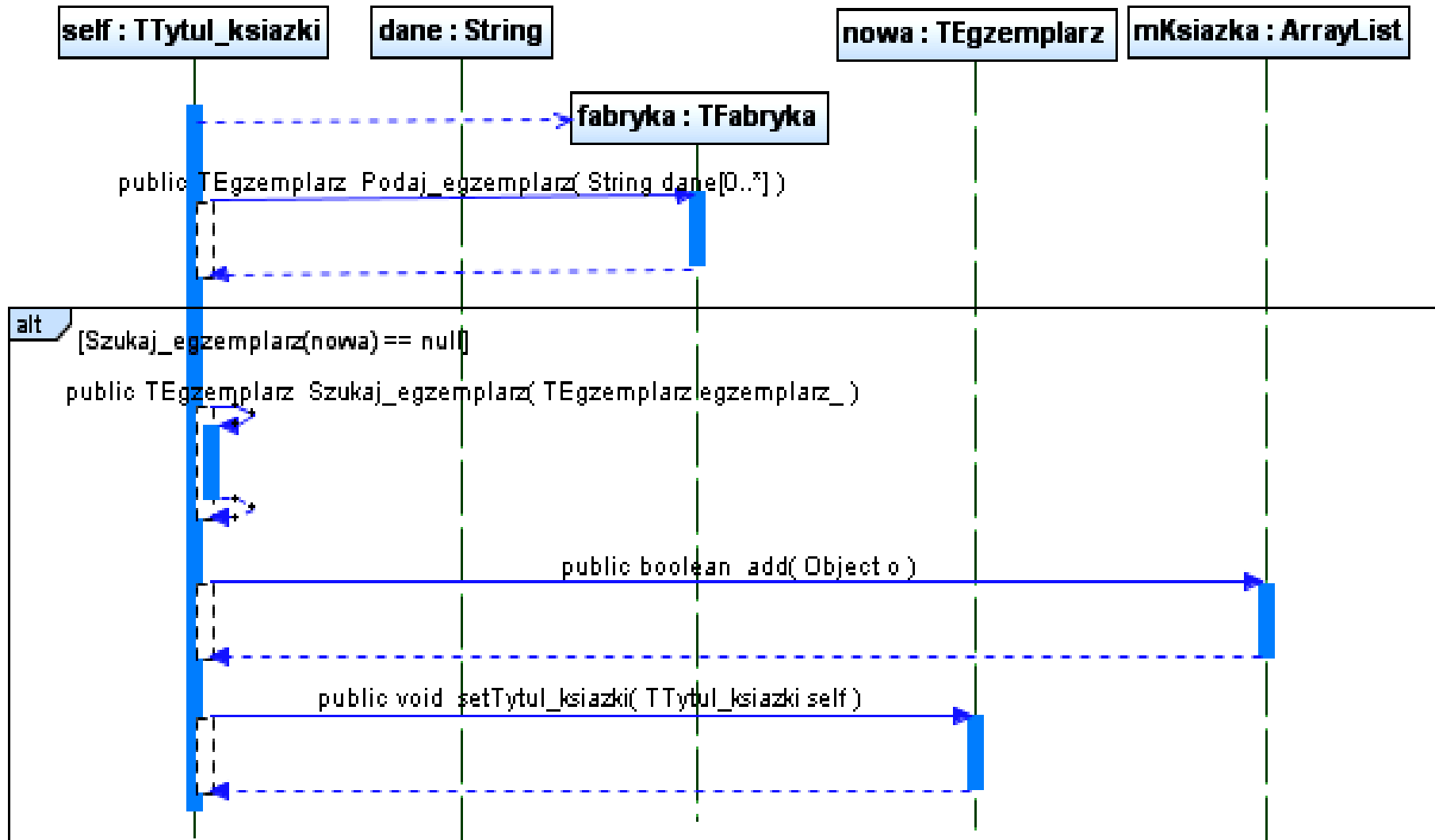
Definiowanie kodu metod realizujących przypadek użycia na podstawie diagramów sekwencji

(4) Dodaj egzemplarz

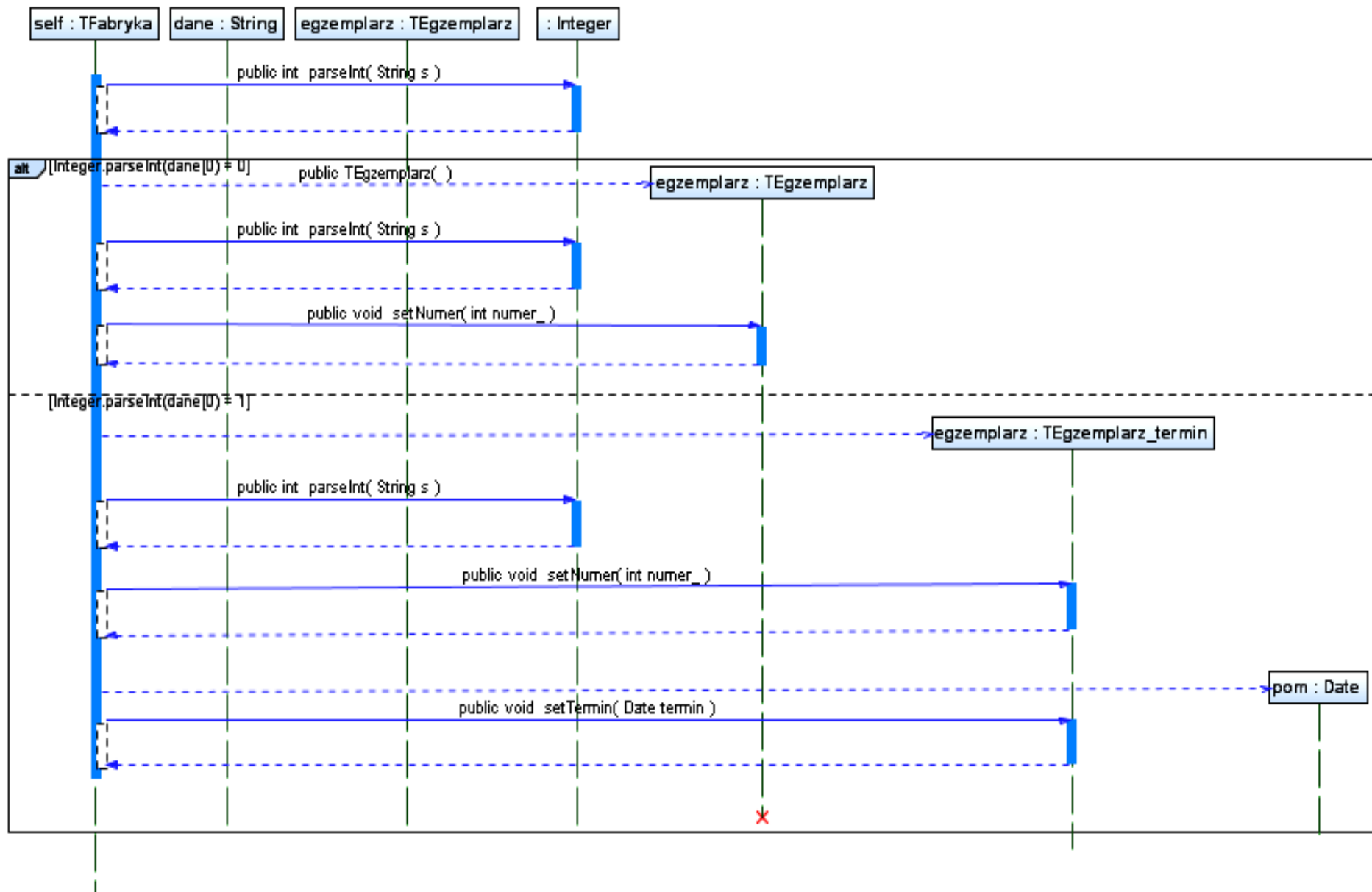
TTytul_książki TAplikacja::dodaj_książke(String dane1[], String dane2[])



void TTytul_książki::dodaj_książke(String dane[])



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_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());  }

}

```

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]
```

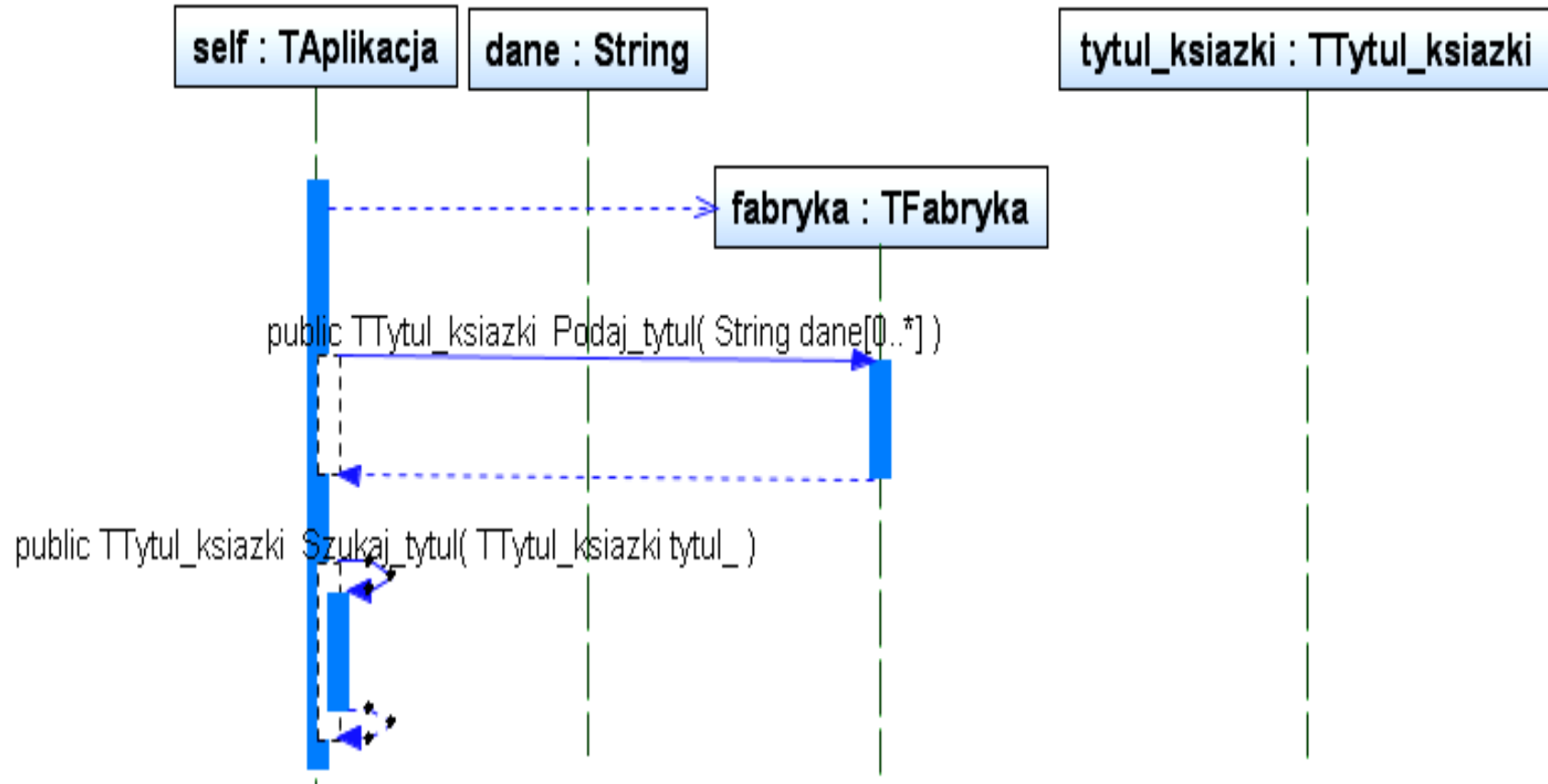
Projekt przypadku użycia
„ **Wyszukiwanie tytułów** ”

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_ksiazki **TAplikacja::Wyszukaj_tytul (String[] dane)**



Projekt przypadku użycia

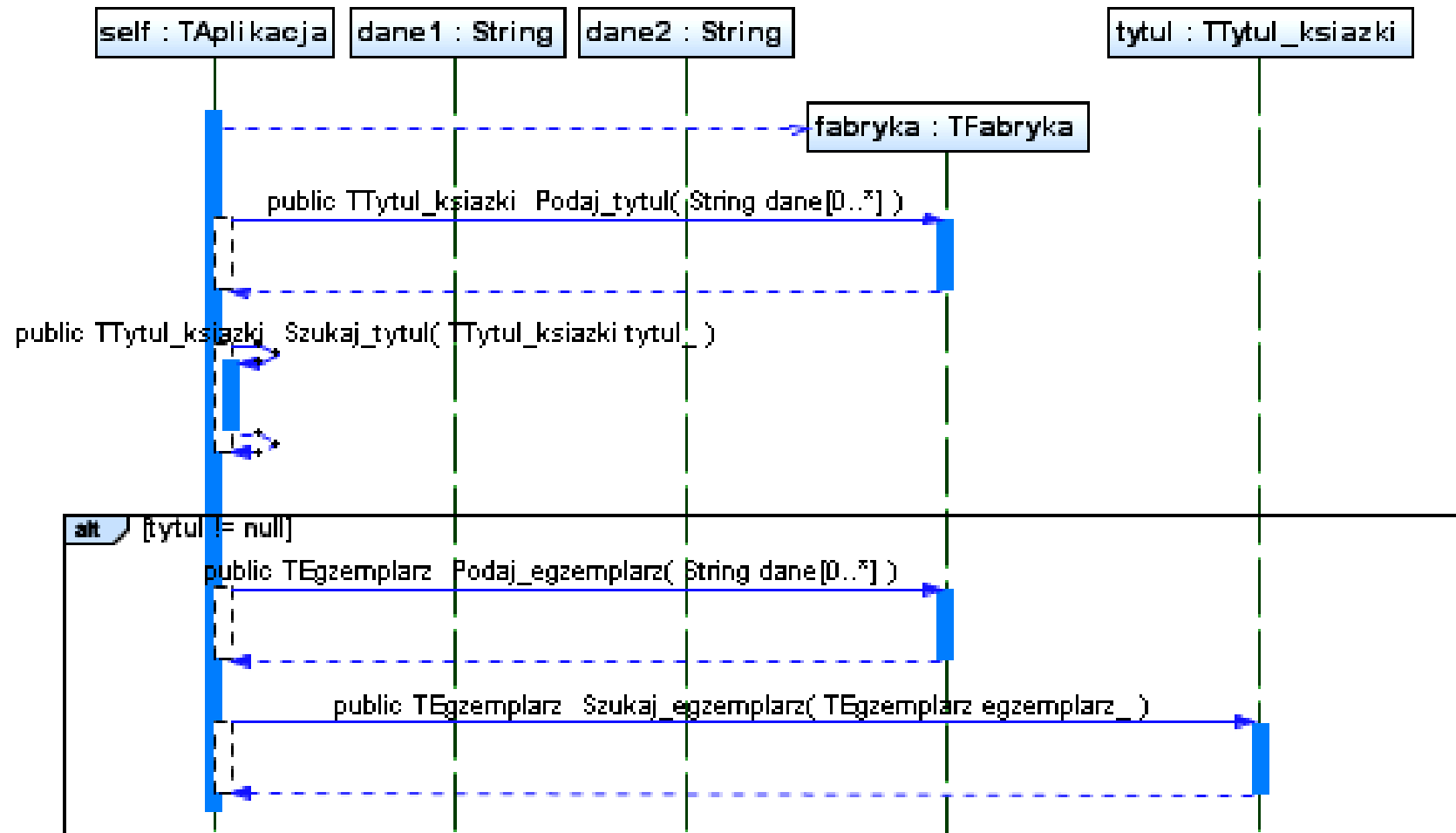
„Wyszukiwanie egzemplarzy”

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 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) { /* */ }
    public void Wyświetl_książki ()     { /* */ }
    public void Wyświetl_tytuły ()      { /* */ }
    public ArrayList<TTytul_książki> getTytul_książki () { /* */ }
    public 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_)     { /* */ }
    public TTytul_książki getTytul_książki () { /* */ }
    public 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_książki Podaj_tytul (String[] dane)      {/** */           }
    public TEgzemplarz Podaj_egzemplarz (String[] dane)    {/** */           }
}
```

```

package wypożyczalnia1;
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 wypożyczalnia1;
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 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
}
```

**// 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());    }

ap.Wyswietl_tytuly();
ap.Wyswietl_ksiazki();

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

```
ca. 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]

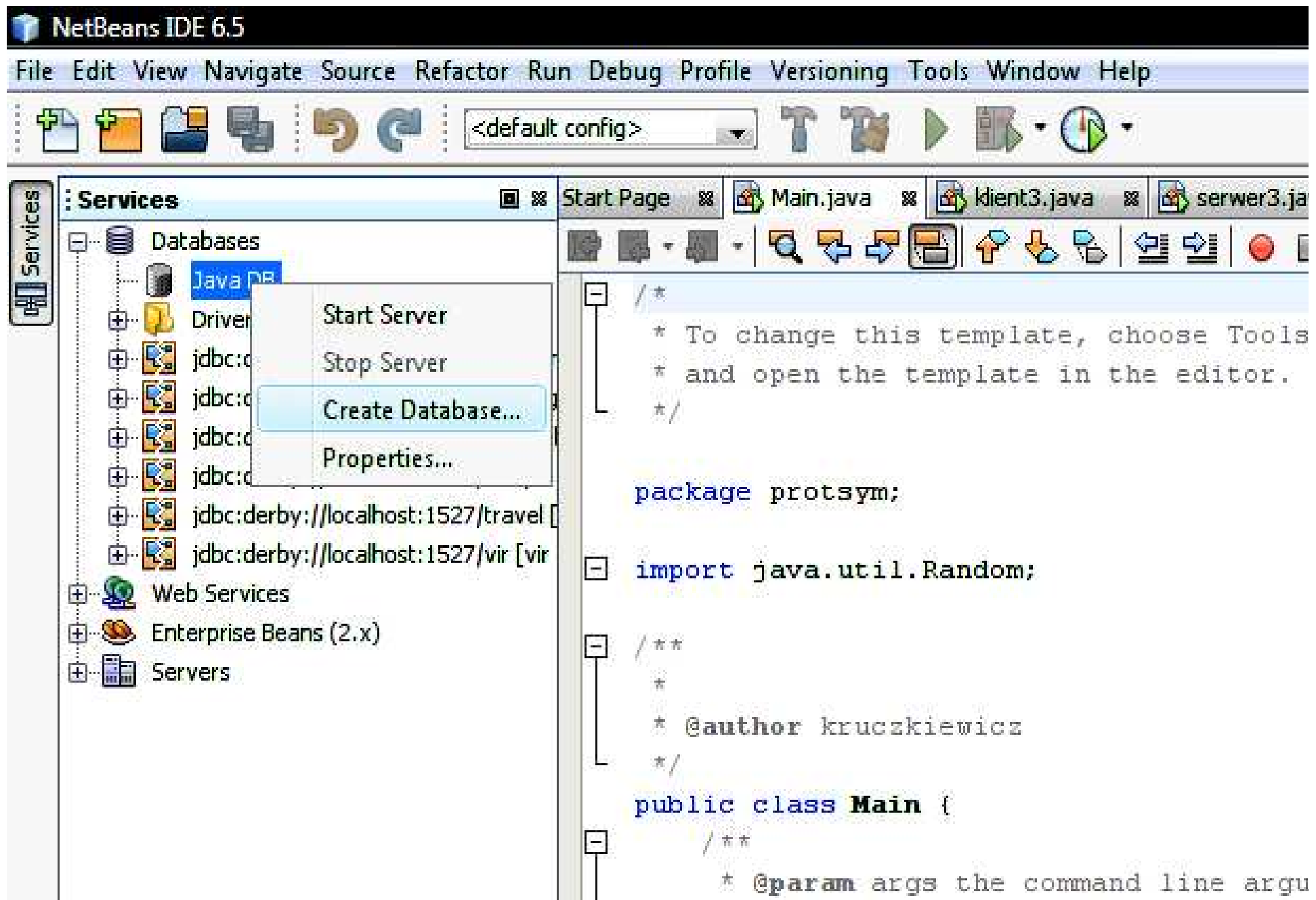
Tytuly ksiazek
Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1
Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2
Tytul: 3 Autor:3 ISBN: 3 Wydawnictwo:3
Tytul: 1 Autor:1 ISBN: 1 Wydawnictwo:1 Aktor:1
Tytul: 2 Autor:2 ISBN: 2 Wydawnictwo:2 Aktor:2
Tytul: 4 Autor:4 ISBN: 4 Wydawnictwo:4 Aktor:4

Ksiazki
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: 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: 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
```

7. Tworzenie bazy danych w systemie baz danych Derby

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
 - Start Server
 - Stop Server
 - Create Database...
 - Properties...
- Driver
- jdbc:...
- jdbc:...
- jdbc:...
- jdbc:...
- jdbc:derby://localhost:1527/travel
- jdbc:derby://localhost:1527/vir

Web Services

Enterprise Beans (2.x)

Servers

Main.java

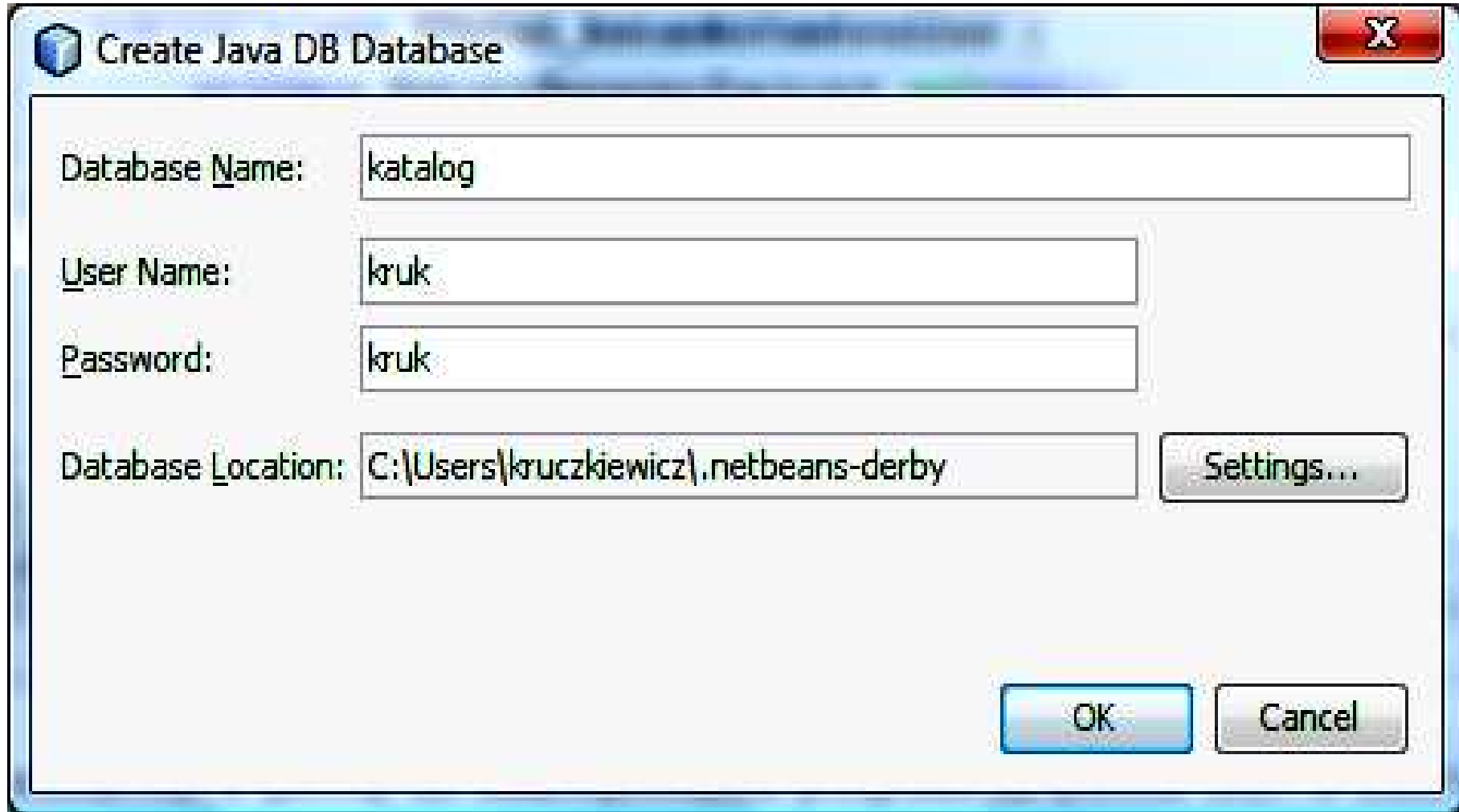
```
/*
 * To change this template, choose Tools
 * and open the template in the editor.
 */

package protsym;

import java.util.Random;

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

Zakładanie 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

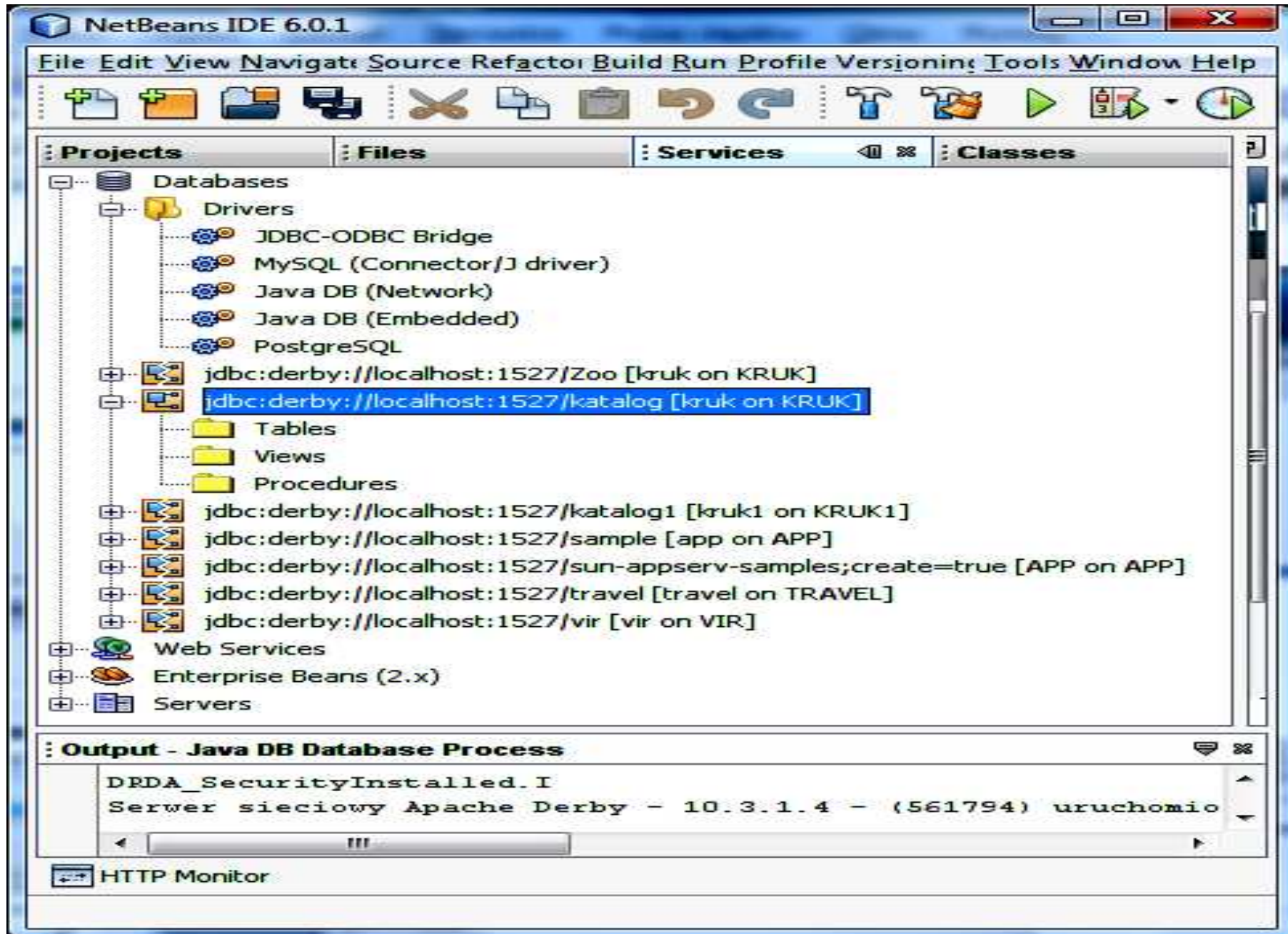
Łączenie z pustą bazą danych (3)

The screenshot displays the NetBeans IDE 6.0.1 interface. The main window shows the 'Welcome to NetBeans IDE 6.0.1' screen with a 'My NetBeans' sidebar. A context menu is open over a database connection in the 'Databases' tree, listing options: Connect..., Disconnect, Execute Command..., Refresh, Delete, and Properties. The 'Output - Java DB Database Process' window at the bottom shows the following text:

```
DRDA_SecurityInstalled.I  
Serwer sieciowy Apache Derby - 10.3.1.4 - (561794) uruchomiony i gotowy do zaakceptowania połączeń na porcie 1527 w 2008-04-16 09:34:18.928 GMT |
```

The Windows taskbar at the bottom shows the system tray with the time 11:55 and the language set to PL. The taskbar includes icons for System Windows, .netbeans-derby, Microsoft PowerPoint, and NetBeans IDE 6.0.1.

Połączenie z pustą bazą danych (4)



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

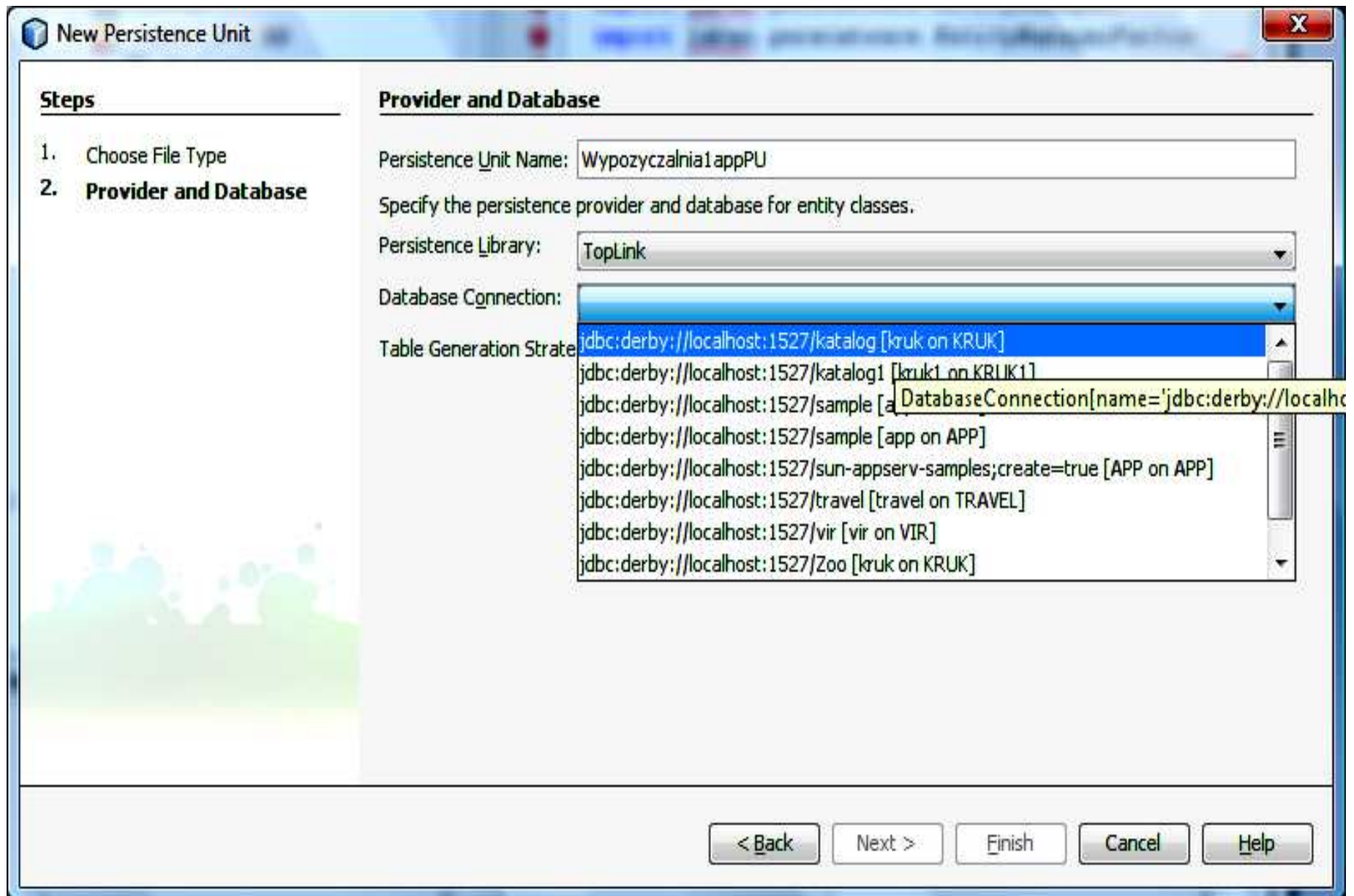
Wstawianie do projektu typu **Java Application** modułu typu **Persistence Unit (1)**

The screenshot shows an IDE window with a project named 'Wypożyczalnia' and a class 'TTytul_książkiController.java' open. The 'New' menu is open, and the 'Persistence Unit...' option is selected. The code editor shows the following 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("WypożyczalniaappPU");  
        }  
        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 =
```

The IDE interface includes a toolbar at the top, a 'Projects' view on the left, and a status bar at the bottom showing '45:42 INS'.

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



Tworzenie modułu utrwalania danych dla technologii TopLink (3)

New Persistence Unit

Steps

1. Choose File Type
2. **Provider and Database**

Provider and Database

Persistence Unit Name: Wypożyczalnia1appPU

Specify the persistence provider and database for entity classes.

Persistence Library: TopLink

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

Table Generation Strategy: Create Drop and Create None

< Back Next > **Finish** Cancel Help

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)

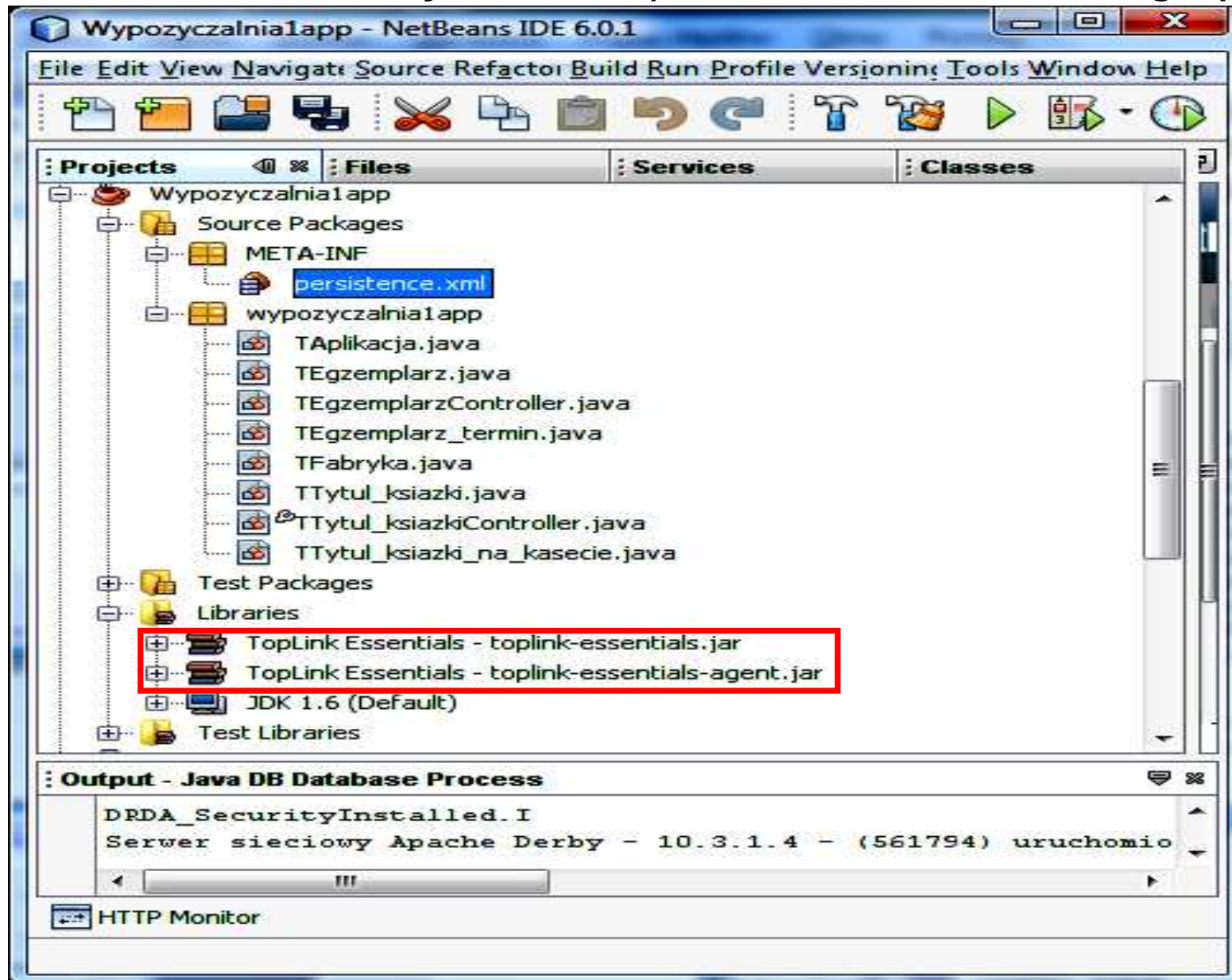
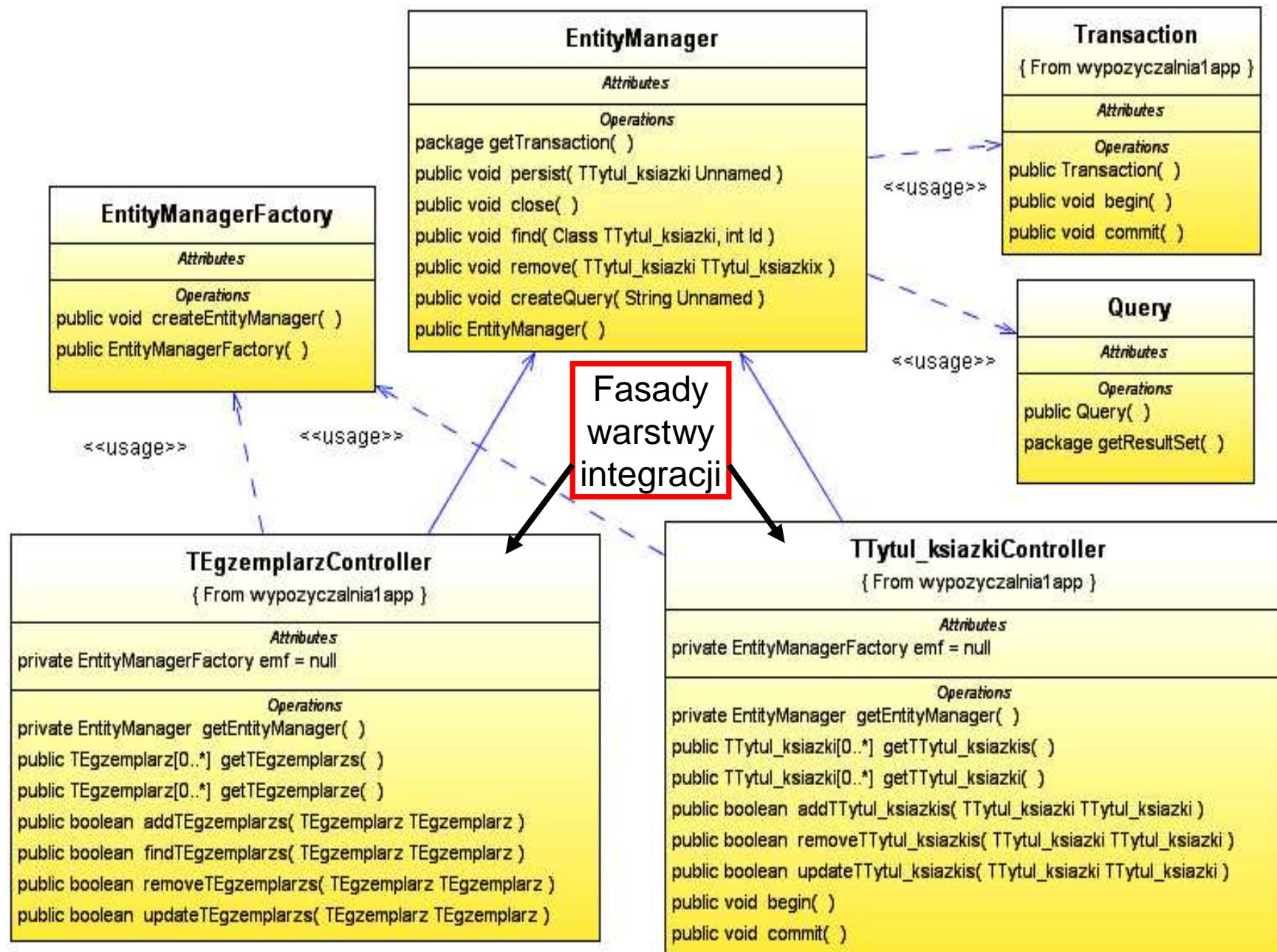
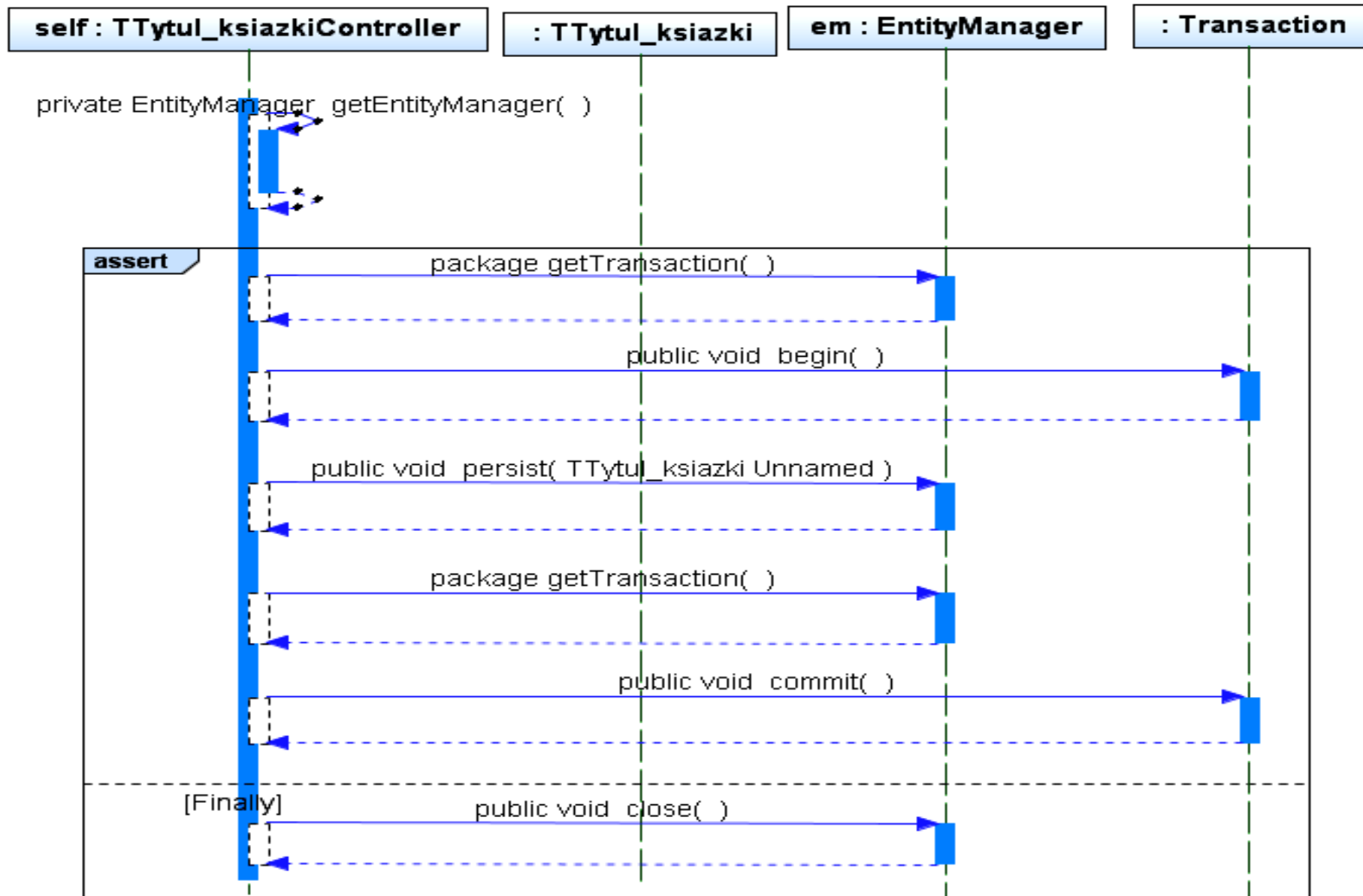


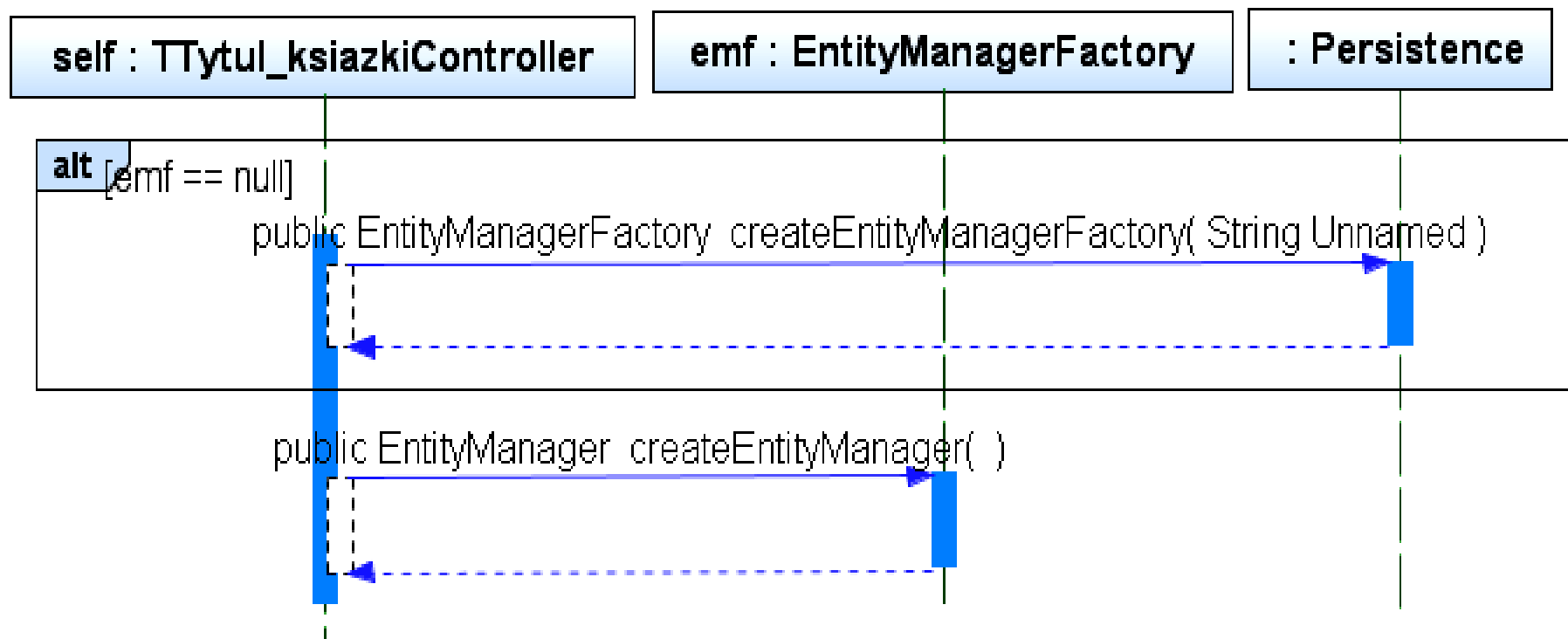
Diagram klas – uproszczony schemat warstwy integracji (5)



Dodawanie tytułów do bazy danych - public boolean addTytuł_książkis(TTytuł_książki TTytuł_książki) (6)



Dodawanie tytułów do bazy danych - **private EntityManager getEntityManager()** (7)



Klasa typu Controller dla każdej z klas utrwalanych obiektów – (8) Realizacja wzorców typu **Domain Store** i **Transfer Object**

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

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

```
public boolean addTTytul_ksiazki( ArrayList<TTytul_ksiazki> tytuly)
{
    EntityManager em = getEntityManager();
    TTytul_ksiazki newTTytul_ksiazki=null;
    try {
        Iterator it = tytuly.iterator();
        em.getTransaction().begin();
        while (it.hasNext())
        {
            newTTytul_ksiazki = (TTytul_ksiazki) it.next();
            if (newTTytul_ksiazki.getId()==null)
                em.merge(newTTytul_ksiazki);
        }
        em.persist(newTTytul_ksiazki);
        em.getTransaction().commit();
    } finally {
        em.close();
        return false; }
}
```

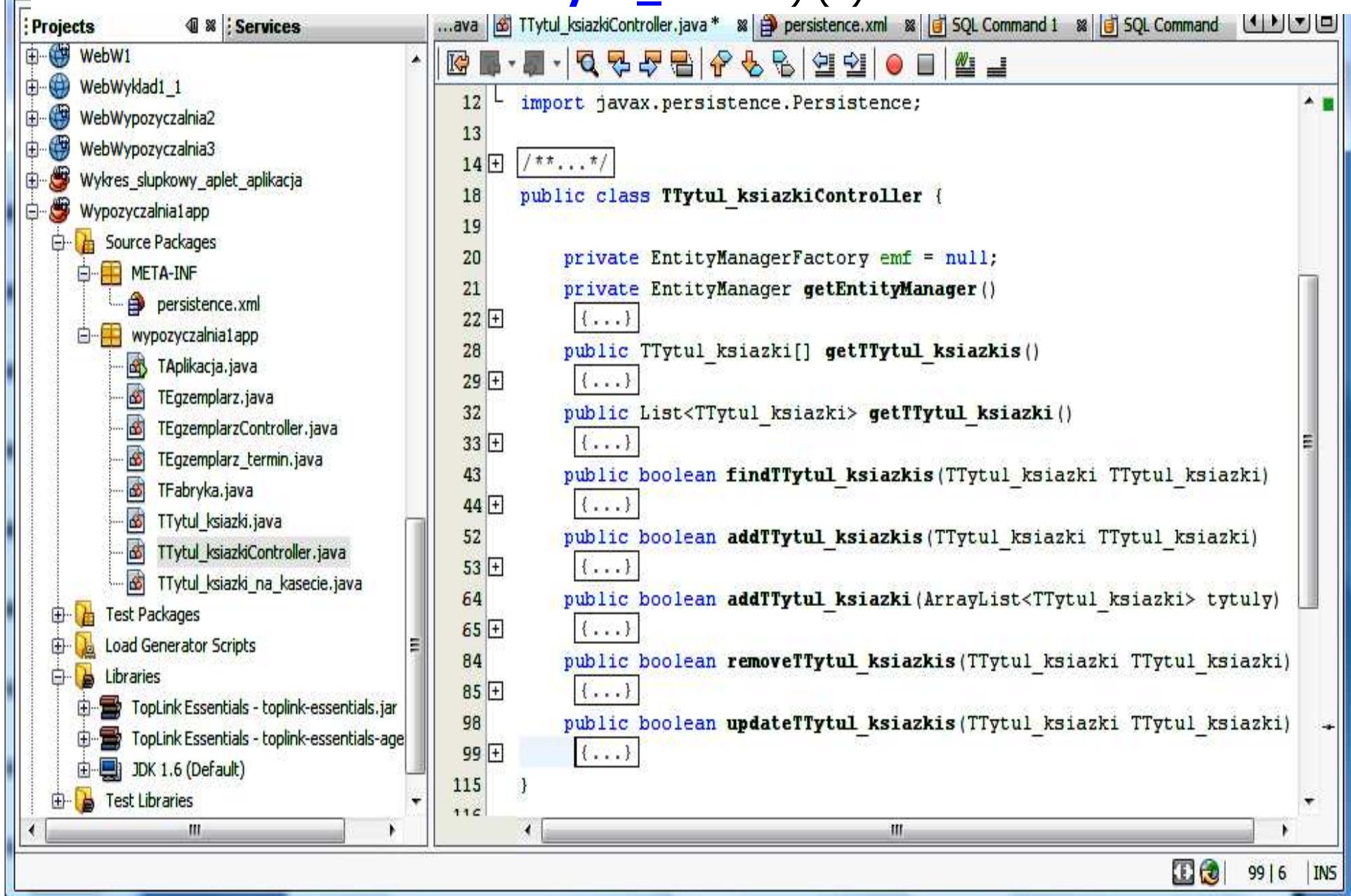
```
public boolean removeTTytul_ksiazki (TTytul_ksiazki TTytul_ksiazki)
{
    EntityManager em = getEntityManager();
    try {
        em.getTransaction().begin();
        TTytul_ksiazki TTytul_ksiazkix =
            em.find(TTytul_ksiazki.class, TTytul_ksiazki.getId());
        em.remove(TTytul_ksiazkix);
        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_książki[] getTTytul_książkis()
{
    return (TTytul_książki[]) getTTytul_książkis().toArray(
        new TTytul_książki[0]);
}
```

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


Klasa warstwy integracji (szablon) – dla każdej klasy typu **Entity** (tutaj dla **TTytul_książki**) (9)

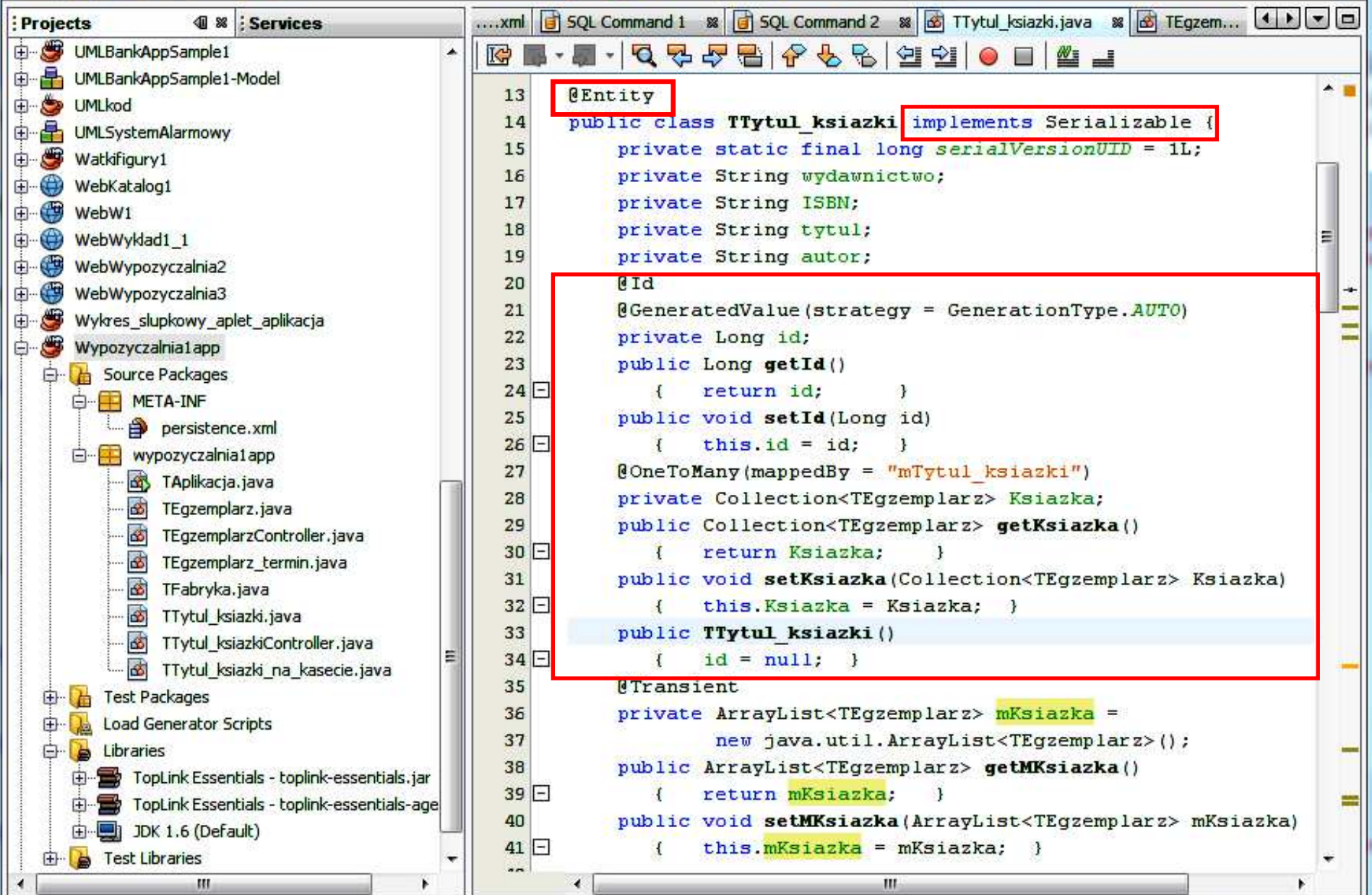


The screenshot displays an IDE window with the following components:

- Projects:** A tree view on the left showing a project named 'Wypożyczalnia1app' with sub-packages 'Source Packages' and 'Test Packages'. The 'Source Packages' folder contains several Java files, including 'TTYtul_książkiController.java' which is currently selected.
- Code Editor:** The main area shows the source code of 'TTYtul_książkiController.java'. The code includes an import for 'javax.persistence.Persistence', a package comment, and the class definition for 'TTYtul_książkiController'. The class contains a private 'EntityManagerFactory emf' field, a private 'EntityManager' field, and several public methods: 'getTTYtul_książkis()', 'getTTYtul_książki()', 'findTTYtul_książkis()', 'addTTYtul_książkis()', 'addTTYtul_książki()', 'removeTTYtul_książkis()', and 'updateTTYtul_książkis()'. Each method body is currently empty, indicated by '(...)'.
- Taskbar:** At the bottom right, the system tray shows the taskbar with the text '99 | 6 | INS'.

```
12  import javax.persistence.Persistence;
13
14  /**...*/
18  public class TTYtul_książkiController {
19
20      private EntityManagerFactory emf = null;
21      private EntityManager entityManager;
22      (...)
28      public TTYtul_książki[] getTTYtul_książkis()
29      (...)
32      public List<TTYtul_książki> getTTYtul_książki()
33      (...)
43      public boolean findTTYtul_książkis(TTYtul_książki TTYtul_książki)
44      (...)
52      public boolean addTTYtul_książkis(TTYtul_książki TTYtul_książki)
53      (...)
64      public boolean addTTYtul_książki(ArrayList<TTYtul_książki> tytuły)
65      (...)
84      public boolean removeTTYtul_książkis(TTYtul_książki TTYtul_książki)
85      (...)
98      public boolean updateTTYtul_książkis(TTYtul_książki TTYtul_książki)
99      (...)
115 }
116
```

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



```
13  @Entity
14  public class TTytul_książki 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_książki")
29      private Collection<TEgzemplarz> Książka;
30      public Collection<TEgzemplarz> getKsiążka()
31          { return Książka; }
32      public void setKsiążka(Collection<TEgzemplarz> Książka)
33          { this.Książka = Książka; }
34      public TTytul_książki()
35          { id = null; }
36
37      @Transient
38      private ArrayList<TEgzemplarz> mKsiążka =
39          new java.util.ArrayList<TEgzemplarz>();
40      public ArrayList<TEgzemplarz> getMKsiążka()
41          { return mKsiążka; }
42      public void setMKsiążka(ArrayList<TEgzemplarz> mKsiążka)
43          { this.mKsiążka = mKsiążka; }
```

The image shows an IDE window with the following components:

- Menu Bar:** File, Edit, View, Navigate, Source, Refactor, Build, Run, Profile, Versioning, Tools, Window, Help.
- Toolbar:** Standard IDE icons for file operations, search, and execution.
- Project Explorer (Left):** Shows a project named 'Wypożyczalnia1app' with source packages (META-INF, wypożyczalnia1app) and test packages. The 'wypożyczalnia1app' package contains several Java files, including 'TTytul_książki_na_kasecie.java' which is currently selected.
- Code Editor (Right):** Displays the source code for 'TTytul_książki_na_kasecie.java'. The code is as follows:

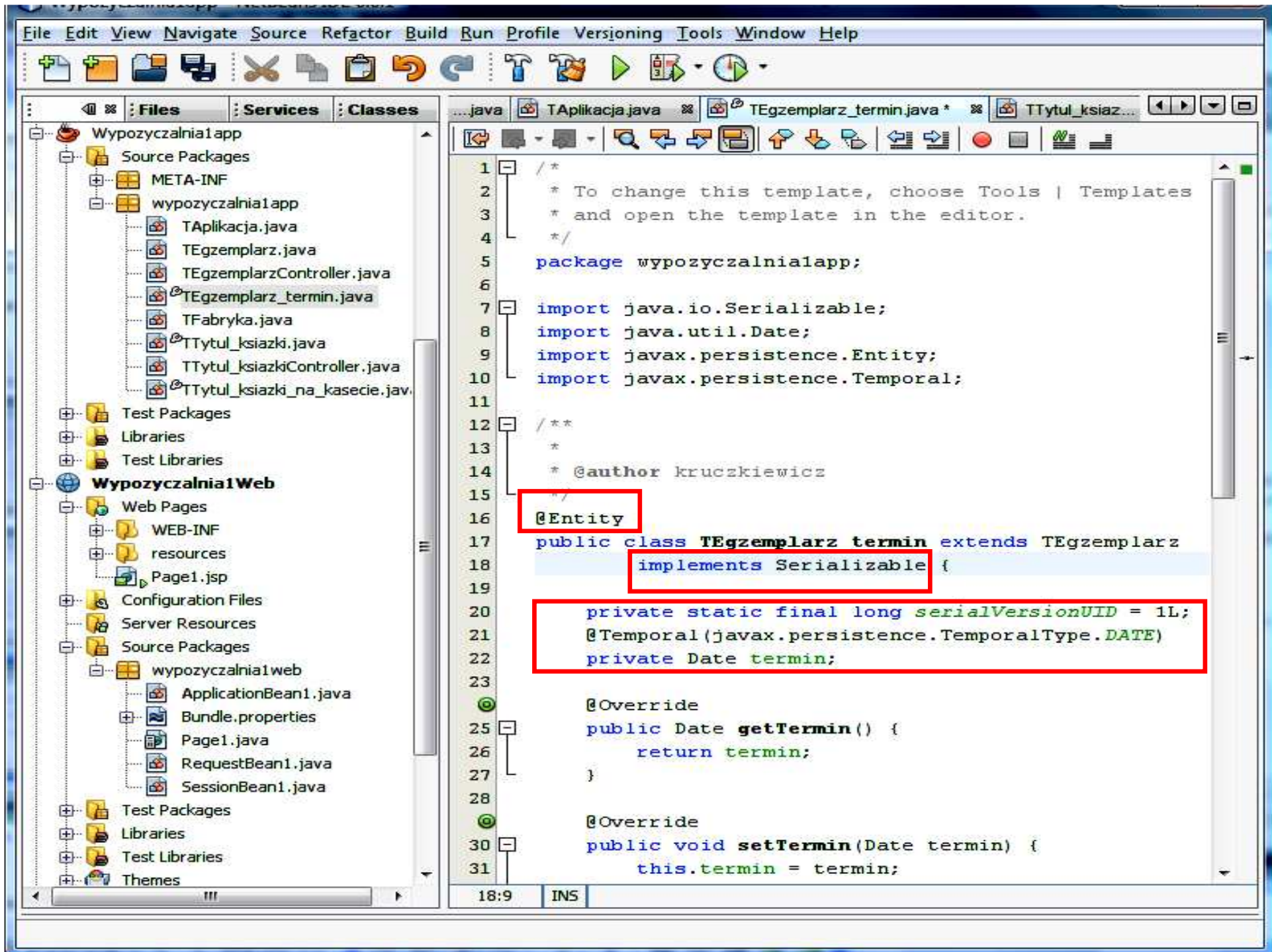
```
5 package wypożyczalnia1app;
6
7 import java.io.Serializable;
8 import javax.persistence.Entity;
9
10 /**
11  *
12  * @author kruczkiewicz
13  */
14 @Entity
15 public class TTytul_książki_na_kasecie extends TTytul_książki
16     implements Serializable {
17
18     private static final long serialVersionUID = 1L;
19     private String aktor;
20
21     @Override
22     public String getAktor() {
23         return aktor;
24     }
25
26     @Override
27     public void setAktor(String aktor) {
28         this.aktor = aktor;
29     }
30
31     /* @Override
32     public boolean equals(Object object) {
```
- Status Bar (Bottom):** Shows the time '16:38' and the cursor position 'INS'.

Zmiana typu klas danych na typ @Entity (11) – dodano adnotacje, nowy atrybut (Id) z metodami. **Należy zestandaryzować nazwy metod dostępu do składowej mTytul_ksiazki!**

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

The screenshot shows an IDE window with the following content:

- Projects:** A tree view on the left showing a project structure with packages like 'wypozyczalnia1app' and 'META-INF'. The file 'TEgzemplarz.java' is selected.
- Code Editor:** The main window displays the Java code for 'TEgzemplarz.java'. The code is annotated with red boxes:
 - Line 11: `@Entity`
 - Line 12: `implements Serializable`
 - Lines 15-17: `@Id` and `@GeneratedValue(strategy = GenerationType.AUTO)` annotations.
 - Lines 18-23: `setId` and `getId` methods.
 - Lines 24-25: `@ManyToOne` annotation.
 - Lines 26-28: `getMTytul_ksiazki` method.
 - Lines 29-31: `setMTytul_ksiazki` method.
 - Lines 32-34: Constructor `TEgzemplarz()`.



Wstawienie do modułu typu Persistence Unit wszystkich klas typu Entity (12)

The screenshot shows the NetBeans IDE interface. On the left, the 'Classes' pane displays a project structure with 'Wypożyczalnia1app' expanded to show source packages and classes. The 'persistence.xml' file is highlighted. The main editor area shows the 'Persistence Units' configuration for 'Wypożyczalnia1appPU'. The configuration includes:

- Persistence Unit Name: Wypożyczalnia1appPU
- Persistence Library: TopLink
- JDBC Connection: jdbc:derby://localhost:1527/katalog [kruk on KRUK]
- Table Generation Strategy: Create Drop and Create None
- Include All Entity Classes in "Wypożyczalnia1app" Module
- Include Entity Classes:
 - wypożyczalnia1app.TTytul_ksiazki
 - wypożyczalnia1app.TEgzemplarz
 - wypożyczalnia1app.TEgzemplarz_termin
 - wypożyczalnia1app.TTytul_ksiazki_na_kasecie

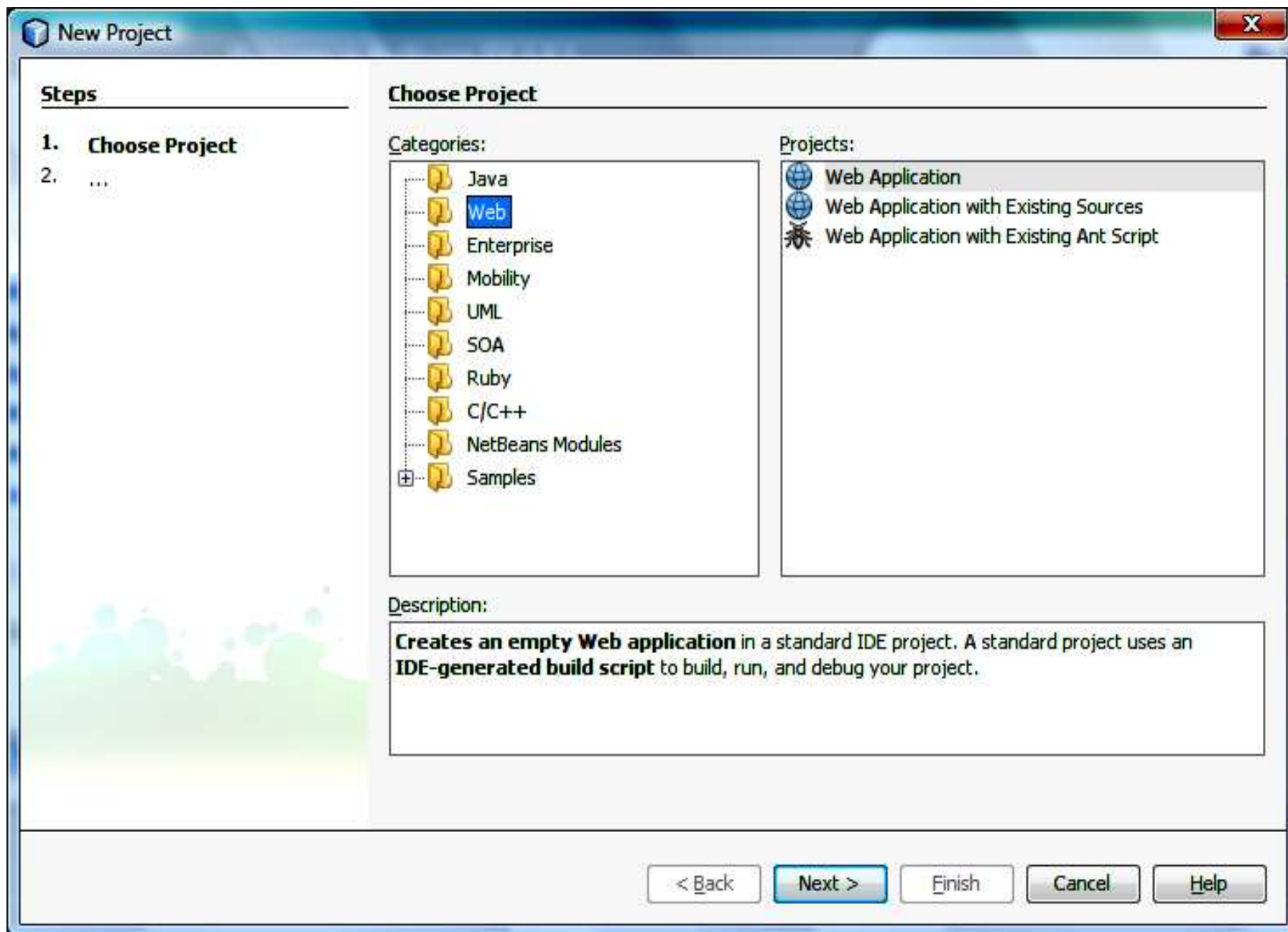
The 'Include All Entity Classes...' checkbox and the list of entity classes are highlighted with a red rectangle. The 'Add Class...' and 'Remove' buttons are visible next to the list.

9. 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)



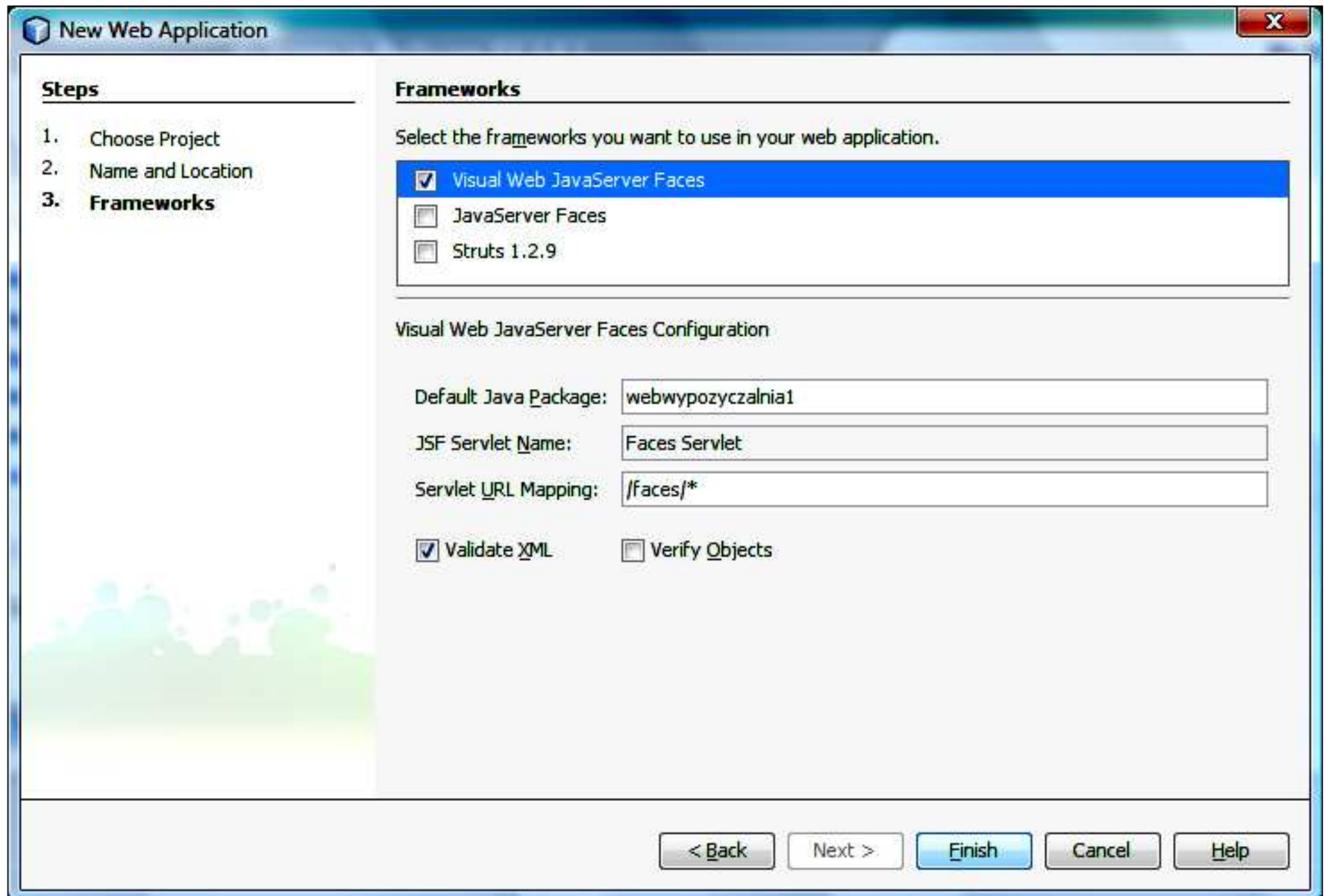
Projekt typu WebApplication tworzony w tym samym katalogu, w którym znajduje się projekt z warstwą biznesową (2)

The screenshot shows the 'New Web Application' wizard in an IDE. The window title is 'New Web Application'. On the left, there is a 'Steps' pane with three steps: 1. Choose Project, 2. Name and Location (which is currently selected), and 3. Frameworks. The main area is titled 'Name and Location' and contains the following fields and controls:

- Project Name:** WebWypożyczalnia1
- Project Location:** E:\Dydaktyka\d1\Komponenty_1\proby (with a 'Browse...' button)
- Project Folder:** E:\Dydaktyka\d1\Komponenty_1\proby\WebWypożyczalnia1
- Add to Enterprise Application:** <None>
- Server:** GlassFish V2 (with an 'Add...' button)
- Java EE Version:** Java EE 5
- Context Path:** /WebWypożyczalnia1
- Set as Main Project

At the bottom of the wizard, there are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'. The 'Next >' button is highlighted in blue.

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



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

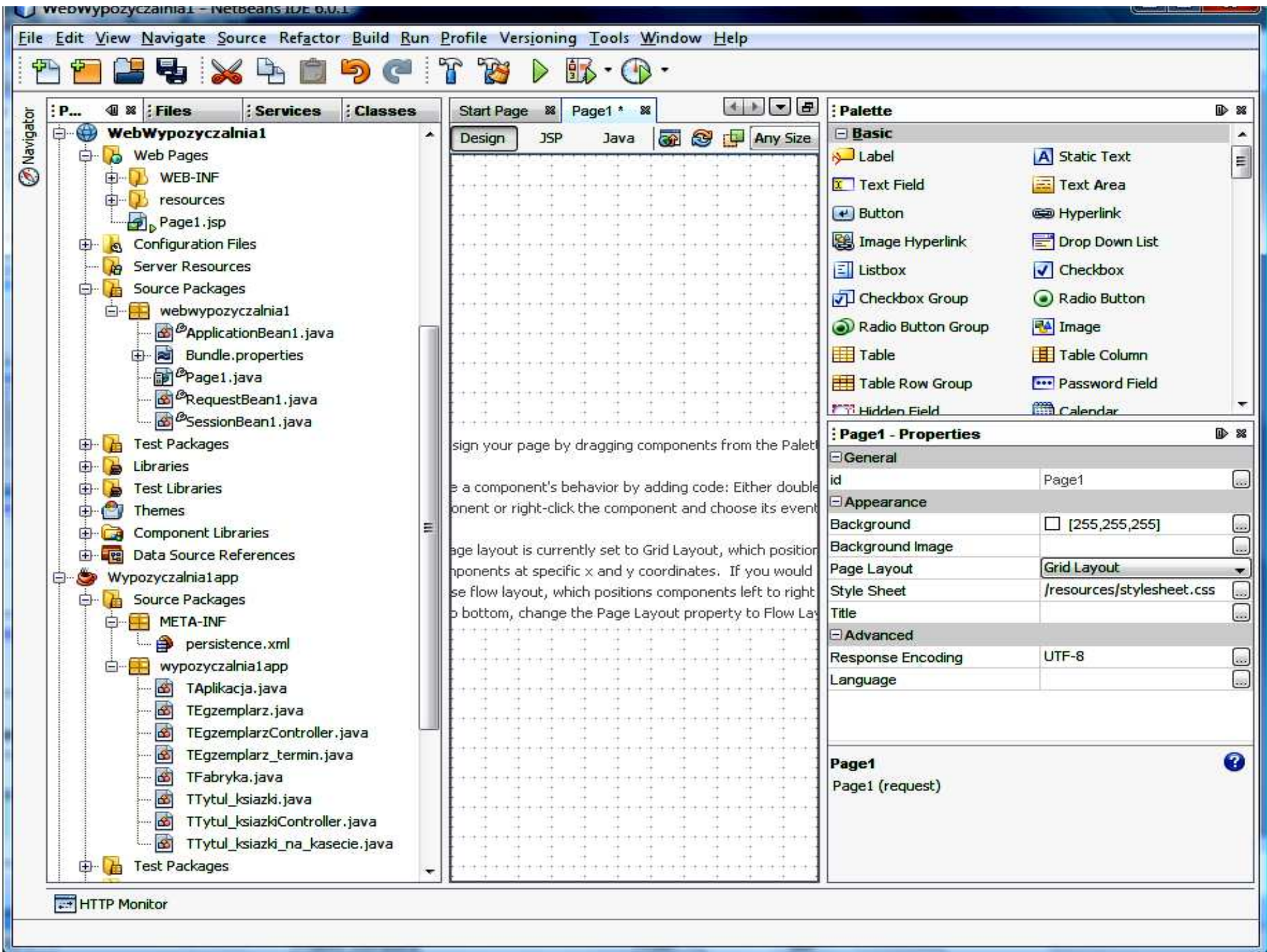
Default Java Package:

JSF Servlet Name:

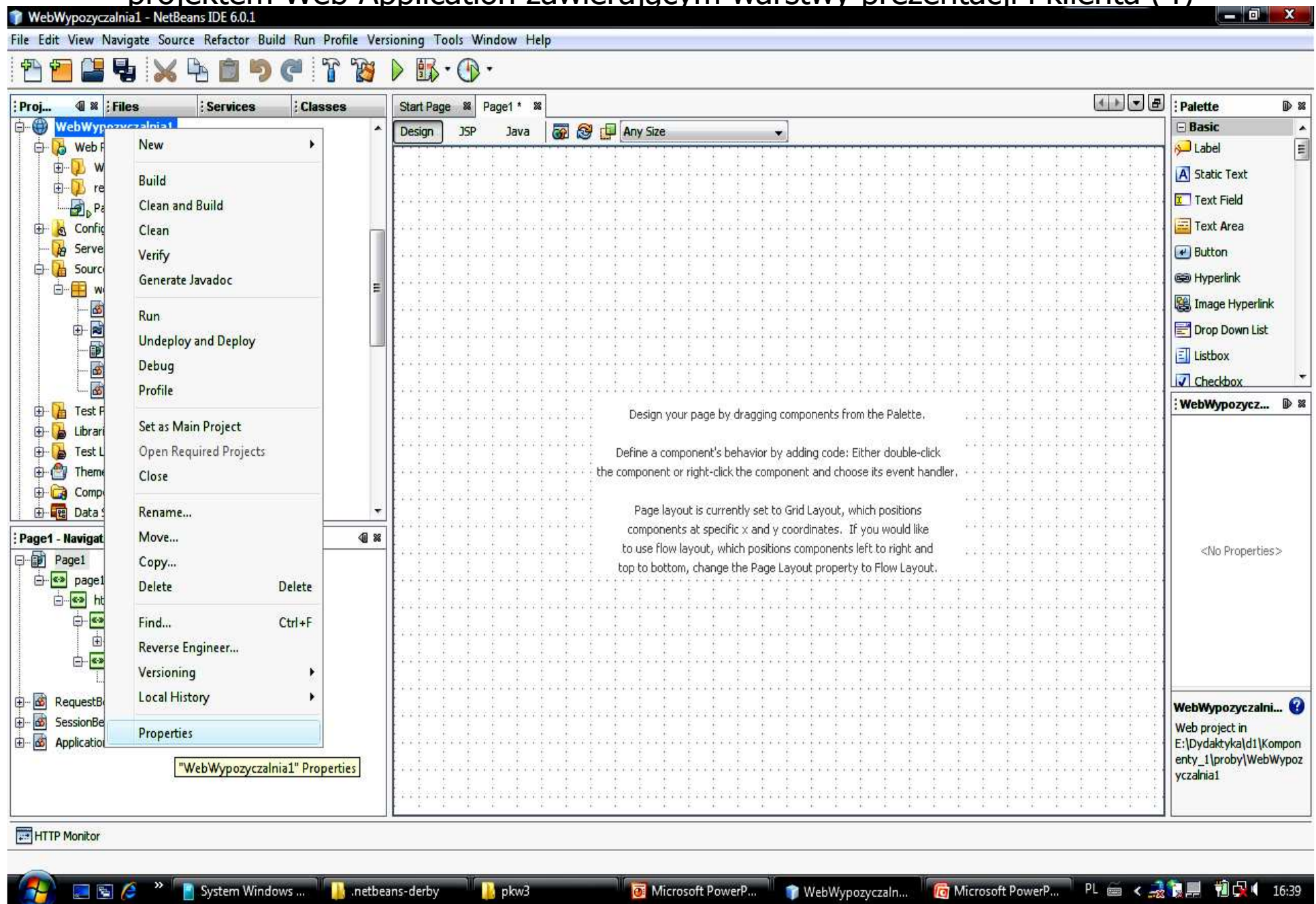
Servlet URL Mapping:

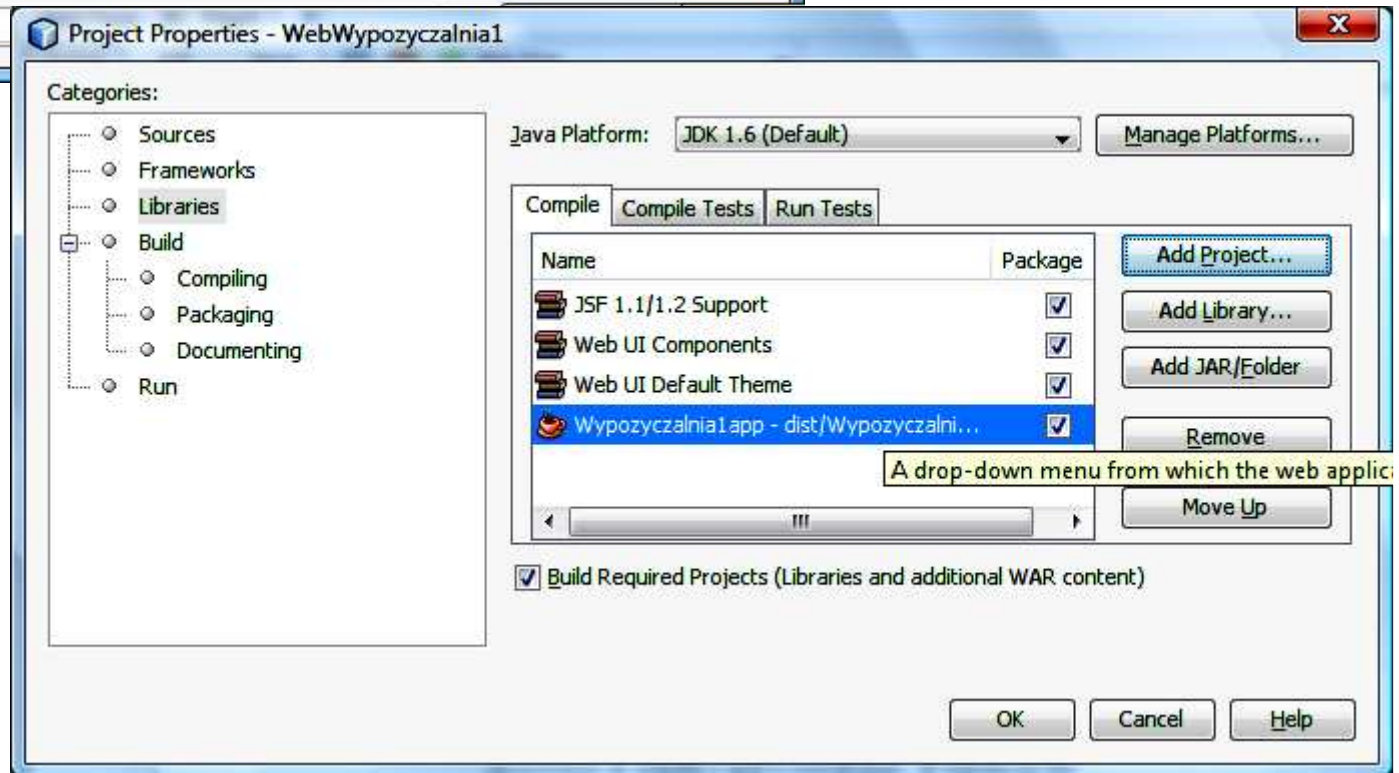
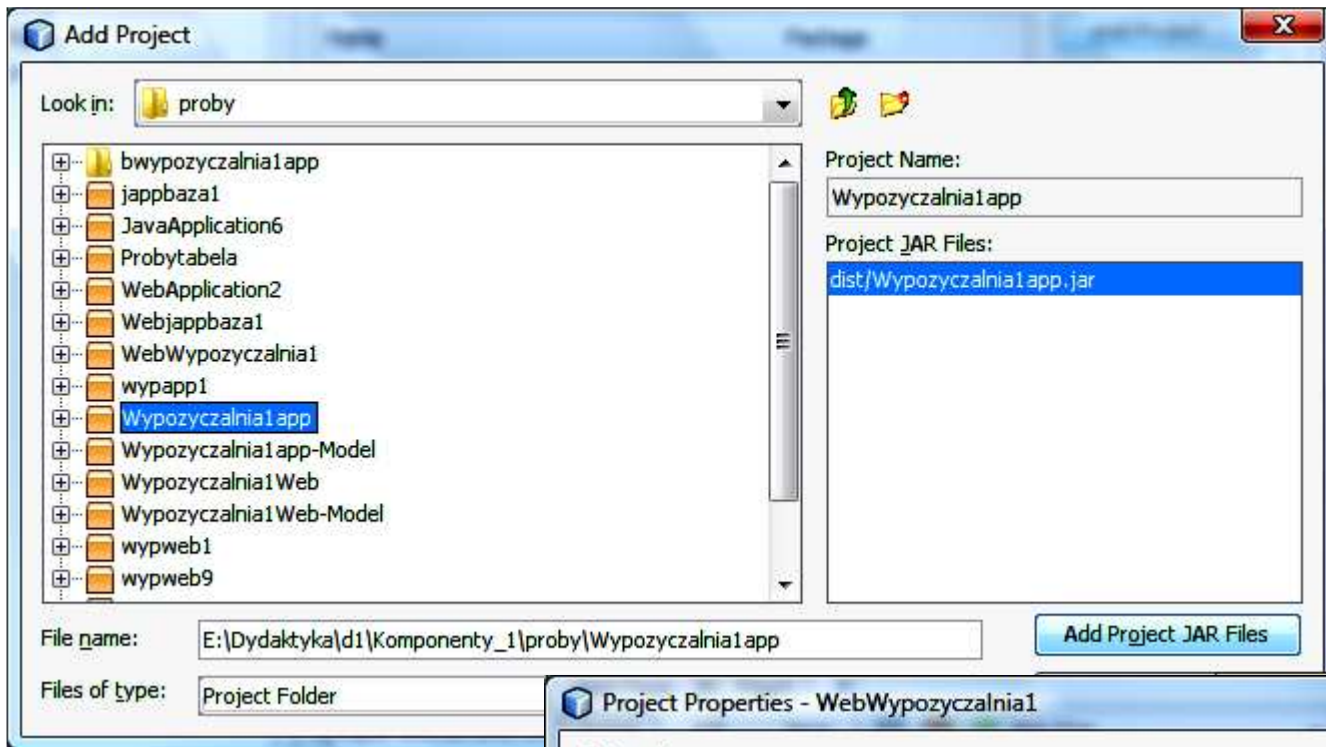
Validate XML Verify Objects

< Back Next > **Finish** Cancel Help

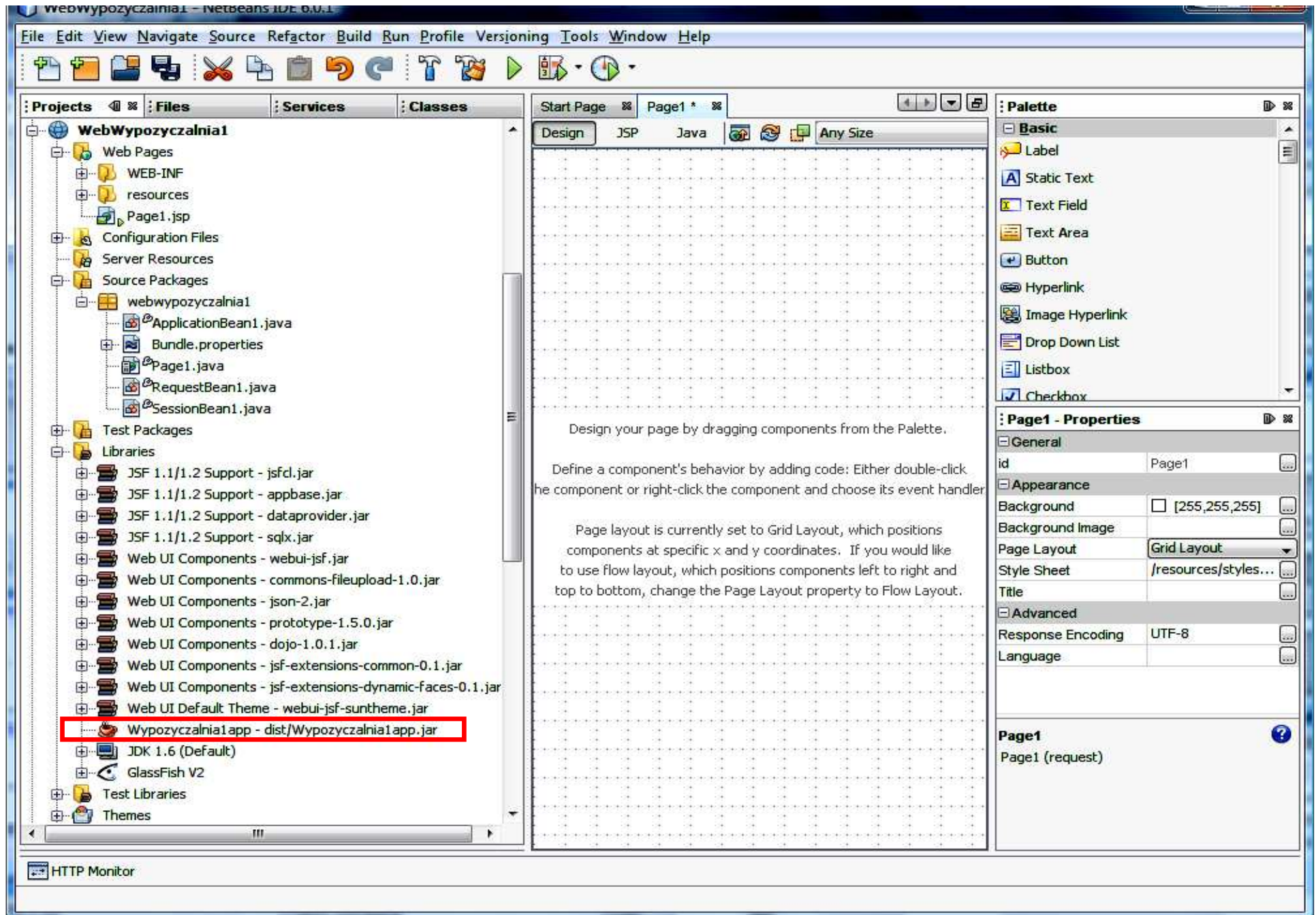


Łączenie projektu typu JavaApplication zawierającym warstwę biznesową i integracji z projektem Web Application zawierającym warstwy prezentacji i klienta (4)





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



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

The screenshot displays an IDE interface for a web application project named "WebWypożyczalnia2". The main workspace shows the design view of "Page1.jsp" in "Design" mode. The design view contains a header image of a sunset over water and a main content area with the text "Przykład wielowarstwowej aplikacji". Below the text are several blue hyperlinks: "Strona główna", "Dodaj tytuły w aplikacji", "Dodaj książki w aplikacji", "Dodaj tytuł do bazy", "Dodaj książkę do bazy", "Przepisz tytuły do bazy", and "Przepisz książki do bazy".

The left sidebar shows the project's file structure under "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

Arrows point from "Logo.jspf" and "Menu.jspf" in the file explorer to the corresponding image and menu area in the design view.

The bottom-left pane shows the "jsp:directive.include:Menu.jspf - Navigator" view, displaying the page structure:

- Page1
 - page1
 - html
 - head1
 - body1
 - form1
 - div
 - div
 - jsp:directive.include:Menu.jspf


Below the navigator are tabs for "RequestBean1", "SessionBean1", and "ApplicationBean1".

The bottom status bar shows "HTTP Monitor".

http://localhost:8080/WebWypożyczalnia2/faces/Page1.jsp - Windows Internet Explorer

http://localhost:8080/WebWyp

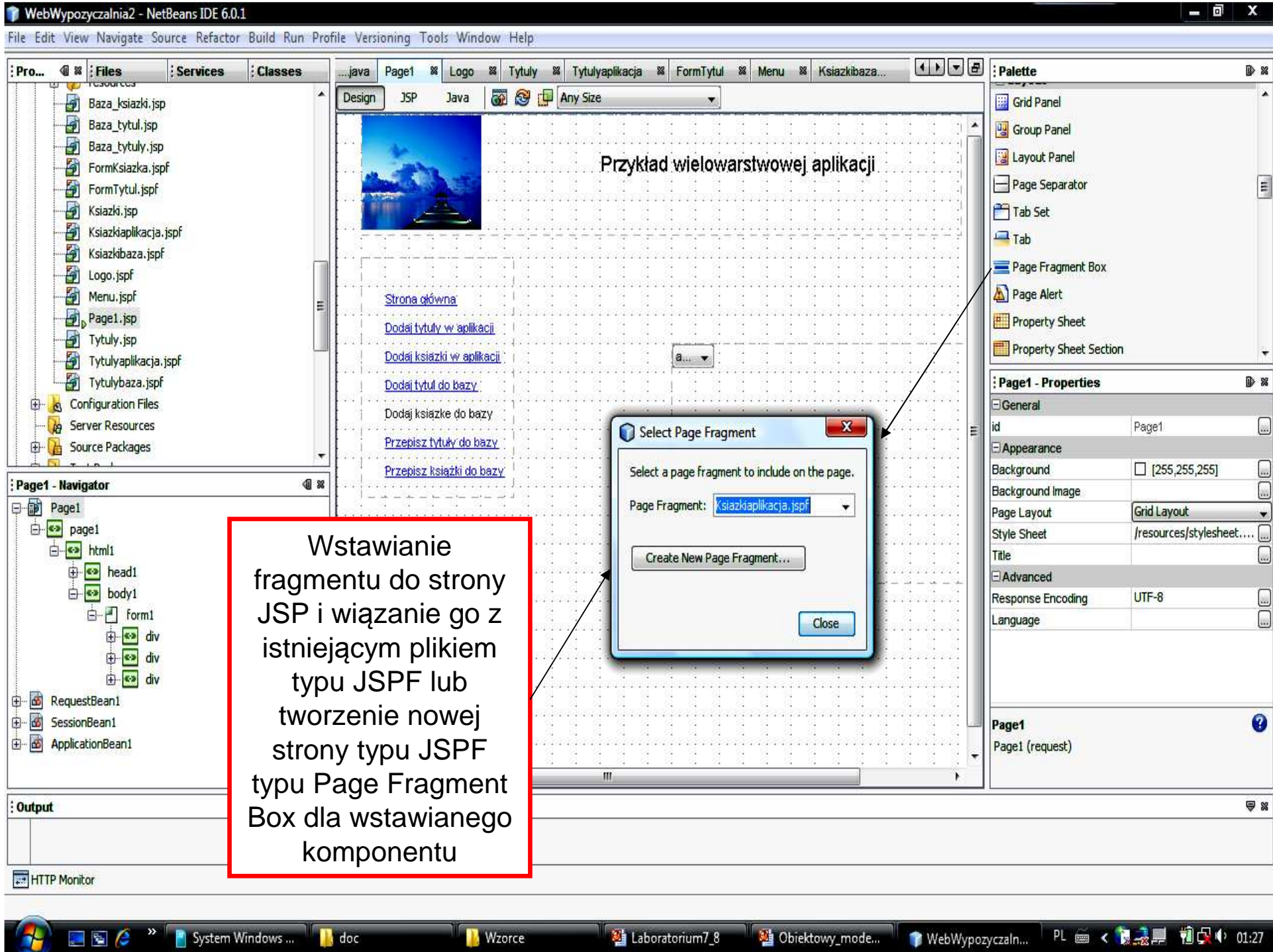
Norton™ Monitorowanie fałszywych witryn jest włączono... Opcje



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](#)

Internet | Tryb chroniony: włączony 100%



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

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

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)

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

The screenshot displays an IDE interface for a web application project named "WebWypożyczalnia2". The left sidebar shows a file explorer with the following structure:

- Web Pages
 - WEB-INF
 - resources
 - Baza_książki.jsp
 - Baza_tytul.jsp
 - Baza_tytuly.jsp
 - FormKsiążka.jspf
 - FormTytul.jspf
 - Książki.jsp
 - Książkiaplikacja.jspf
 - Książki baza.jspf
 - Logo.jspf
 - Menu.jspf
 - Page1.jsp
 - Tytuly.jsp
 - Tytulyaplikacja.jspf
 - Tytuly baza.jspf

The main design view shows a page titled "Przykład wielowarstwowej aplikacji" (Example of a multi-layered application). It features a navigation menu on the left with the following items:

- [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](#)~~

The central part of the page contains a form for adding a title:

Tytuł	<input type="text"/>
Autor	<input type="text"/>
ISBN	<input type="text"/>
Wydawnictwo	<input type="text"/>
Aktor	<input type="text"/>

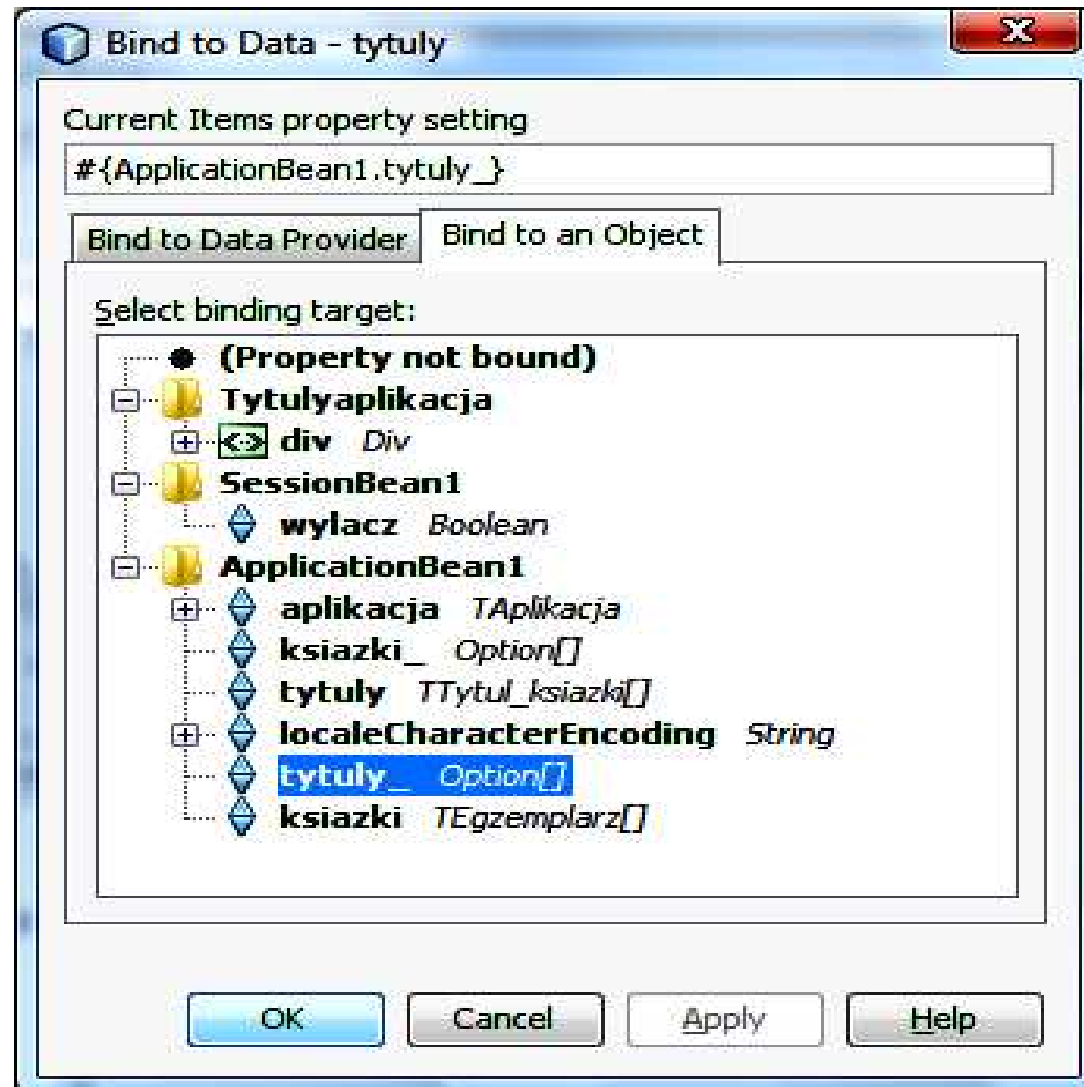
Below the form is a dropdown menu labeled "a..." and a "Dodaj tytuł" button. On the right side, there is a "System Messages" box containing a red message: "List of all message summaries".

The bottom-left pane, titled "Tytuly - Navigator", shows the component tree for the "Tytuly" page:

- Tytuly
 - page1
 - html1
 - head1
 - body1
 - form1
 - div
 - div
 - div
 - messageGroup1
 - div
 - dodajtytul:Dodaj tytuł

At the bottom of the IDE, there are two beans defined: RequestBean1 and SessionBean1.

Bindowanie tablicy **tytuly_** obiektów typu **Option**, zawierających dane o tytułach, pobrane z warstwy biznesowej



http://localhost:8080/WebWypożyczalnia2/faces/Tytuly.jsp - Windows Internet Explorer


http://localhost:8080/WebWyp

Wyszukaj

Norton™

Monitorowanie fałszywych witryn jest włączono...

Opcje



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	<input type="text"/>
Autor	<input type="text"/>
ISBN	<input type="text"/>
Wydawnictwo	<input type="text"/>
Aktor	<input type="text"/>

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

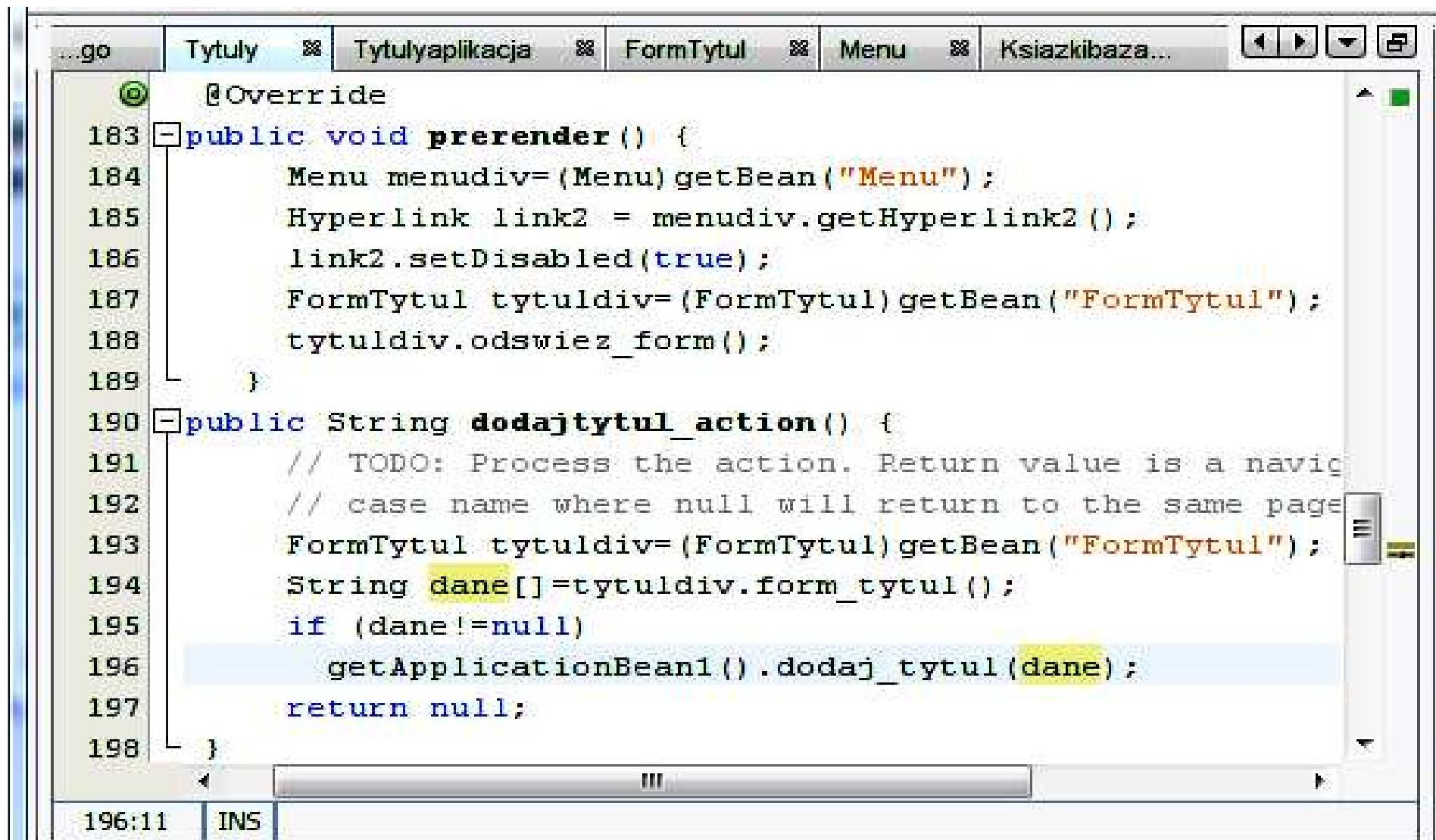
Tytuł: 6 Autor: 6 ISBN: 6 Wydawnictwo: 6

Internet | Tryb chroniony: włączony

100%

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 }
```

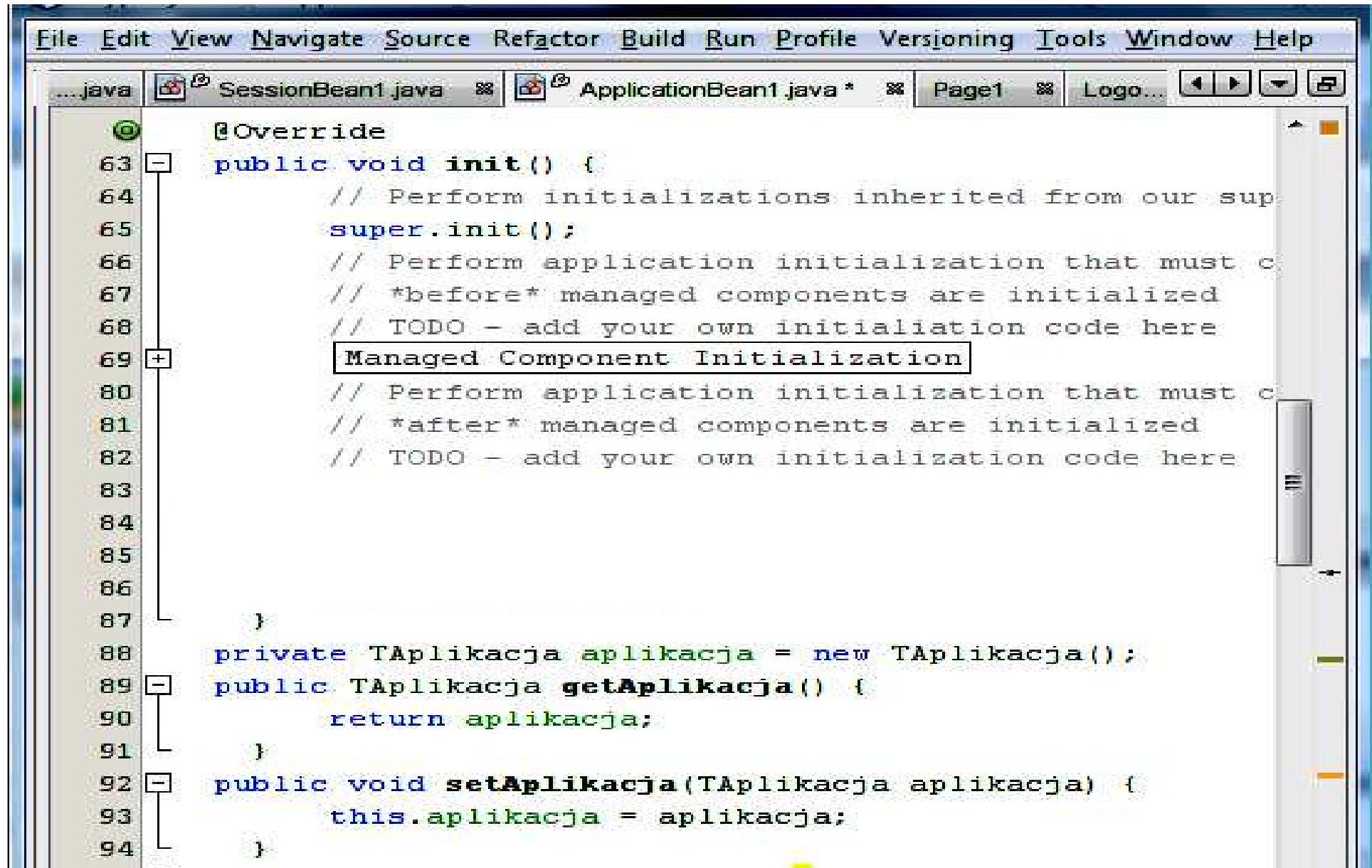
196:11 | INS

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  {
213      tytul.yaplikacjaPanel.setRendered(true);
214      tytul.setText("");
215      autor.setText("");
216      ISBN.setText("");
217      wydawnictwo.setText("");
218      aktor.setText("");
219  }
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  }
```

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_kasce* 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() {
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  }
```

Projekty formularza „Przepisz tytuły do bazy” (Baza_tytuly.jsp) – (9)

The screenshot displays an IDE interface for a web application project named "WebWypożyczalnia2". The main design view shows a page titled "Przykład wielowarstwowej aplikacji" with a grid background. The page content includes:

- A navigation menu with links: [Strona główna](#), [Dodaj tytuły w aplikacji](#), [Dodaj książki w aplikacji](#), [Dodaj tytuł do bazy](#), [Dodaj książki do bazy](#), [Przepisz tytuły do bazy](#), and [Przepisz książki do bazy](#).
- A button labeled "Zapisz tytuły do bazy".
- A table titled "Tytuły" with the following data:

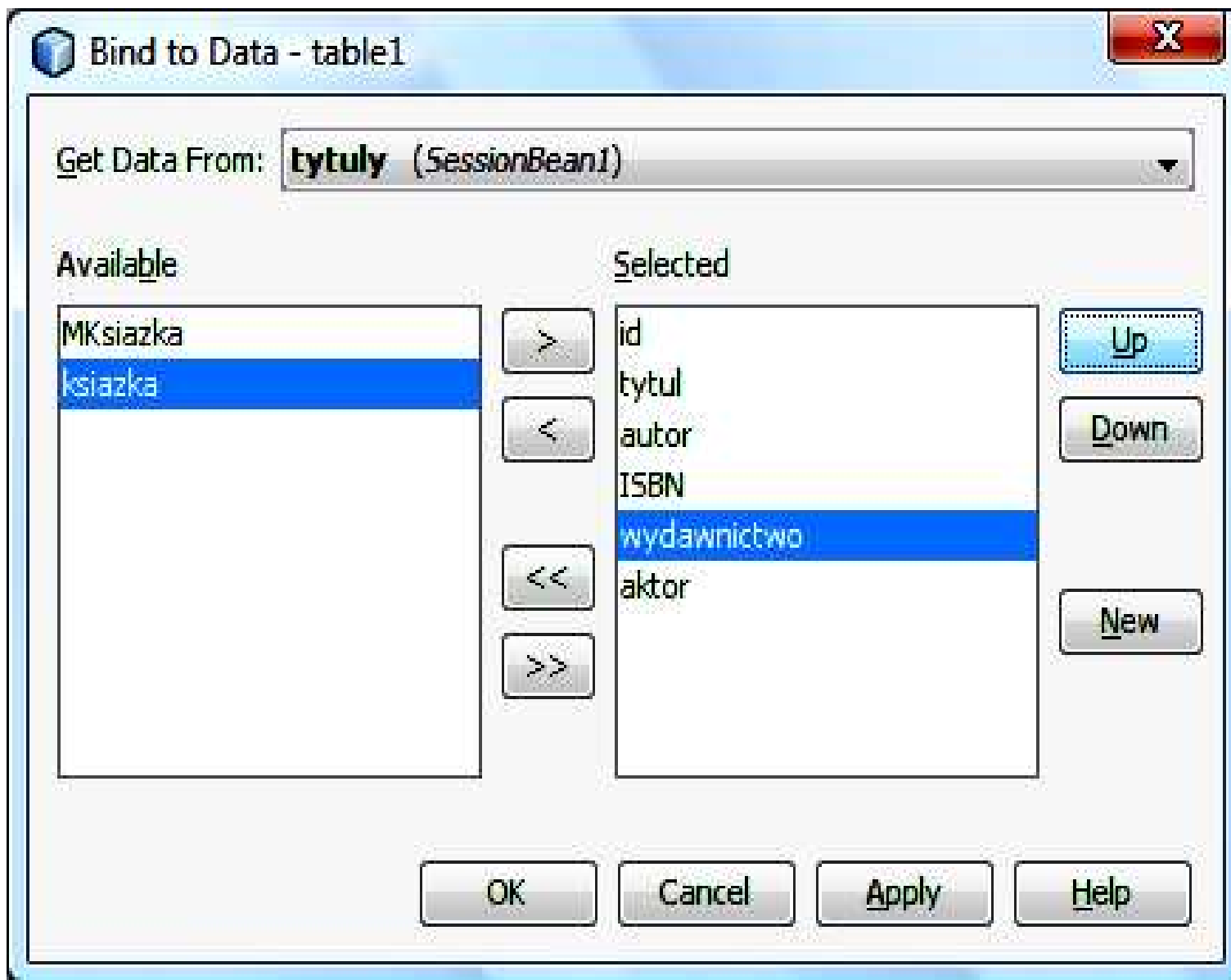
id	tytuł	autor	ISBN	wydawnictwo	aktor
123	abc	abc	abc	abc	abc
123	abc	abc	abc	abc	abc
123	abc	abc	abc	abc	abc

The "Baza_tytuly - Navigator" shows the page structure:

- Baza_tytuly
 - page1
 - html1
 - head1
 - body1
 - form1
 - div
 - div
 - div
 - jsp:directive.include:Tytylba
 - addTitleToBaz:Zapisz tytuły do b

The RequestBean1 section at the bottom shows a RequestBean1 object.


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



http://localhost:8080/WebWypożyczalnia2/faces/Baza_tytuly.jsp - Windows Internet Exp...

http://localhost:8080/WebWypo

Norton™ Monitorowanie fałszywych witryn jest włączone



Przykład wielowarstwowej aplikacji

Zapisz tytuły do bazy

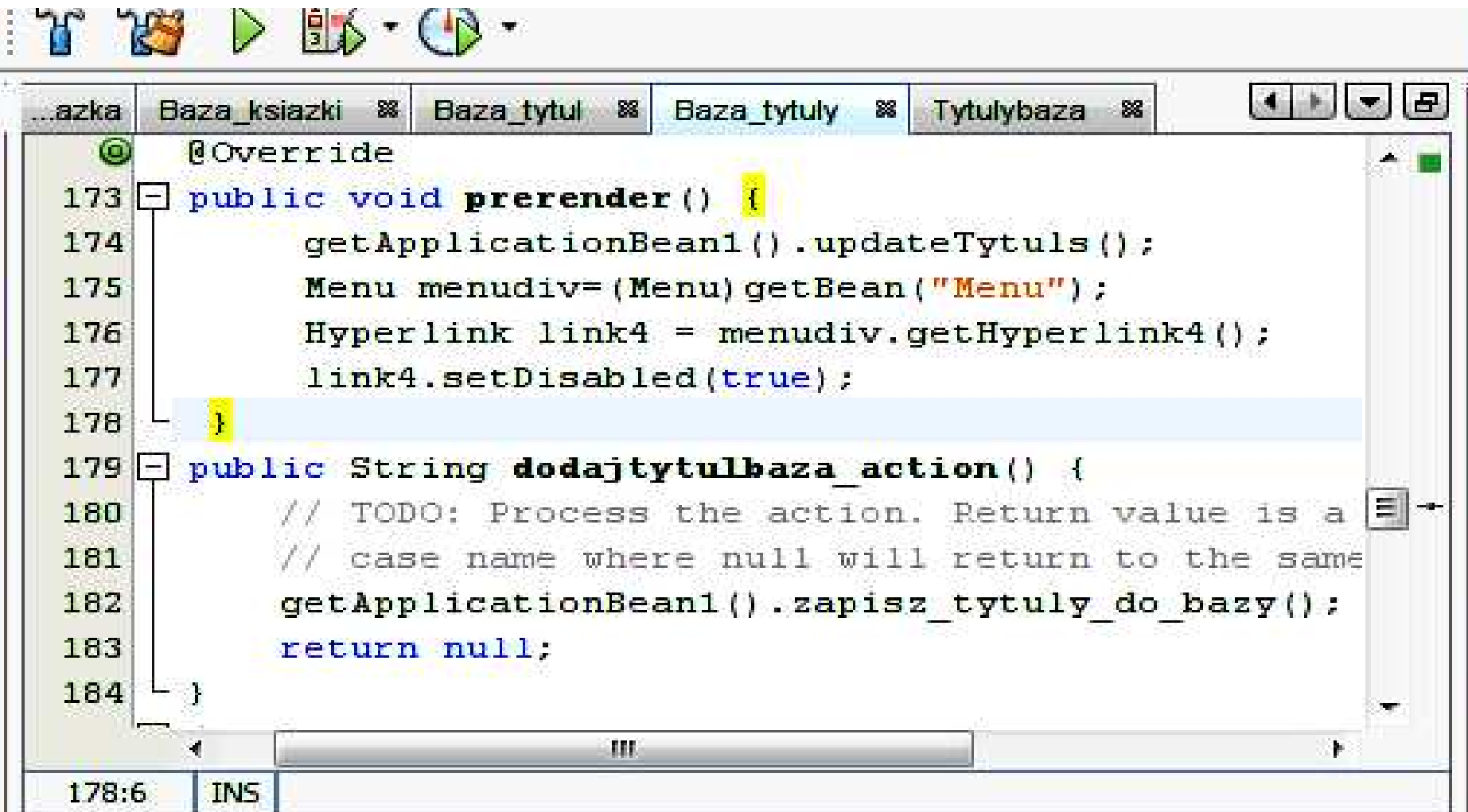
- Strona główna
- Dodaj tytuły w aplikacji
- Dodaj książki w aplikacji
- Dodaj tytuł do bazy
- Dodaj książkę do bazy
- Przepisz tytuły do bazy
- Przepisz książki do bazy

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	

Internet | Tryb chroniony: włączony 100%

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)



```
@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

```
...ava ApplicationBean1.java *  TTytul_książkiController.java  persistence.xml  SQL Command 1  SQL
private TTytul_książki tytuly[];
public TTytul_książki[] getTytuly() {
    return tytuly;
}
public void setTytuly(TTytul_książki[] tytuly) {
    this.tytuly = tytuly;
}
public void updateTytuls() {
    TTytul_książkiController tytulController = new TTytul_książkiController();
    tytuly = tytulController.getTTytul_książkis();
}
public void zapisz_tytuly_do_bazy() {
    // Add the new Entity to the database using UserController
    TTytul_książkiController Tytul_książkiController =
        new TTytul_książkiController();
    Tytul_książkiController.addTTytul_książki(getAplikacja().getTytul_książki());
}
```

Zsynchronizowanie zawartości bazy danych i warstwy biznesowej przy starcie aplikacji – wywołanie metod update... przy tworzeniu obiektu ApplicationBean1 (10)

```
@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 < tytulys.length; i++)
        aplikacja.getTytul_ksiazki().add(tytulys[i]);
    Iterator it = aplikacja.getTytul_ksiazki().iterator();
    while (it.hasNext()) {
        TTytul_ksiazki tytul = (TTytul_ksiazki) it.next();
        for (int j = 0; j < ksiazkis.length; j++) {
            TTytul_ksiazki tytul1 = ksiazkis[j].getMTytul_ksiazki();
            if (tytul1 != null)
                if (tytul1.equals(tytul))
                    { tytul.getMKsiazka().add(ksiazkis[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 (11)

```
...java TApplikacja.java * Tegzemplarz.java SQL Command 1 SQL Command 6 TTytul_książki.java TE...
9 package wyposzczalniaapp;
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 Wswietl_książki() {...}
82     public synchronized void Wswietl_tytuly() {...}
91     public static void main(String t[]) {...}
160 }
```


Wygenerowane tabele bazy danych podczas bindowania tablicy **tytuly** obiektów typu **Entity** z komponentem typu **DropDownList**

The screenshot shows a database management tool interface. On the left, a tree view displays the database schema for 'jdbc:derby://localhost:1527/katalog [kruk on KRUK]'. The 'Tables' folder is expanded, showing 'TEGZEMPLARZ' and 'TTYTUL_KSIAZKI'. The 'TEGZEMPLARZ' table has columns: ID, DTYPE, NUMER, MTYTUL_KSIAZKI_ID, and TERMIN. The 'TTYTUL_KSIAZKI' table has columns: ID, DTYPE, ISBN, TYTUL, WYDAWNICTWO, AUTOR, and AKTOR. A foreign key relationship is shown between 'MTYTUL_KSIAZKI_ID' in 'TEGZEMPLARZ' and 'TTYTUL_KSIAZKI.ID' in 'TTYTUL_KSIAZKI'.

On the right, a SQL Command window shows the query: `select * from KRUK.TEGZEMPLARZ`. Below the query, a data table is displayed with the following columns: ID, DTYPE, NUMER, MTYTUL_K..., and TERMIN. The table contains 5 rows of data.

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



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

```
SQL Command 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. The left pane displays a tree view of the database schema for 'jdbc:derby://localhost:1527/katalog [kruk on KRUK]'. The tree is expanded to show the 'Tables' folder, which contains several tables and their columns:

- SEQUENCE**: SEQ_NAME, SEQ_COUNT
- TEGZEMPLARZ**: ID, DTYPE, NUMER, MTYTUL_KSIAZKI_ID, TERMIN
- TTYTUL_KSIAZKI**: ID, DTYPE, ISBN, TYTUL, WYDAWNICTWO, AUTOR, AKTOR

The right pane shows a query window with the following SQL statement:

```
select * from
```

Below the query window, the results of the query are displayed in a table:

SEQ_NAME	SEQ_COUNT
SEQ_GEN	251

At the bottom of the interface, there is a status bar that reads: "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

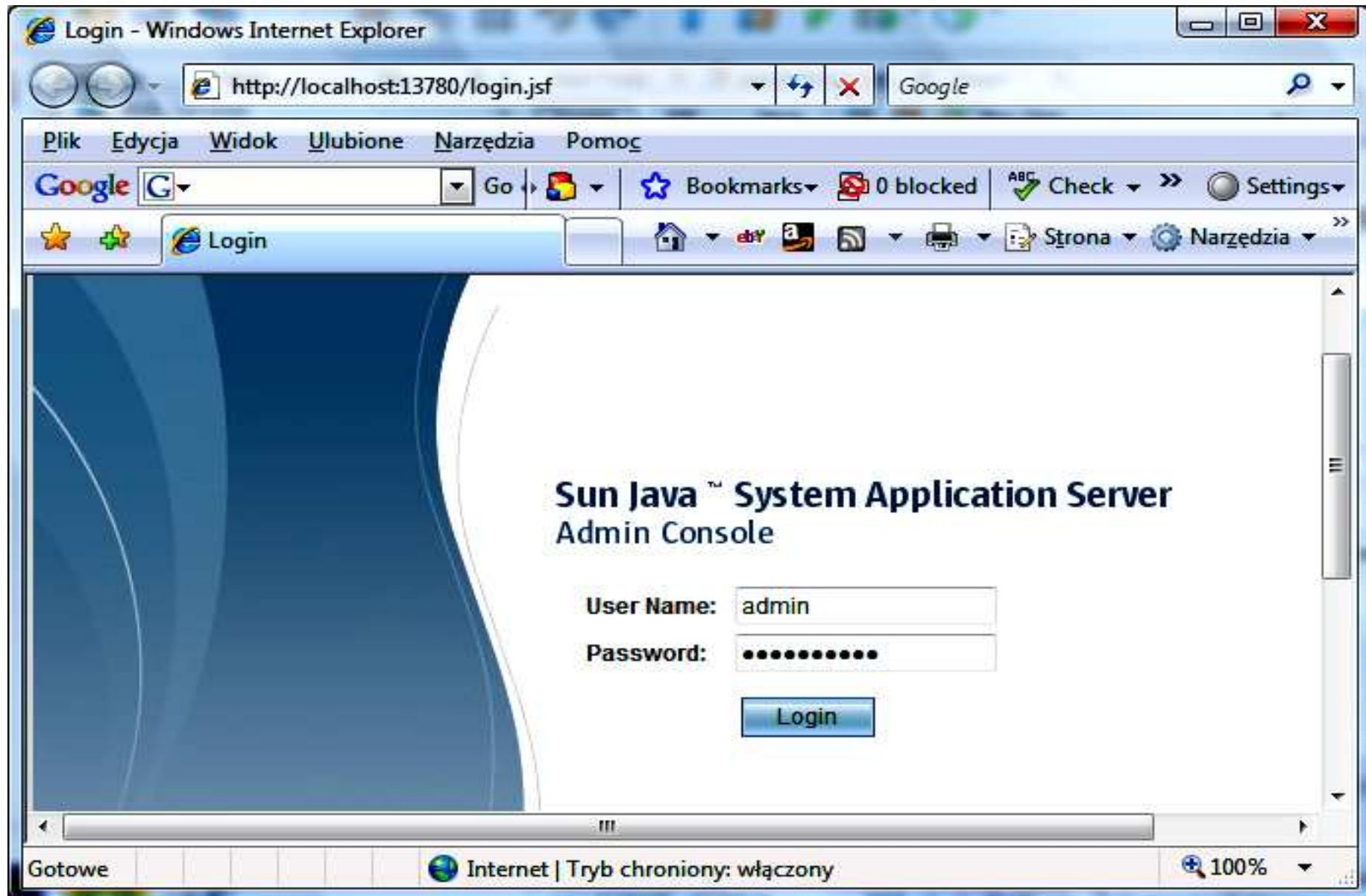
Name	ID
Type	TABLE
Nulls allowed	<input type="checkbox"/>
Data type	BIGINT
Default value	null
Column size	19
Decimal digits	0
Position	1
Notes	

ID Primary Key

Close

**10. 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 "Sun Java System Application Server 9.1_02 Admin...". The console header includes "Home", "Version", "Logout", and "Help" buttons, along with user information: "User: admin", "Domain: personalDomain", and "Server: localhost".

The main content area shows a navigation tree on the left with "Security" > "Realms" > "file" selected. The right pane displays the "New File Realm User" dialog box with the following fields:

- User ID ***: klient
- Group List**: klienci
- New Password ***: [masked]
- Confirm New Password ***: [masked]

Buttons for "OK" and "Cancel" are visible in the top right of the dialog. The status bar at the bottom shows "Gotowe", "Internet | Tryb chroniony: włączony", and "100%".

Sun Java System Application Server 9.1_02 Admin Console - Windows Internet Explorer

http://localhost:13780/?rs555659554=rs1459281478

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

Configuration > Security > Realms > file

File Users

Manage user accounts for the currently selected security realm.

Back

Users (2)

New... Delete

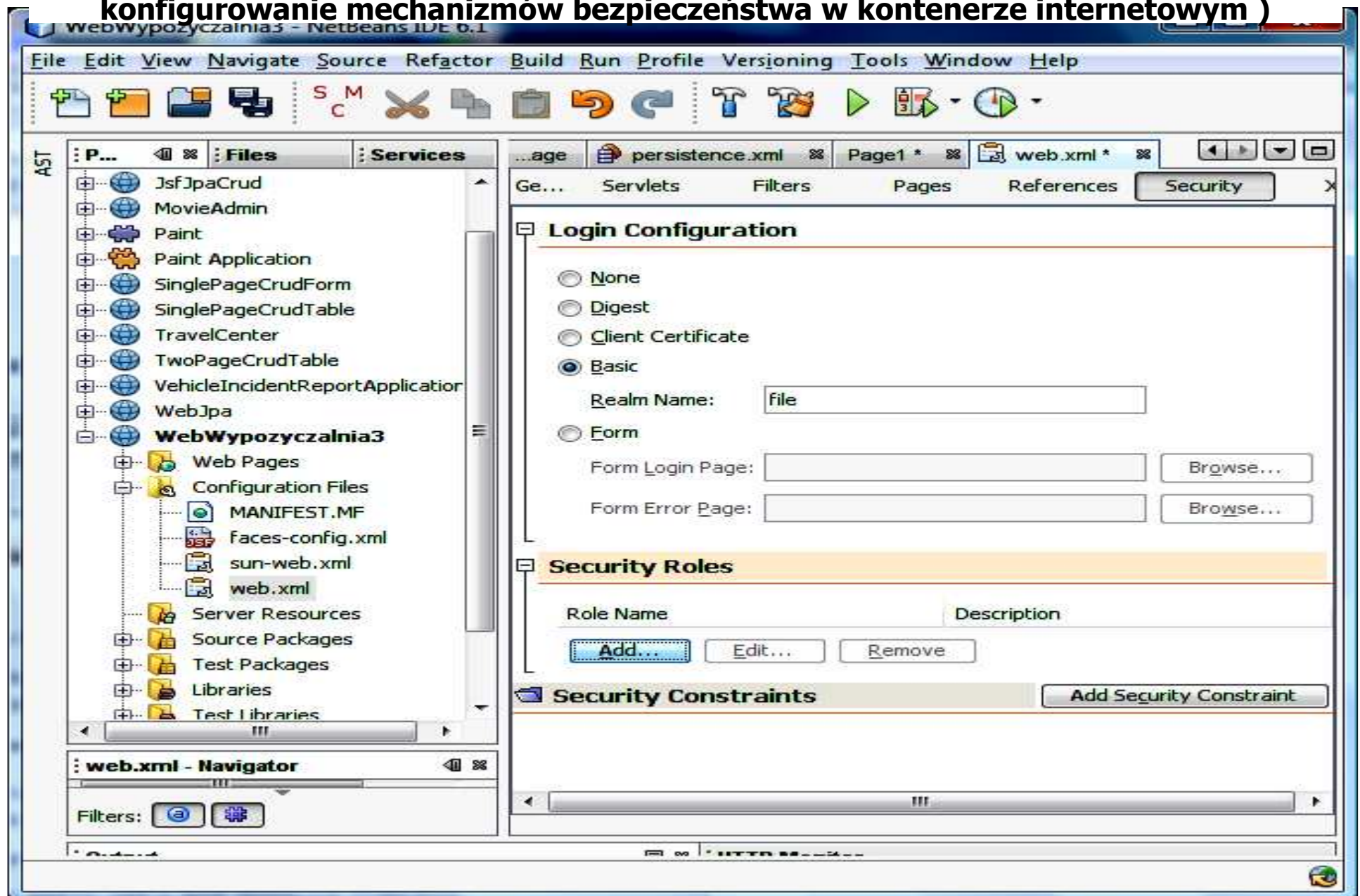
	User ID	Group List
<input type="checkbox"/>	klient	klienci
<input type="checkbox"/>	administrator	klienci

Service Assemblies

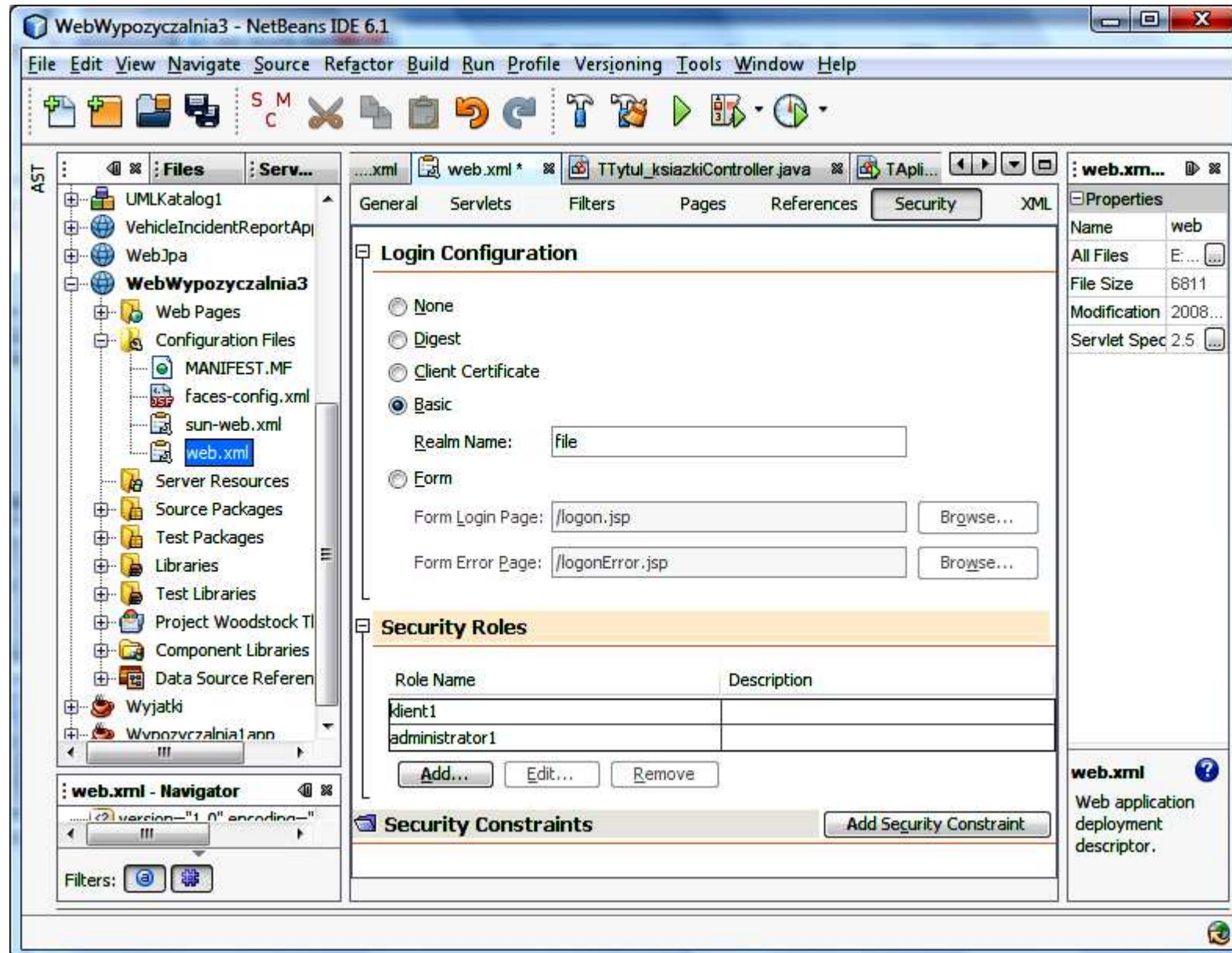
- Components
- Shared Libraries
- Custom MBeans
- Resources
- Configuration
 - Web Container
 - EJB Container
 - Java Message Servi
 - Security
 - Realms
 - admin-realm
 - file
 - certificate
 - JACC Providers
 - Audit Modules
 - Message Securi
 - Transaction Service

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*

The screenshot shows an IDE window with the 'Security' tab selected. The 'Security Roles' section contains a table with two roles: 'klient1' and 'administrator1'. The 'Security Constraints' section shows an 'AdministratorConstraint' with the following configuration:

AdministratorConstraint [Remove]

Display Name: AdministratorConstraint

Web Resource Collection:

Name	URL Pattern	HTTP Method	Description
Administrator	/faces/*	GET, POST, HEAD, PUT, OPTIONS, TRACE, DELETE	

[Add...] [Edit...] [Remove]

Enable Authentication Constraint

Description: [Text Field]

Role Name(s): administrator1 [Edit]

Enable User Data Constraint

Description: [Text Field]

Transport Guarantee: NONE [Dropdown]

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

...ava ShowCartBean.java LocaleBean.java BookDB.java BookDBAO.java ContextListener.java Method.java [r/o] bookcashier.jsp InsertTag.java web.xml

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:

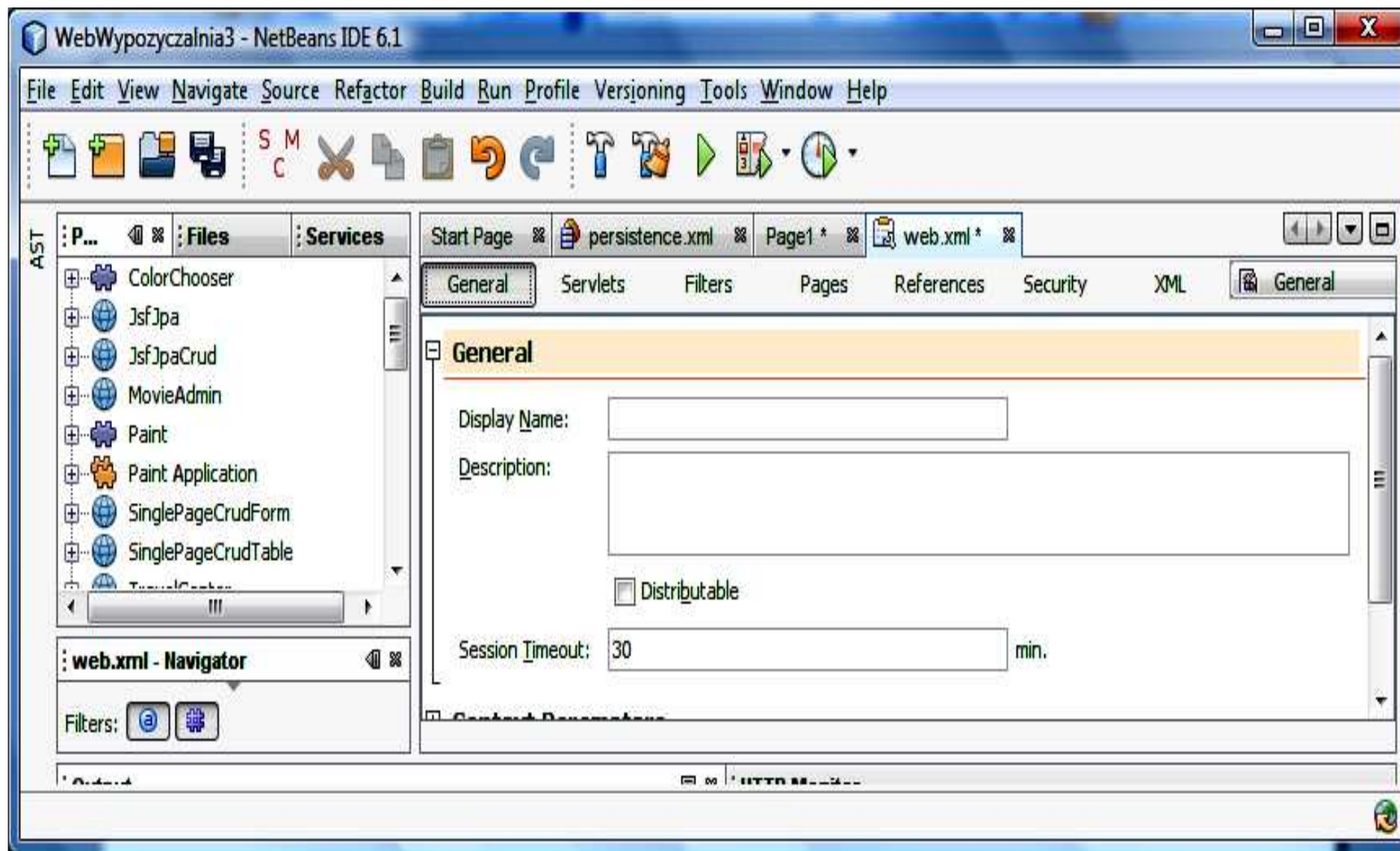
Role Name(s): klient1 Edit

Enable User Data Constraint

Description:

Transport Guarantee: NONE

(7) Ustawienie czasu sesji

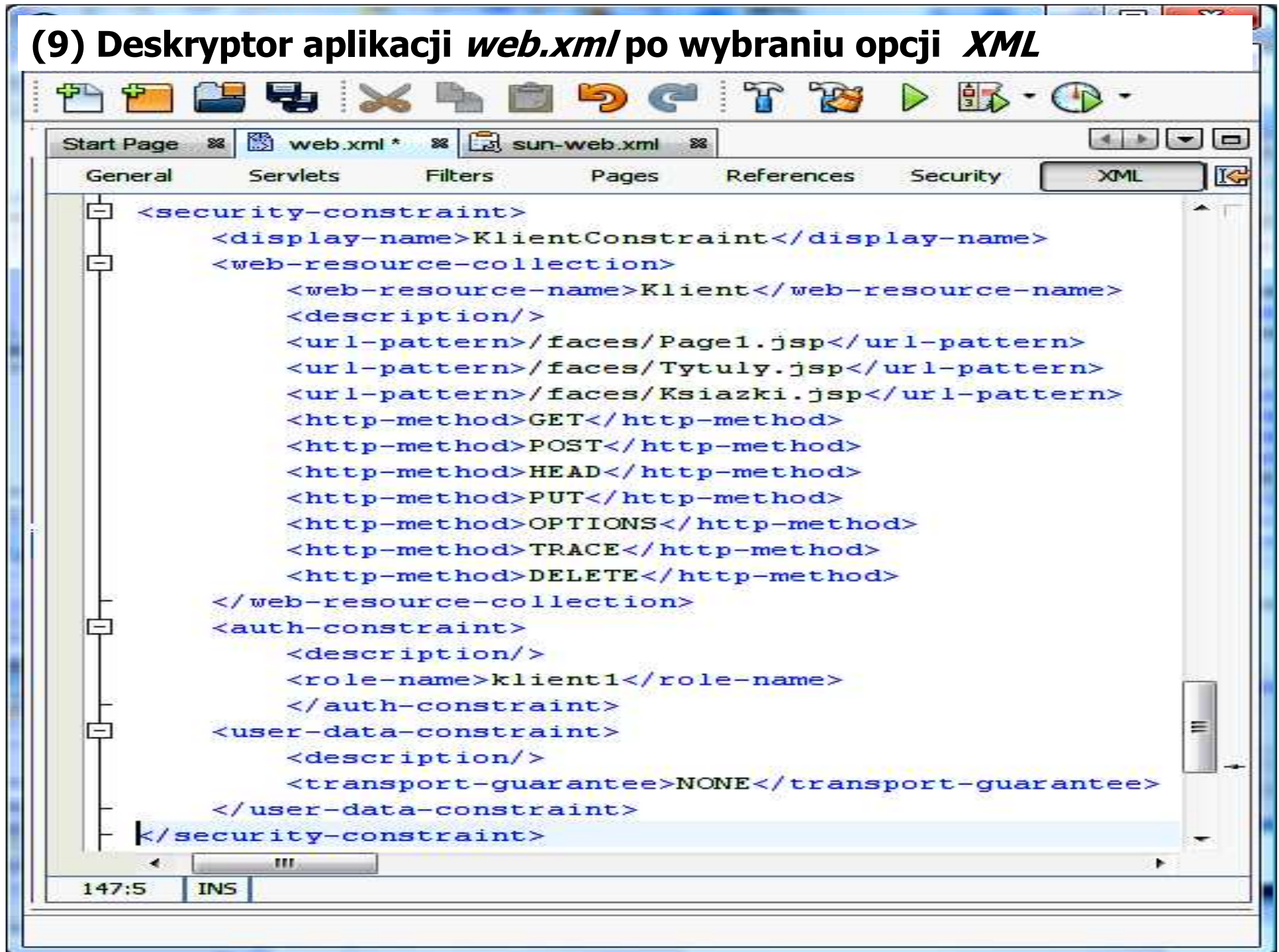


(8) Deskryptor aplikacji *web.xml* po wybraniu opcji *XML*

```
<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>
```

114:9 | INS

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

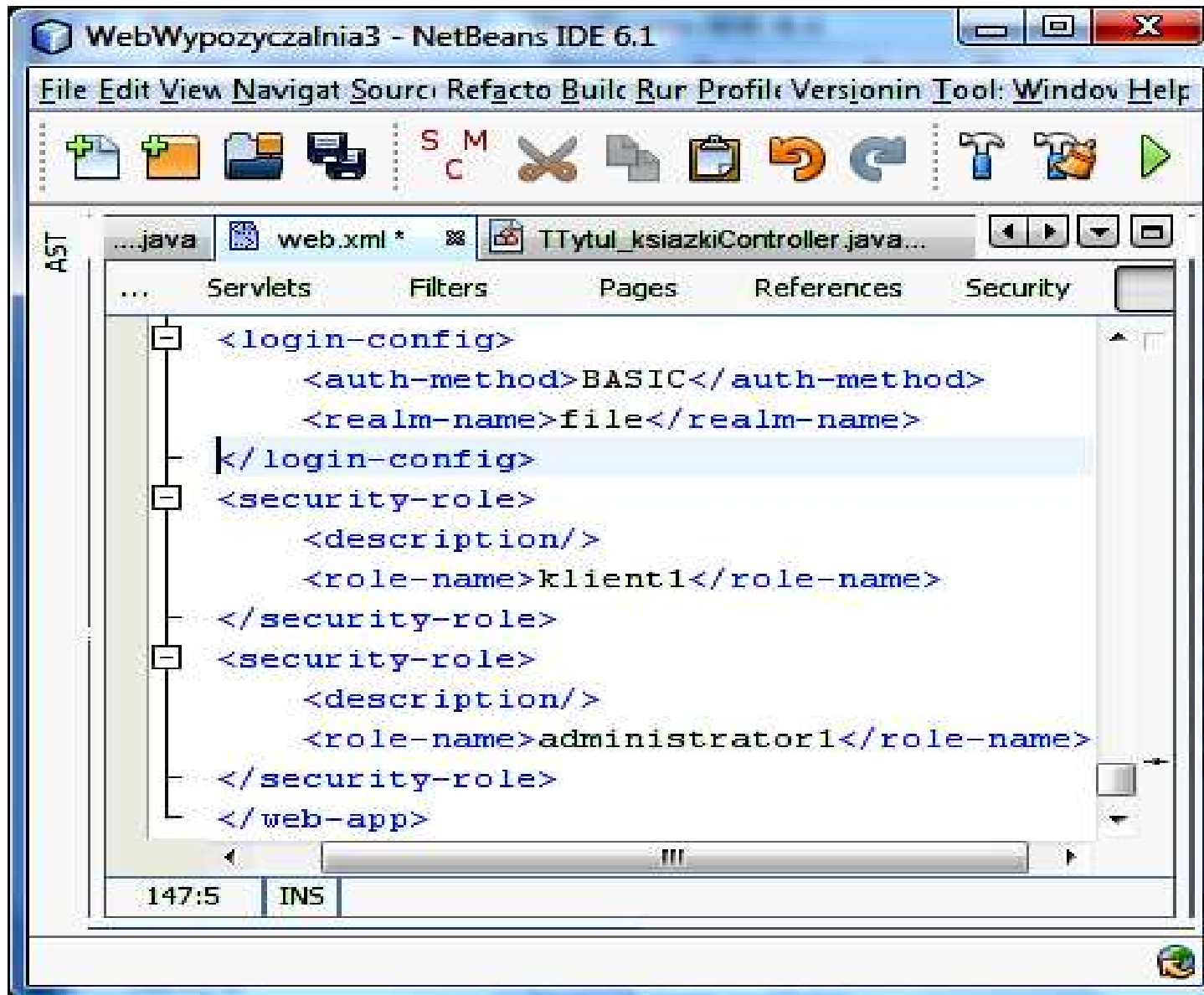


The screenshot shows an IDE window with the following tabs: Start Page, web.xml *, and sun-web.xml *. The XML view is active, displaying the following code:

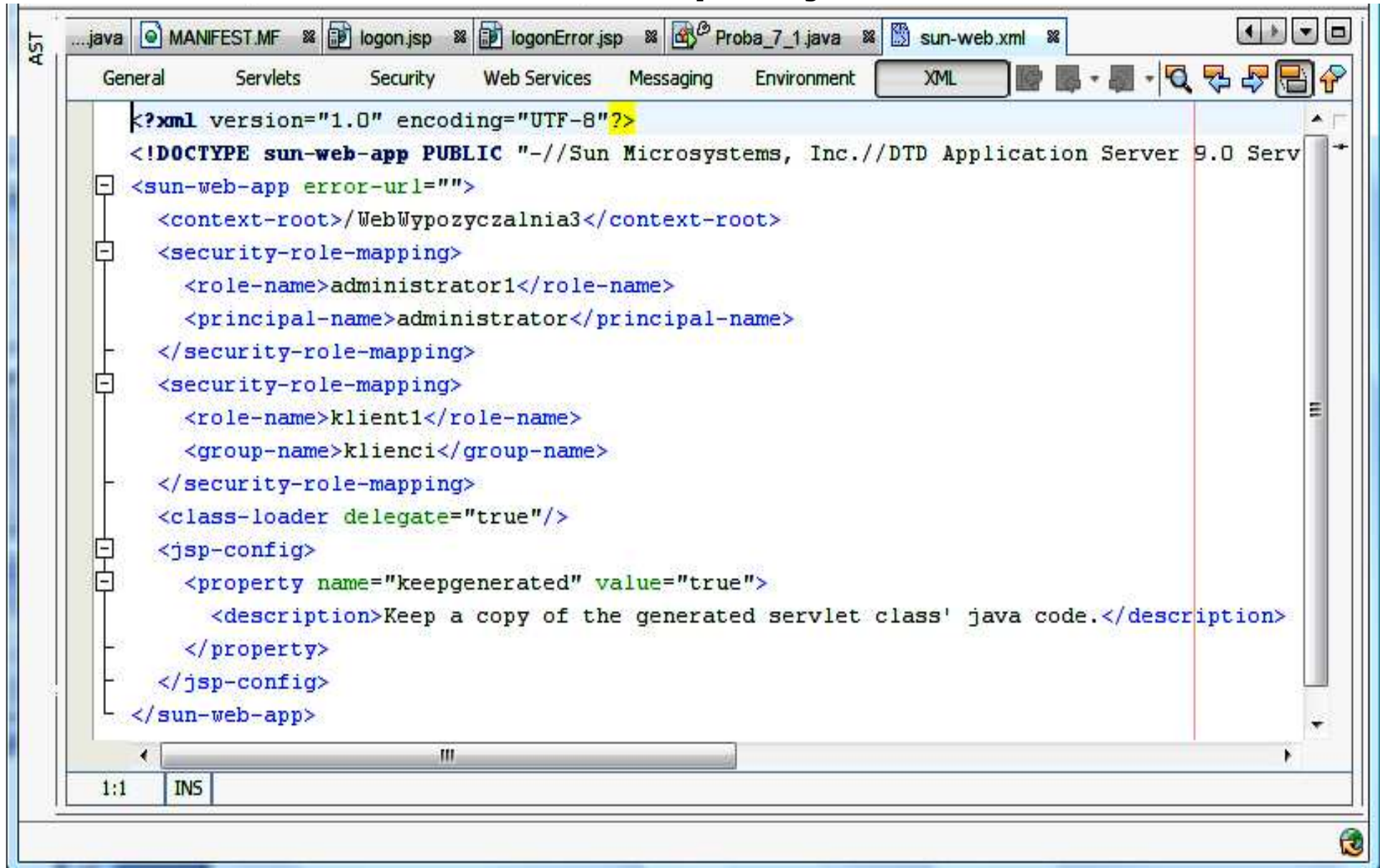
```
<security-constraint>
  <display-name>KlientConstraint</display-name>
  <web-resource-collection>
    <web-resource-name>Klient</web-resource-name>
    <description/>
    <url-pattern>/faces/Page1.jsp</url-pattern>
    <url-pattern>/faces/Tytuly.jsp</url-pattern>
    <url-pattern>/faces/Ksiazki.jsp</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>
```

The status bar at the bottom indicates the cursor is at line 147, column 5, with the text "INS" displayed.

(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 screenshot shows an IDE window with the file 'sun-web.xml' open. The XML content 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>/WebWypożyczalnia3</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>
```

The IDE interface includes a toolbar with 'XML' selected, a left-hand tree view, and a status bar at the bottom showing '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: administrator

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: klient

Hasło:

Zapamiętaj moje hasło

OK Anuluj

http://localhost:8081/WebWypożyczalnia3/ - Windows Internet Explorer

http://localhost:8081/WebWypożyczalnia3/ Google

Google Szukaj Zaloguj się

http://localhost:8081/Web... Strona Narzędzia



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


Gotow

http://localhost:8081/WebWypożyczalnia3/faces/Książki.jsp - Windows Internet Explorer

http://localhost:8081/WebWypożyczalnia3/faces/Książki.jsp Google

Google Szukaj Zaloguj się

http://localhost:8081/WebWypożyczalnia3... Strona Narzędzia



Przykład wielowarstwowej aplikacji

Numer

Termin

Tytuł: 1 Autor: 1 ISBN: 1 Wydawnictwo: 1

Tytuł: 1 Autor: 1 ISBN: 1 Wydawnictwo: 1

Tytuł: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2

Tytuł: 3 Autor: 3 ISBN: 3 Wydawnictwo: 3 Aktor: 3

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

Dodaj książkę

Gotowe

Internet | Tryb chroniony: wyłączony 100%

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

