

**Zasady generowania kluczy
głównych
Język „Java Persistence”
Podstawowa architektura
wielowarstwowych aplikacji w
oparciu o wzorce
oprogramowania**

Autor

Zofia Kruczkiewicz

Wzorce oprogramowania 6

1. Różne mechanizmy generowania kluczy głównych w procesie mapowania modelu obiektowego na relacyjny

Zasady generowania kluczy głównych (1)

@Id oznacza adnotację dla klucza głównego.

Adnotacja **@GeneratedValue (strategy=GenerationType.<...>)** oznacza strategię tworzenia wartości kluczy głównych, zależną od systemu baz danych

(`javax.persistence.GeneratedValue`, `javax.persistence.GenerationType`).

1. **Domyślna strategia** pozwala na przejęcie odpowiedzialności za generowanie kluczy głównych przez TopLink, który posługuje się pomocniczą tabelą przechowującą wartości potrzebne do generowania kluczy – **przykład 1**

@Entity

```
public class Inventory implements Serializable {  
    @Id  
    @GeneratedValue(strategy=GenerationType.AUTO)  
    private long id;
```

2. **Wykorzystanie mechanizmu identyczności kolumny do generowania wartości klucza głównego** oznacza odpowiedzialność za generowanie wartości kluczy głównych przez bazę danych – **przykład 2**

@Entity

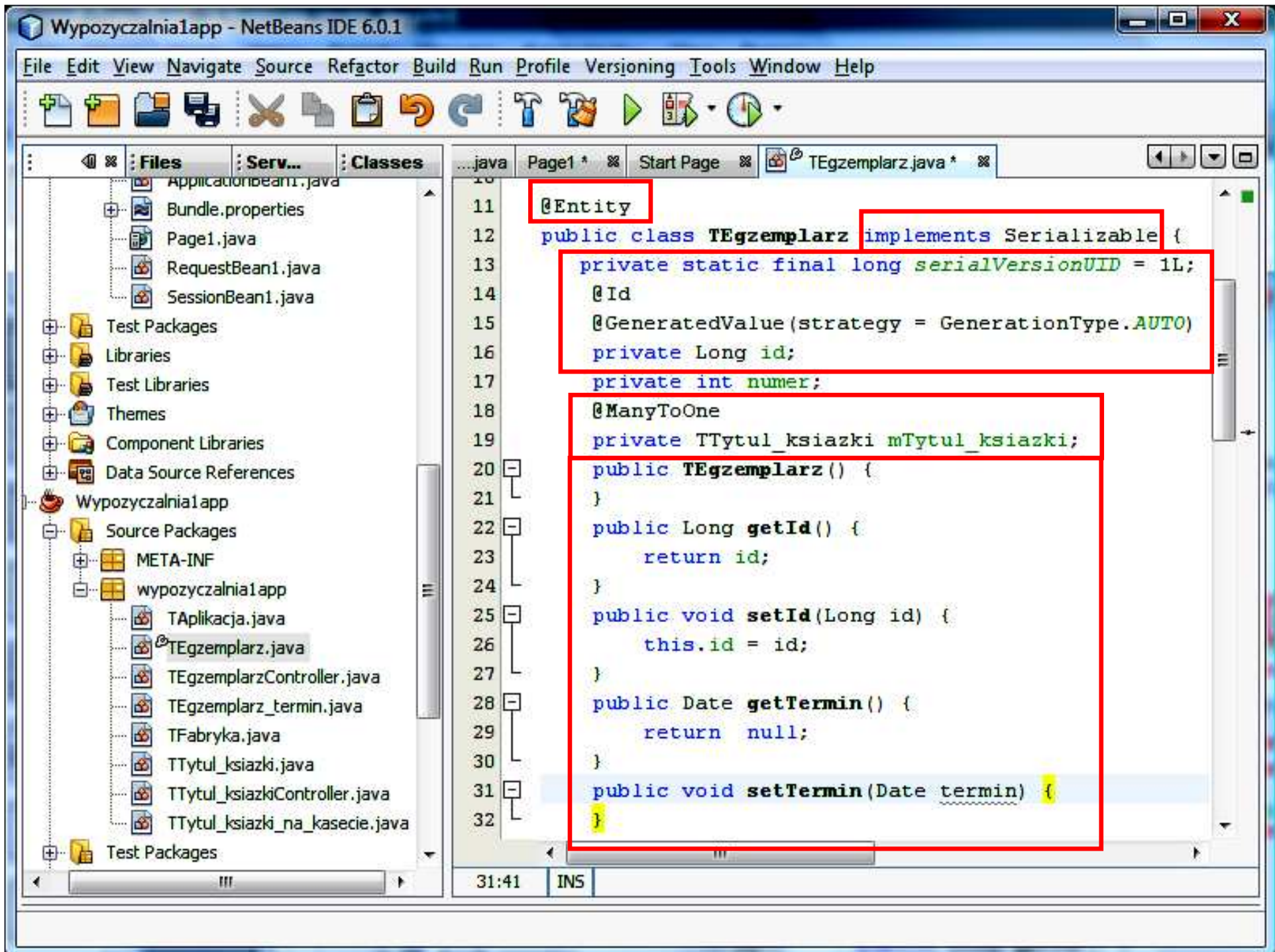
```
public class Inventory implements Serializable {  
    @Id  
    @GeneratedValue(strategy=GenerationType.IDENTITY)  
    private long id;
```

Przykład 1 - @GeneratedValue(strategy=GenerationType.AUTO)

The screenshot shows the NetBeans IDE 6.0.1 interface. The left pane displays the project structure for 'Wypożyczalnia1app'. The main editor window shows the source code of 'TTYtul_książki.java'. The code is as follows:

```
13 @Entity
14 public class TTYtul_książki implements Serializable {
15     private static final long serialVersionUID = 1L;
16     @Id
17     @GeneratedValue(strategy = GenerationType.AUTO)
18     private Long id;
19     private String wydawnictwo;
20     private String ISBN;
21     private String tytul;
22     private String autor;
23
24     @OneToMany(mappedBy = "mTytul_książki")
25     private Collection<TEgzemplarz> Książka;
26     @Transient
27     private ArrayList<TEgzemplarz> mKsiążka =
28         new java.util.ArrayList<TEgzemplarz>();
29     public ArrayList<TEgzemplarz> getMKsiążka() {
30         return mKsiążka;
31     }
32     public void setMKsiążka(ArrayList<TEgzemplarz> mKsiążka)
33     {
34         this.mKsiążka = mKsiążka;
35     }
36     public Long getId() {
37         return id;
38     }
39     public void setId(Long id) {
40         this.id = id;
41     }
42     public Collection<TEgzemplarz> getKsiążka() {
43         return Książka;
44     }
45     public void setKsiążka(Collection<TEgzemplarz> Książka) {
46         this.Książka = Książka;
47     }
48 }
```

Annotations and class name are highlighted with red boxes. A large red box highlights the entire class body. The status bar at the bottom shows '27:13 INS'.





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 14
1 select * from KRUK.TEGZEMPLARZ
```

1:1 | INS

ID	DTYPE	NUMER	MTYTUL_K...	TERMIN
3	TEgzemplarz_termin		1	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 a database schema for 'jdbc:derby://localhost:1527/katalog [kruk on KRUK]'. The tree is expanded to show the 'SEQUENCE' table, which contains columns 'SEQ_NAME' and 'SEQ_COUNT'. Below it are other tables: 'TEGZEMPLARZ' and 'TTYTUL_KSIAZKI'. The 'TTYTUL_KSIAZKI' table has columns 'ID', 'DTYPE', 'ISBN', 'TYTUL', 'WYDAWNICTWO', 'AUTOR', and 'AKTOR'. A foreign key relationship is shown between 'TTYTUL_KSIAZKI' and 'TEGZEMPLARZ' on the 'MTYTUL_KSIAZKI_ID' column.

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

```
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

The screenshot shows the NetBeans IDE 6.0.1 interface. The 'Projects' pane on the left displays a database schema for 'jdbc:derby://localhost:1527/katalog [kruk on KRUK]'. The 'Tables' folder is expanded, showing a table named 'TEGZEMPLARZ'. The 'ID' column is selected, and the 'ID - Properties' dialog is open. The dialog shows the following properties:

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	

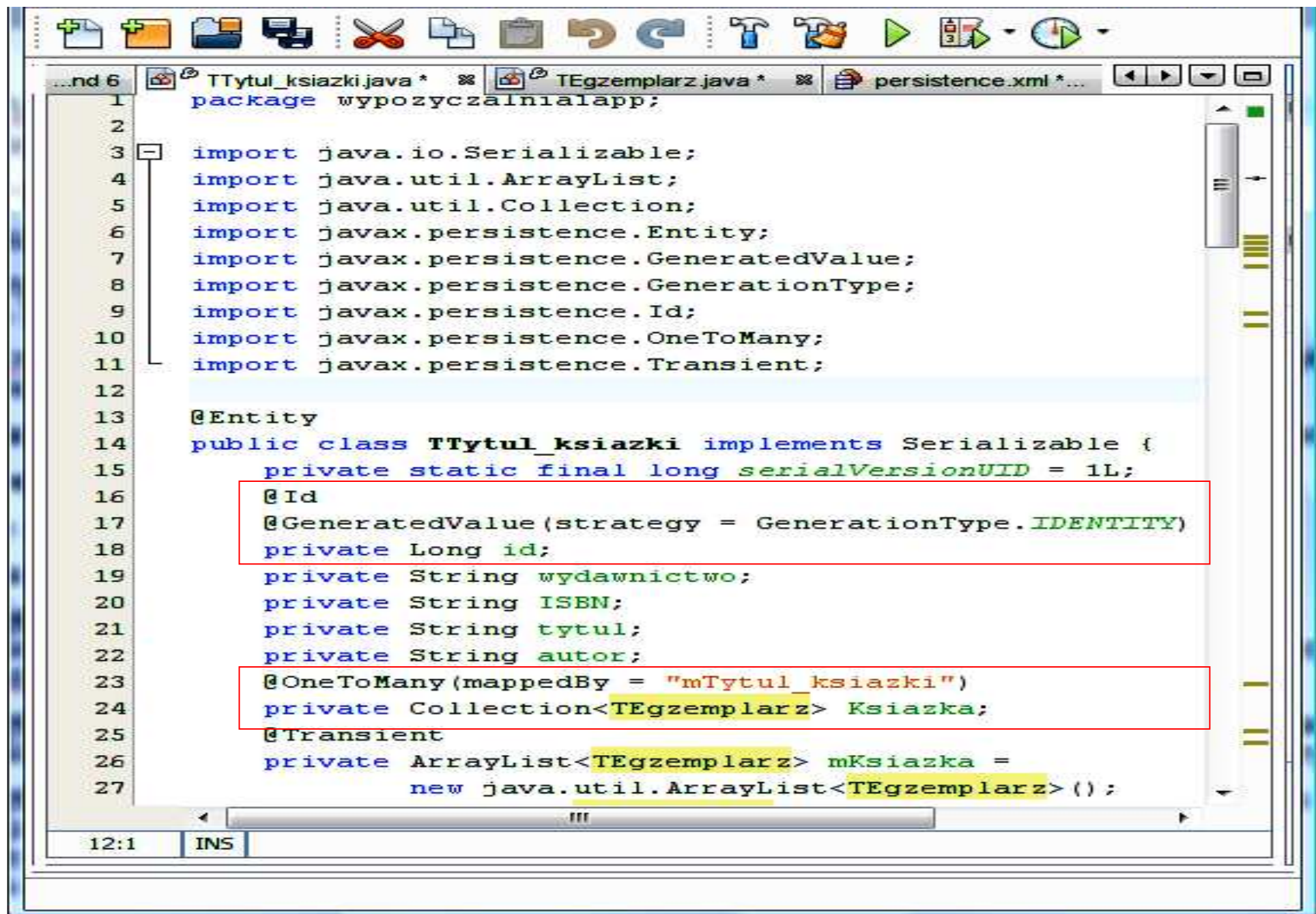
Below the properties table, the column is identified as a **Primary Key**. A 'Close' button is visible at the bottom of the dialog.

The screenshot shows the NetBeans IDE 6.0.1 interface. The 'Projects' pane on the left displays a database schema for 'jdbc:derby://localhost:1527/katalog [kruk on KRUK]'. The 'Tables' folder is expanded, showing a table named 'TTYTUL_KSIAZKI'. The 'ID' column is selected, and the 'ID - Properties' dialog is open. The dialog shows the following properties:

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	

Below the properties table, the column is identified as a **Primary Key**. A 'Close' button is visible at the bottom of the dialog.

Przykład 2 - @GeneratedValue(strategy=GenerationType.IDENTITY)



```
...nd 6  TTytul_książki.java *  TEGzemplarz.java *  persistence.xml *...
1      package wypożyczalniaapp;
2
3      import java.io.Serializable;
4      import java.util.ArrayList;
5      import java.util.Collection;
6      import javax.persistence.Entity;
7      import javax.persistence.GeneratedValue;
8      import javax.persistence.GenerationType;
9      import javax.persistence.Id;
10     import javax.persistence.OneToMany;
11     import javax.persistence.Transient;
12
13     @Entity
14     public class TTytul_książki implements Serializable {
15         private static final long serialVersionUID = 1L;
16         @Id
17         @GeneratedValue(strategy = GenerationType.IDENTITY)
18         private Long id;
19         private String wydawnictwo;
20         private String ISBN;
21         private String tytul;
22         private String autor;
23         @OneToMany(mappedBy = "mTytul_książki")
24         private Collection<TEgzemplarz> Książka;
25         @Transient
26         private ArrayList<TEgzemplarz> mKsiążka =
27             new java.util.ArrayList<TEgzemplarz> ();
```

12:1 | INS



...java * TEgzemplarz.java * persistence.xml * SQL Command 9...

```
1 package wypożyczalnia1app;
2
3 import java.io.Serializable;
4 import java.util.Date;
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9 import javax.persistence.ManyToOne;
10
11 @Entity
12 public class TEgzemplarz implements Serializable {
13     private static final long serialVersionUID = 1L;
14     @Id
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private Long id;
17     private int numer;
18     @ManyToOne
19     private TTytul_ksiazki mTytul_ksiazki;
```

15:55

INS



Projects Files Services Classes

jdbc:derby://localhost:1527/katalog1 [kruk1 on KRUK1]

- Tables
 - 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

...java SQL Command 12

```
1 select * from KRUK1.TEGZEMPLARZ
```

1:1 INS

ID	DTYPE	NUMER	MTYTUL_K...	TERMIN
1	TEgzemplarz		1	NULL
2	TEgzemplarz_termin		1	2008-05-06
3	TEgzemplarz		2	NULL



Projects Files Ser... Classes

jdbc:derby://localhost:1527/katalog1 [kruk1 on KRUK1]

- Tables
 - 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

...ge1 TEGzemplarzController.java SQL Command 12 SQL Command 13

```
1 select * from KRUK1.TTYTUL_KSIAZKI
```

1:1 INS

ID	DTYPE	ISBN	TYTUL	WYDAWNI...	AUTOR	AKTOR
1	TTYtul_ksiazki	1	1	1	1	NULL
2	TTYtul_ksiazki_na_kasecie	2	2	2	2	2

Właściwości kluczy głównych przy zastosowanej strategii IDENTITY

The image displays two screenshots of the NetBeans IDE 6.0.1 interface, illustrating the configuration of a primary key using the IDENTITY strategy in a database. Both screenshots show the 'ID - Properties' dialog box for a primary key.

Left Screenshot: The 'ID - Properties' dialog is open for the 'ID' column in the 'TEGZEMPLARZ' table. The properties are:

Property	Value
Name	ID
Type	TABLE
Nulls allowed	<input type="checkbox"/>
Data type	BIGINT
Default value	AUTOINCREMENT: start 1 increment 1
Column size	19
Decimal digits	0
Position	1
Notes	Primary Key

Right Screenshot: The 'ID - Properties' dialog is open for the 'ID' column in the 'TTYTUL_KSIAZKI' table. The properties are:

Property	Value
Name	ID
Type	TABLE
Nulls allowed	<input type="checkbox"/>
Data type	BIGINT
Default value	AUTOINCREMENT: start 1 increment 1
Column size	19
Decimal digits	0
Position	1
Notes	Primary Key

Zasady generowania kluczy głównych (2)

3. Specyfikowanie metody generowania za pomocą tabeli (3.1 - za pomocą domyślnej tabeli TopLink lub 3.2 – za pomocą tabeli zdefiniowanej przez programistę = [przykład 3 \(wg przykładu z Java EE 5 Tutorial \)](#))

3.1. @Entity
`public class Inventory implements Serializable {
 @Id
 @GeneratedValue(strategy=GenerationType.TABLE)
 private long id;`

3.2. @Entity
`public class Inventory implements Serializable {
 @Id
 @GeneratedValue(generator="InvTab")
 @TableGenerator(name="InvTab", table="ID_GEN",
 pkColumnName="ID_NAME", valueColumnName="ID_VAL",
 pkColumnValue="INV_GEN")
 private long id;`

```
np  
ID_GEN  
ID_NAME  
INV_GEN  
ID_VAL  
<last generated value >  
// Nazwy kolumn  
// wartości kolejnej krotki
```

4. Wykorzystanie sekwencyjnego mechanizmu nadawania wartości kluczy głównych wspieranych przez bazy danych (4.1 - za pomocą domyślnego mechanizmu lub 4.2 – za pomocą generatora zdefiniowanego przez programistę)

4.1. @Entity
`public class Inventory implements Serializable {
 @Id
 @GeneratedValue(strategy=GenerationType.SEQUENCE)
 private long id;`

4.2. @Entity
`public class Inventory implements Serializable {
 @Id
 @GeneratedValue(generator="InvSeq")
 @SequenceGenerator(name="InvSeq",
 sequenceName="INV_SEQ", allocationSize=5)
 private long id;`

Przykład 3 – @GeneratedValue(strategy = GenerationType.TABLE, generator = "accountIdGen") - wg Java EE 5 Tutorial
 Adnotacja dla relacji *Many To Many* pomiędzy encjami *Account* i *Customer* (@ManyToMany). Encja *Account* jest mapowana w bazie danych za pomocą tabeli *BANK_ACCOUNT* (adnotacja @Table)

The screenshot shows an IDE with the following components:

- Project Explorer (Left):** Shows a project structure with packages like 'Source Packages', 'Test Packages', and 'Libraries'. The 'Account.java' file is selected under 'bank1.entity'.
- Code Editor (Center):** Displays the source code for 'Account.java'. The code includes:
 - Annotations: @Entity, @Table(name = "BANK_ACCOUNT"), @NamedQueries (with several queries), @TableGenerator, @Id, @GeneratedValue, @Column, @JoinTable, @ManyToMany, @JoinColumn, @OneToMany, @Column, @Temporal, @Column, @Column, @Column.
 - Class Declaration: public class Account implements java.io.Serializable {
 - Attributes: private Long id; private Collection<Customer> customers; private Collection<Tx> tx; private Date beginBalanceTimeStamp; private BigDecimal beginBalance; private BigDecimal creditLine; private BigDecimal balance; private String description; private String type;
- Account.java - Navigator (Bottom Left):** Lists the attributes of the Account class: beginBalance: BigDecimal, beginBalanceTimeStamp: Date, creditLine: BigDecimal, customers: Collection<Customer>, description: String, id: Long, tx: Collection<Tx>, type: String.
- Annotations and Relationships (Red Boxes):**
 - Top Box:** Encloses the @TableGenerator, @Id, @GeneratedValue, and @Column annotations. An arrow points from this box to the @JoinTable annotation.
 - Middle Box:** Encloses the @JoinTable annotation. An arrow points from this box to the @ManyToMany annotation.
 - Bottom Box:** Encloses the @JoinColumn, @OneToMany, and @Column annotations. An arrow points from this box to the @ManyToMany annotation.
- Explanatory Text Boxes (Right):**
 - Top Box:** "Relacja Many To Many między encjami Account i Customer mapowana w bazie danych za pomocą tablicy asocjacyjnej BANK_CUSTOMER_ACCOUNT_XREF zadeklarowanej adnotacją @JoinTable"
 - Bottom Box:** "Relacja One To Many między encjami Account i Tx mapowana w bazie danych za pomocą @JoinColumn (kolumna Account_Id w tabeli BANK_TX w bazie danych i w encji Tx atrybut account) i @OneToMany – w encji Account kolekcja tx"

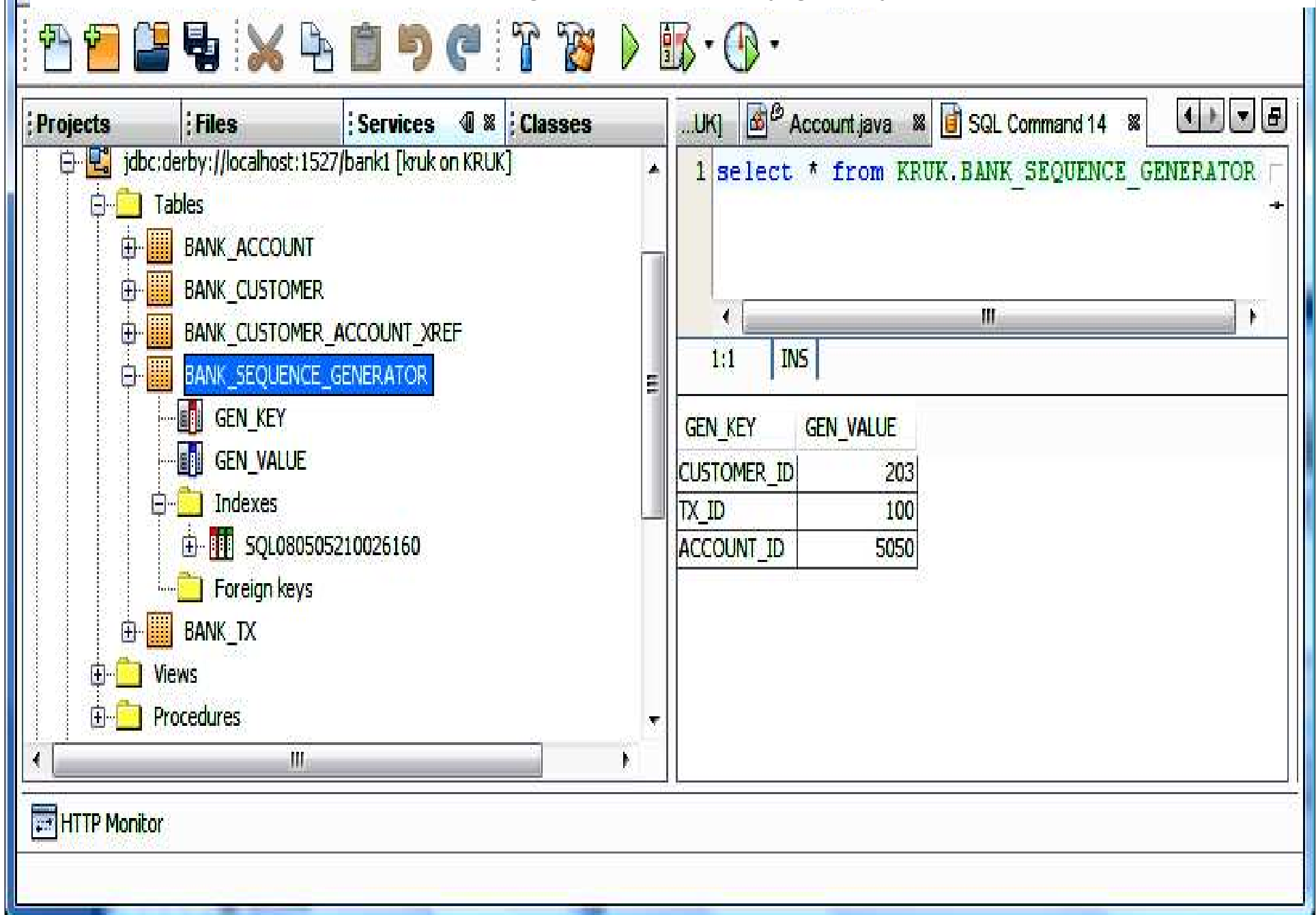
Encja *Customer* jest mapowana w bazie danych za pomocą tabeli *BANK_CUSTOMER* (adnotacja *@Table*). Adnotacja *@ManyToMany* dla relacji *Many To Many* zawiera jedynie z jednej strony relacji atrybut *mappedBy* deklarujący pole relacji *customers* w encji *Account*

```
44 @Entity
45 @Table(name = "BANK_CUSTOMER")
46 @NamedQueries({@NamedQuery(name = "Customer.findById", query = "SELECT a FROM Customer a WHERE a.id = :id")
47 ,@NamedQuery(name = "Customer.findByLastName", query = "SELECT a FROM Customer a WHERE a.lastName LIKE :lastName")
48 ,@NamedQuery(name = "Customer.findByFirstName", query = "SELECT a FROM Customer a WHERE a.firstName = :firstName")
49 ,@NamedQuery(name = "Customer.findByMiddleInitial", query = "SELECT a FROM Customer a WHERE a.middleInitial = :middleInitial")
50 ,@NamedQuery(name = "Customer.findByStreet", query = "SELECT a FROM Customer a WHERE a.street = :street")
51 ,@NamedQuery(name = "Customer.findByCity", query = "SELECT a FROM Customer a WHERE a.city = :city")
52 ,@NamedQuery(name = "Customer.findByState", query = "SELECT a FROM Customer a WHERE a.state = :state")
53 ,@NamedQuery(name = "Customer.findByZip", query = "SELECT a FROM Customer a WHERE a.zip = :zip")
54 ,@NamedQuery(name = "Customer.findByPhone", query = "SELECT a FROM Customer a WHERE a.phone = :phone")
55 ,@NamedQuery(name = "Customer.findByEmail", query = "SELECT a FROM Customer a WHERE a.email = :email")
56 ,@NamedQuery(name="Customer.FindAllCustomersOfAccount",query="SELECT c FROM Customer c JOIN c.accounts a WHERE a.id = :accountId")})
57 public class Customer implements java.io.Serializable {
58     @ManyToMany(mappedBy = "customers")
59     private Collection<Account> accounts;
60     @TableGenerator(name = "customerIdGen", table = "BANK_SEQUENCE_GENERATOR", pkColumnName = "GEN_KEY",
61         valueColumnName = "GEN_VALUE", pkColumnValue = "CUSTOMER_ID", initialValue = 203, allocationSize = 1)
62     @Id
63     @GeneratedValue(strategy = GenerationType.TABLE, generator = "customerIdGen")
64     @Column(name = "CUSTOMER_ID", nullable = false)
65     private Long id;
66     @Column(name = "FIRST_NAME")
67     private String firstName;
68     @Column(name = "LAST_NAME")
69     private String lastName;
70     @Column(name = "MIDDLE_INITIAL")
71     private String middleInitial;
72     private String city;
73     private String email;
74     private String phone;
75     private String state;
76     private String street;
```

```
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
...java persistence.xml Flight.java Hotel.java Trip.java persistence.xml Tx.java * AccountDe...
49 @Entity
50 @Table(name = "BANK_TX")
51 @NamedQueries({
52     @NamedQuery(name = "Tx.FindById", query = "SELECT a FROM Tx a WHERE a.id = :id")
53     ,@NamedQuery(name = "Tx.FindByTimeStamp",
54         query = "SELECT a FROM Tx a WHERE a.timeStamp = :timeStamp")
55     ,@NamedQuery(name = "Tx.FindByAmount", query = "SELECT a FROM Tx a WHERE a.amount = :amount")
56     ,@NamedQuery(name = "Tx.FindByBalance", query = "SELECT a FROM Tx a WHERE a.balance = :balance")
57     ,@NamedQuery(name = "Tx.FindByDescription",
58         query = "SELECT a FROM Tx a WHERE a.description = :description")
59     ,@NamedQuery(name = "Tx.FindTxByDates",
60         query = "SELECT t FROM Tx t JOIN t.account a WHERE a.id = :accountId "
61             + "AND t.timeStamp BETWEEN :startDate AND :endDate"))}
62 public class Tx implements java.io.Serializable {
63     @TableGenerator(name = "txIdGen", table = "BANK_SEQUENCE_GENERATOR", pkColumnName = "GEN_KEY",
64         valueColumnName = "GEN_VALUE", pkColumnValue = "TX_ID",
65         initialValue = 100, allocationSize = 10)
66     @Id
67     @GeneratedValue(strategy = GenerationType.TABLE, generator = "txIdGen")
68     @Column(name = "TX_ID", nullable = false)
69     private Long id;
70     @JoinColumn(name = "ACCOUNT_ID")
71     @ManyToOne(cascade = CascadeType.ALL)
72     private Account account;
73     @Column(name = "TIME_STAMP")
74     @Temporal(TemporalType.TIMESTAMP)
75     private Date timeStamp;
76     private BigDecimal amount;
77     private BigDecimal balance;
78     private String description;
```

Encja Tx jest mapowana w bazie danych za pomocą tabeli BANK_TX (adnotacja @Table). Adnotacja @ManyToOne dla relacji Many to One między encjami Tx i Account nie może zawierać atrybutu mappedBy. Adnotacja @JoinColumn służy do mapowania odwzorowania tej relacji w bazie danych w postaci klucza obcego Account_ID w tabeli BANK_TX (pełni tę samą rolę, co atrybut mappedBy po stronie relacji np. One To Many w adnotacji @OneToMany, czyli atrybut account w encji Tx)

Tabela generatora kluczy głównych



The screenshot displays a database management tool interface. The left pane shows a tree view of a database schema for 'jdbc:derby://localhost:1527/bank1 [kruk on KRUK]'. The 'Tables' folder is expanded, and 'BANK_SEQUENCE_GENERATOR' is selected. Below it, 'GEN_KEY' and 'GEN_VALUE' are listed. The right pane shows an SQL query: `1 select * from KRUK.BANK_SEQUENCE_GENERATOR`. Below the query, the results are displayed in a table:

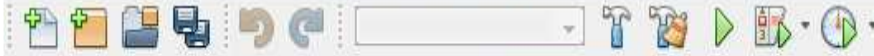
GEN_KEY	GEN_VALUE
CUSTOMER_ID	203
TX_ID	100
ACCOUNT_ID	5050

The bottom of the interface shows an 'HTTP Monitor' tab.

Model relacyjny dla modelu obiektowego

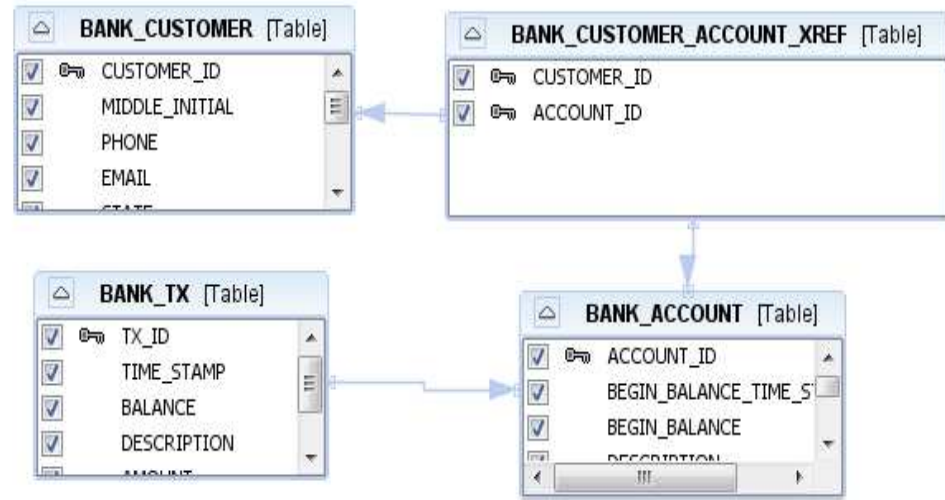
The screenshot displays the NetBeans IDE interface for a database project. The left sidebar shows a project tree with folders for Tables, Indexes, Foreign keys, and other database objects. The main workspace shows a diagram of three tables: BANK_TX, BANK_CUSTOMER_ACCOUNT_XREF, and BANK_ACCOUNT. BANK_TX is linked to BANK_ACCOUNT via its ACCOUNT_ID column. BANK_CUSTOMER_ACCOUNT_XREF is linked to both BANK_TX and BANK_ACCOUNT via their respective ACCOUNT_ID columns. BANK_CUSTOMER_ACCOUNT_XREF is also linked to BANK_CUSTOMER via its CUSTOMER_ID column. Below the diagram is a table with columns: Column, Alias, Table, Output, Sort Type, Sort Order, Criteria, and Order. The first row shows TX_ID from KRUK.BANK_TX. Below this is a SQL query editor containing the following query:

```
SELECT ALL
  KRUK.BANK_TX.TX_ID, KRUK.BANK_TX.AMOUNT, KRUK.BANK_TX.BALANCE, KRUK.BANK_TX.TIME_STAMP,
  KRUK.BANK_TX.DESCRPTION, KRUK.BANK_TX.ACCOUNT_ID,
  KRUK.BANK_ACCOUNT.ACCOUNT_ID, KRUK.BANK_ACCOUNT.BALANCE, KRUK.BANK_ACCOUNT.DESCRPTION,
  KRUK.BANK_ACCOUNT.BEGIN_BALANCE, KRUK.BANK_ACCOUNT.TYPE,
  KRUK.BANK_ACCOUNT.BEGIN_BALANCE_TIME_STAMP, KRUK.BANK_ACCOUNT.CREDIT_LINE,
  KRUK.BANK_CUSTOMER.CUSTOMER_ID, KRUK.BANK_CUSTOMER.EMAIL, KRUK.BANK_CUSTOMER.PHONE,
  KRUK.BANK_CUSTOMER.LAST_NAME, KRUK.BANK_CUSTOMER.STATE, KRUK.BANK_CUSTOMER.CITY,
  KRUK.BANK_CUSTOMER.STREET, KRUK.BANK_CUSTOMER.MIDDLE_INITIAL, KRUK.BANK_CUSTOMER.ZIP,
  KRUK.BANK_CUSTOMER.FIRST_NAME, KRUK.BANK_CUSTOMER_ACCOUNT_XREF.CUSTOMER_ID,
  KRUK.BANK_CUSTOMER_ACCOUNT_XREF.ACCOUNT_ID
FROM KRUK.BANK_TX
INNER JOIN KRUK.BANK_ACCOUNT ON KRUK.BANK_TX.ACCOUNT_ID = KRUK.BANK_ACCOUNT.ACCOUNT_ID, KRUK.BANK_CUSTOMER
INNER JOIN KRUK.BANK_CUSTOMER_ACCOUNT_XREF ON KRUK.BANK_CUSTOMER_ACCOUNT_XREF.CUSTOMER_ID =
  KRUK.BANK_CUSTOMER.CUSTOMER_ID
WHERE KRUK.BANK_CUSTOMER_ACCOUNT_XREF.ACCOUNT_ID = KRUK.BANK_ACCOUNT.ACCOUNT_ID
```



Projects: KRUK

- Files
 - Tables
 - BANK_ACCOUNT
 - BANK_CUSTOMER
 - BANK_CUSTOMER_ACCOUNT_XREF
 - CUSTOMER_ID
 - ACCOUNT_ID
 - Indexes
 - SQL080505210024210
 - SQL080505210025860
 - SQL080505210026020
 - Foreign Keys
 - BKNCSTMRCCNTCSTMRD
 - CUSTOMER_ID -> BANK_CUSTOMER.CUSTOMER_ID
 - BKNCSTMRCCNTXCNTD
 - ACCOUNT_ID -> BANK_ACCOUNT.ACCOUNT_ID
 - BANK_SEQUENCE_GENERATOR
 - BANK_TX
 - TX_ID
 - TIME_STAMP
 - BALANCE
 - DESCRIPTION
 - AMOUNT
 - ACCOUNT_ID
 - Indexes
 - SQL080505210025450
 - SQL080505210026130
 - Foreign Keys
 - BANK_TX_ACCOUNT_ID
 - ACCOUNT_ID -> BANK_ACCOUNT.ACCOUNT_ID
 - Views
 - Procedures



Column	Alias	Table	Output	Sort Type	Sort Order	Criteria
CUSTOMER_ID		KRUK.BANK_CUSTOMER	<input checked="" type="checkbox"/>			
MIDDLE_INITIAL		KRUK.BANK_CUSTOMER	<input checked="" type="checkbox"/>			
PHONE		KRUK.BANK_CUSTOMER	<input type="checkbox"/>			

```

SELECT ALL KRUK.BANK_CUSTOMER.CUSTOMER_ID, KRUK.BANK_CUSTOMER.MIDDLE_INITIAL, KRUK.BANK_CUSTOMER.ACCOUNT_ID
FROM KRUK.BANK_CUSTOMER, KRUK.BANK_ACCOUNT
INNER JOIN KRUK.BANK_CUSTOMER_ACCOUNT_XREF
ON KRUK.BANK_CUSTOMER_ACCOUNT_XREF.ACCOUNT_ID = KRUK.BANK_ACCOUNT.ACCOUNT_ID,
KRUK.BANK_TX
WHERE KRUK.BANK_CUSTOMER_ACCOUNT_XREF.CUSTOMER_ID = KRUK.BANK_CUSTOMER.CUSTOMER_ID
AND KRUK.BANK_TX.ACCOUNT_ID = KRUK.BANK_ACCOUNT.ACCOUNT_ID
  
```

CUSTOMER...	MIDDLE_I...	PHONE	EMAIL	STATE	LAST_NAME	STREET	FIRST_NAME	ZIP	CIT
-------------	-------------	-------	-------	-------	-----------	--------	------------	-----	-----

2. Wprowadzenie do języka zapytań „Java Persistence”

wg The Java EE 5 Tutorial

Terminologia

- ***Schemat abstrakcyjny (Abstract schema)***: Abstrakcyjny schemat utrwalania (**utrwalane encje, ich stan i powiązania**), na którym operują zapytania. Język zapytań tłumaczy zapytania nad abstrakcyjnym schematem utrwalania na zapytania wykonywane na schemacie bazy danych, do której odwzorowano encje.
- ***Typ abstrakcyjnego schematu utrwalania***: Wszystkie wyrażenia wyznaczają wynik należący do pewnego typu utrwalanych encji. Typ abstrakcyjnego schematu utrwalania wywodzi się z klas z adnotacjami typu Entity
- ***Nawigacja***: Przechodzenie po związkach między encjami w wyrażeniach języka zapytań – operatorem nawigacji jest kropka.
- ***Ścieżka wyrażenia***: Wyrażenie, które nawiguje do pola stanu encji lub pola relacji.
- ***Pole stanu***: Pole utrwalanej encji
- ***Pole relacji***: Pole utrwalanej encji wykorzystane w relacji w wyrażeniu nawigacji, które jest typu *abstrakcyjnego schematu utrwalania* powiązanej encji

Proste zapytania

Podstawowe zapytanie typu Select

```
SELECT p FROM Customer p
```

Wynik: Wszystkie wystąpienia encji Customer.

Opis: Klauzula **FROM** deklaruje zmienną identyfikacji p, z pominięciem słowa kluczowego **AS**.

Jeśli użyto słowo kluczowe **AS**,

```
FROM AS p
```

element **Customer** jest abstrakcyjnym schematem encji **Customer**.

Eliminowanie duplikatów wartości

```
SELECT DISTINCT p FROM Customer p WHERE p.city = ?'Warszawa'
```

Wynik: Wystąpienia encji **Customer** (klientów) z wartościami atrybutu **city** równych wartości parametru zapytania ?'Warszawa'.

Opis: Słowo kluczowe **DISTINCT** eliminuje powtórzenia wystąpień wartości **city** w zbiorze klientów.

Klauzula **WHERE** wyszczególnia wystąpienia encji **Customer** za pomocą sprawdzania pola **city** utrwalanej encji **Customer**. Element ?'Warszawa' jest parametrem zapytania jako stała wartość.

Użycie parametrów o podanych nazwach

```
SELECT DISTINCT p FROM Customer p
```

```
WHERE p.city = :city AND p.firstName = :firstname
```

Wynik: Wystąpienia encji **Customer** (klienci) posiadający wyspecyfikowane atrybuty **city** i **firstname**.

Opis: Atrybuty **city** i **firstname** są utrwalanymi polami encji **Customer**. Klauzula **WHERE** porównuje wartości pól z wartościami parametrów zapytania, używając metody **Query.setNamedParameter**. Język zapytań pozwala nazywanie parametrów zapytania wprowadzając symbol (:) poprzedzający nazwę parametru. Pierwszym parametrem wejściowym jest **:city**, drugim jest **:firstname**.

Nawigacja powiązаныmi encjami (1)

Proste zapytanie dotyczące encji powiązanych relacją

```
SELECT DISTINCT p FROM Customer p, IN(p. accounts) t
```

Wynik: Wszystkie wystąpienia encji **Customer**, które należą do wystąpień encji **Account**.

Opis: Klazura **FROM** deklaruje dwie identyfikujące zmienne: **p**, **t**. Zmienna **p** reprezentuje encję **Customer**, zmienna **t** reprezentuje encję **Account**. Deklaracja zmiennej **t** odpowiada wcześniejszej deklaracji zmiennej **p**. Słowo kluczowe **IN** oznacza, że **accounts** jest kolekcją powiązanych wystąpień encji **Account**. Wyrażenie **p.accounts** nawiguje od encji **Customer** do powiązanej encji **Account**. Operator **kropka** w wyrażeniu **p.accounts** jest operatorem nawigacji.

Można zamiennie użyć polecenie **JOIN** do wyrażenia takiego samego zapytania:

```
SELECT DISTINCT p FROM Customer p JOIN p.accounts t
```

lub:

```
SELECT DISTINCT p FROM Customer p WHERE p.accounts IS NOT EMPTY
```

Nawigacja do jednowartościowych pól relacji

Użycie klauzuli **JOIN**:

```
SELECT t FROM Tx t JOIN t.account a
```

```
WHERE a.beginBalanceTimeStamp = '2008-05-23'
```

```
OR a.beginBalanceTimeStamp = '2008-05-28'
```

W przykładzie zapytanie zwraca wszystkie wystąpienia encji **Tx**, które są powiązane z wystąpieniami **a** encji **Account** (**t.account**) i w tych wystąpieniach encji **Account** wartość atrybutu **beginBalanceTimeStamp** jest równa wartościom '008-05-23' lub '2008-05-28' .

Nawigacja powiązаныmi encjami (2)

Użycie parametrów wejściowych w zapytaniach na powiązanych encjach

```
SELECT DISTINCT p FROM Customer p, IN (p.accounts) AS t WHERE t.type = :type
```

Wynik : Wystąpienia encji **Customer**, których rachunki posiadają atrybut **type** jest równy wartości parametru **:type**.

Opis : To zapytanie jest podobne do poprzedniego przypadku, różni się wprowadzeniem wejściowego parametru **:type**. Słowo kluczowe **AS** w klauzuli **FROM** jest opcjonalne. W klauzuli **WHERE**, wyrażenie **t.type** (operator kropki) przetwarzające pole utrwalane **type** pełni rolę operatora eliminacji niż operatora nawigacji. Jest ono zastosowane, ponieważ pole relacji **p.accounts** jest kolekcją powiązanych wystąpień encji **Account** z wystąpieniami encji **Customer**, a wyrażenia nie mogą nawigować polami relacji, które są kolekcjami (kolekcje są symbolami końcowymi wyrażen nawigacji). Dlatego wyrażenie **p.accounts.type** (jest to nieprawidłowe wyrażenie) zastąpiono wyrażeniem **t.type**, udostępniającym utrwalane pole **type**.

Zapytanie na wielokrotnych relacjach

```
SELECT DISTINCT p FROM Customer p, IN (p.accounts) t JOIN t.tx q  
WHERE q.account = :account
```

Wynik: Wystąpienia **p** różnych encji **Customer** posiadających rachunki zawierające się w zbiorze transakcji **tx** i są równe parametrowi **:account**

Opis: Wyrażenia w zapytaniu nawigują trzema relacjami. Wyrażenie **p.accounts** nawiguje relacją **Customer-Account**, i wyrażenie **t.tx** relacją **Account-Tx** oraz **q.account** relacją **Tx-Account**. W innych przykładach, wejściowymi parametrami są obiekty typu String, w przykładzie parametrem jest obiekt typu **account**. Ten typ oznacza **pole relacji** w wyrażeniu porównania klauzuli **WHERE**.

Nawigacja między powiązаныmi polami

```
SELECT DISTINCT p FROM Account p, IN (p.tx) t  
WHERE t.account.balance = :balance
```

Wynik: Wystąpienia encji **Account**, które posiadają wyspecyfikowaną wartość **balance** za pomocą parametru **:balance**.

Opis: Pole **balance** należy do encji **Account**. Aby osiągnąć pole **account**, zapytanie musi nawigować od encji **Account** do **Tx** (**p.tx**) i potem od encji **Tx** do encji **Account(t.account)**. Ponieważ pole relacji **account** nie jest kolekcją, dlatego prawidłowe jest wyrażenie **t.account.balance**.

Zapytania z różnymi wyrażeniami warunkowymi klauzuli WHERE

- **Wyrażenie LIKE**

SELECT p FROM Customer p WHERE p.lastName LIKE 'Mich%'

Wynik: Wszystkie wystąpienia encji **Customer**, których atrybuty **lastName** zaczynają się od podłańcucha "Mich."

Opis: Wyrażenia **LIKE** używają maski ze znakami % do wyszukiwania łańcuchów. W tym przypadku wyrażenie **LIKE** wykrywa wartości "Michael" i "Michelle,, za pomocą użytej maski (wzorca).

- **Wyrażenie IS NULL**

SELECT t FROM Tx t WHERE t.account IS NULL

Wynik: Wszystkie wystąpienia encji **Tx** związane z atrybutem **account** równym **NULL**.

Opis: Wyrażenie **IS NULL** może być użyte do testowania zbioru powiązań między wystąpieniami dwóch encji.

- **Wyrażenie IS EMPTY**

SELECT p FROM Customer p WHERE p.accounts IS EMPTY

Wynik: Wszystkie wystąpienia encji **Customer**, które nie należą do wystąpień encji **Account**.

Opis: Pola relacji są kolekcjami wystąpień encji **Customer**. Jeśli klient nie posiada rachunków, wtedy kolekcja wystąpień encji **Account** w polu **accounts** jest pusta, wtedy wyrażenie **IS EMPTY** jest równe **TRUE**.

- **Wyrażenie BETWEEN**

SELECT DISTINCT p FROM Account p
WHERE p.balance BETWEEN :lowerBalance AND :higherBalance

Wynik : Wszystkie wystąpienia encji **Account**, których atrybut **balance** zawiera się między podanymi granicami za pomocą parametrów **:lowerBalance** i **:higherBalance**.

Opis: Wyrażenie **BETWEEN** ma trzy arytmetyczne wyrażenia: pole utrwalania (**p.balance**) i dwa wejściowe parametry (**:lowerBalance** i **:higherBalance**). Następujące wyrażenie jest równoważne wyrażeniu **BETWEEN**:

p.balance >= :lowerBalance AND p.balance <= :higherBalance

- **Operatory porównania**

SELECT DISTINCT p1 FROM Account p1, Account p2
WHERE p1.balance > p2.balance AND p2.type = :type

Wynik: Wszystkie wystąpienia encji **Account**, których atrybut **balance** jest wyższy od wartości atrybutu **balance** tych wystąpień, których atrybut **type** jest równa wartości parametru **type**.

Opis: Klauzula **FROM** deklaruje dwie zmienne identyfikujące **p1** i **p2** tego samego typu **Account**. Obie zmienne są potrzebne, ponieważ klauzula **WHERE** porównuje atrybut **balance** jednego rachunku **p2** z atrybutem **balance** innego rachunku **p1**.

Zapytania typu Update i Delete

Następujące wyrażenia pokazują, jak użyć wyrażenia UPDATE i DELETE w zapytaniach. UPDATE i DELETE operują na wielokrotnych encjach w odniesieniu do warunków użytych w klauzuli WHERE. Klauzula WHERE w zapytaniach UPDATE i DELETE pełni taką samą rolę jak w zapytaniach SELECT.

Update

```
UPDATE Customer p SET p.city = 'Warszawa'  
WHERE p.middleInitial < :middleInitial
```

Opis: To zapytanie ustawia atrybut *city* zbioru wystąpień encji **Customer**, jeśli stan początkowy konta rachunku (wartość atrybutu **middleInitial**) była mniejsza niż wyspecyfikowana w parametrze **:middleInitial**

Delete

```
DELETE FROM Customer p  
WHERE p.lastName = :lastName AND p.accounts IS EMPTY
```

Opis: Zapytanie usuwa wszystkich klientów (zbioru wystąpień encji **Customer**, którzy mają atrybut **lastName** ma wartość „**lastName**”), którzy nie posiadają rachunków (atrybut **p.accounts** typu kolekcja jest pusty)

Przykład 4 – adnotacje zapytań, przykład zastosowania

```
@NamedQueries ({
  @NamedQuery(name = "Customer.FindById",
    query = "SELECT a FROM Customer a WHERE a.id = :id")
  ,@NamedQuery(name = "Customer.FindByLastName",
    query = "SELECT a FROM Customer a WHERE a.lastName LIKE :lastName")
  ,@NamedQuery(name = "Customer.FindByFirstName",
    query = "SELECT a FROM Customer a WHERE a.firstName = :firstName")
  ,@NamedQuery(name = "Customer.FindByMiddleInitial",
    query = "SELECT a FROM Customer a WHERE a.middleInitial = :middleInitial")
  ,@NamedQuery(name = "Customer.FindByStreet",
    query = "SELECT a FROM Customer a WHERE a.street = :street")
  ,@NamedQuery(name = "Customer.FindByCity",
    query = "SELECT a FROM Customer a WHERE a.city = :city")
  ,@NamedQuery(name = "Customer.FindByState",
    query = "SELECT a FROM Customer a WHERE a.state = :state")
  ,@NamedQuery(name = "Customer.FindByZip",
    query = "SELECT a FROM Customer a WHERE a.zip = :zip")
  ,@NamedQuery(name = "Customer.FindByPhone",
    query = "SELECT a FROM Customer a WHERE a.phone = :phone")
  ,@NamedQuery(name = "Customer.FindByEmail",
    query = "SELECT a FROM Customer a WHERE a.email = :email")
  ,@NamedQuery(name="Customer.FindAllCustomersOfAccount",
    query="SELECT c FROM Customer c JOIN c.accounts a WHERE a.id = :accountId" )))
```

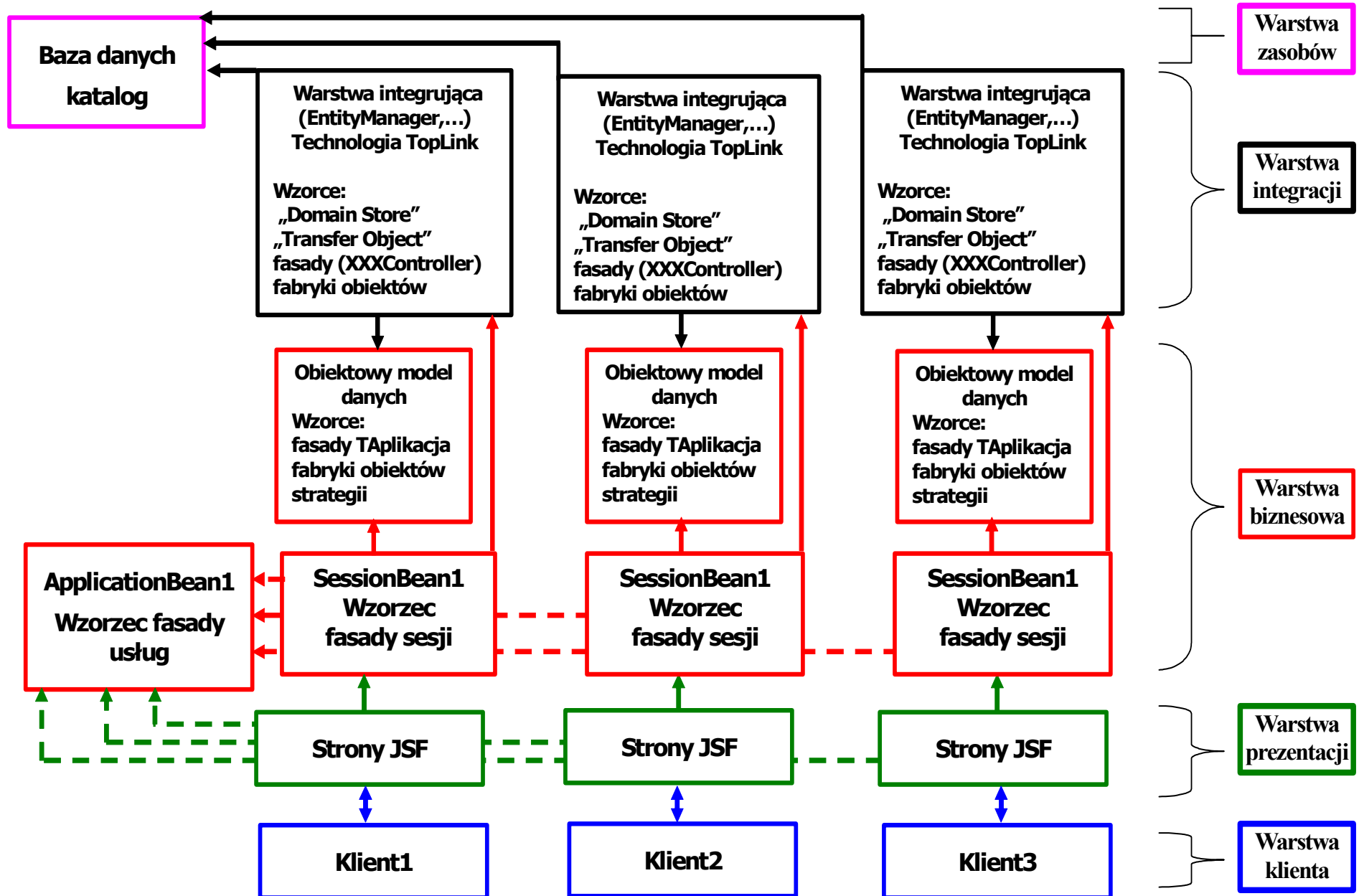
```
@NamedQuery(name="Customer.FindAllCustomersOfAccount",
            query="SELECT c FROM Customer c JOIN c.accounts a
                  WHERE a.id = :accountId")
```

```
customers = em.createNamedQuery("Customer.FindAllCustomersOfAccount")
            .setParameter("accountId", accountId)
            .getResultList();
```

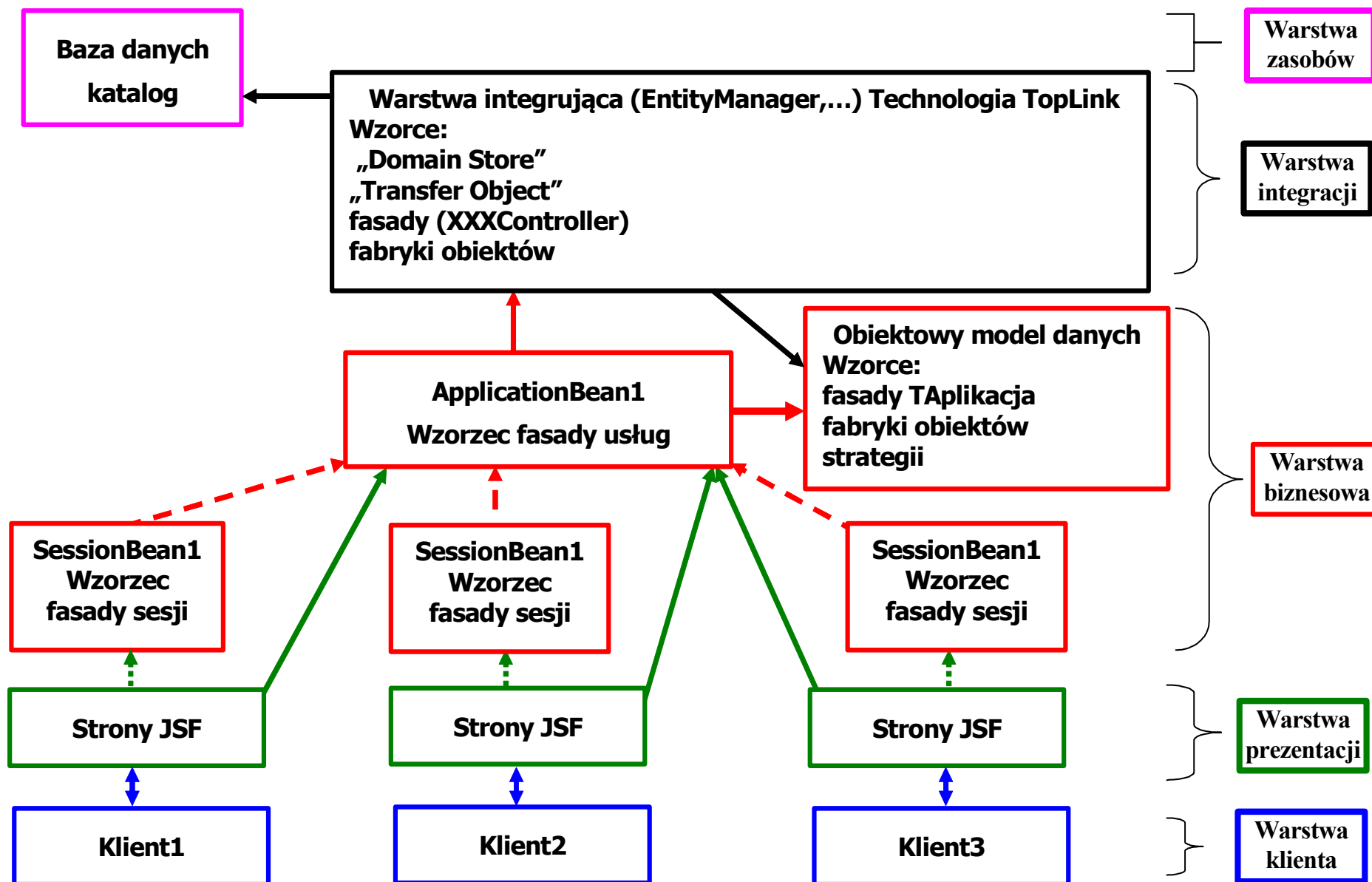
```
public List<Long> getCustomerIds(Long accountId) throws RuntimeException
{
    Debug.print("AccountControllerBean getCustomerIds");
    List<Customer> customers = null;
    EntityManager em=getEntityManager();
    if (accountId == null)
        { throw new RuntimeException("null accountId"); }
    try
    { customers = em.createNamedQuery( "Customer.FindAllCustomersOfAccount")
                  .setParameter("accountId", accountId).getResultList();
      if (customers == null)
          { throw new RuntimeException(); }
    } catch (Exception ex)
        { throw new RuntimeException(ex); }
    em.close();
    return copyCustomerIdsToList(customers);
}
```

**3. Wzorce warstwy
biznesowej: fasada sesji i
fasada usług**

Architektura aplikacji pięciowarstwowej (przykład z wykładu 5 - punkt 4)- linie przerywane oznaczają powiązania nie wykorzystane w aplikacji



Architektura aplikacji pięciowarstwowej (przykład z wykładu 6 – punkt 4)- linie przerywane oznaczają powiązania nie wykorzystywane w aplikacji



Problem 1 – udostępnianie komponentów i usług biznesowych zdalnym klientom

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

Uwagi:

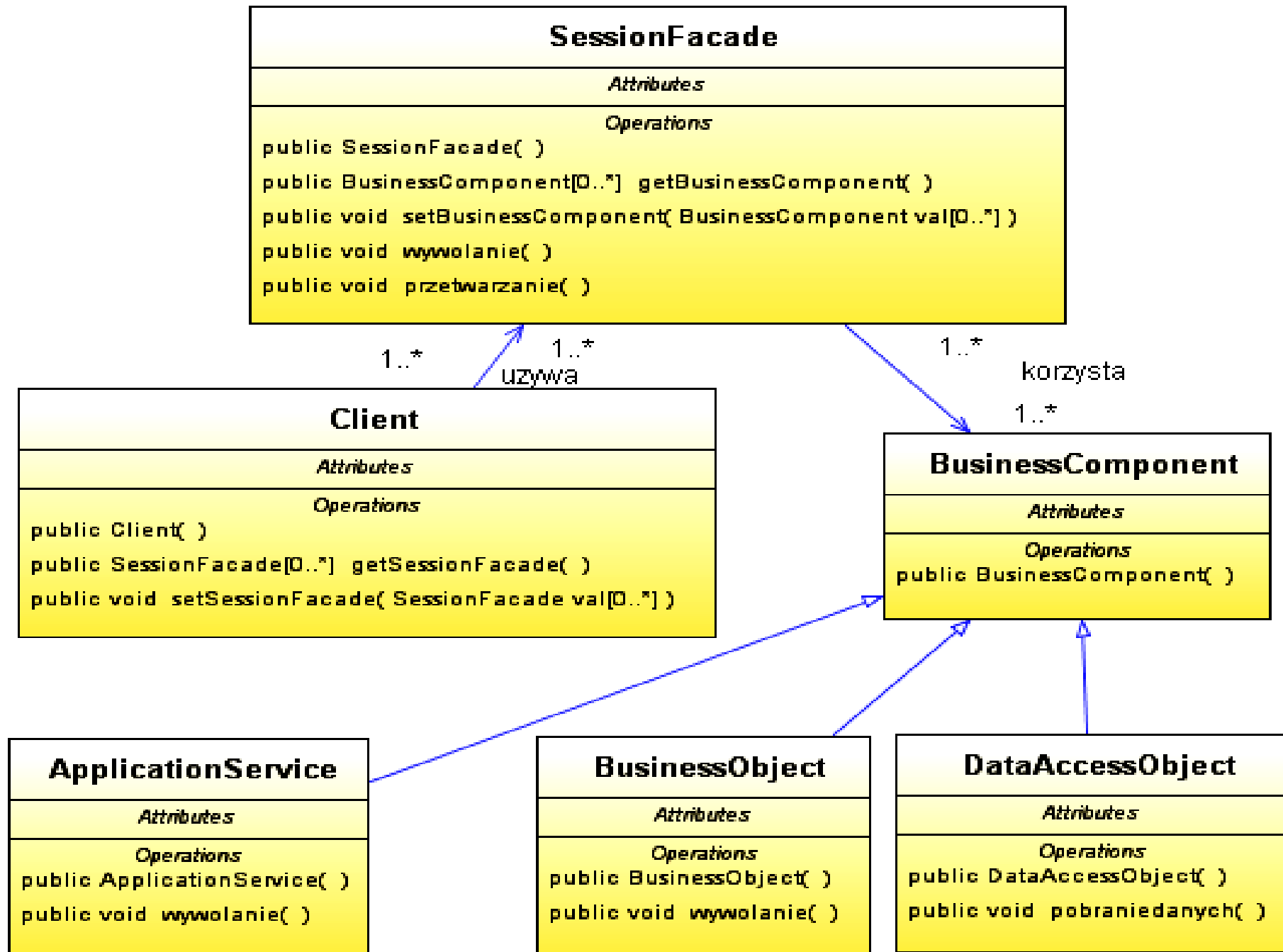
1. Często **warstwa biznesowa** jest realizowana po stronie serwera w postaci obiektów *Business Object*, zwykłych obiektów Javy lub komponentów Entity – wszystkie typu „Entity”.
 - Kiedy klienci bezpośrednio komunikują się z komponentami usług biznesowych, zmiana interfejsu komponentu biznesowego ma bezpośredni wpływ na kod klienta.
 - Bezpośredni dostęp wymaga, by klient zawierał złożoną logikę zajmującą się koordynacją i interakcją pomiędzy wieloma komponentami biznesowymi powiązanymi złożonymi relacjami. Wtedy kod klienta zawiera złożone operacje wyszukiwania, zarządzania transakcjami, bezpieczeństwem i sam przeprowadza przetwarzanie biznesowe.
 - Kiedy istnieje wiele typów klientów, bezpośredni dostęp do komponentów biznesowych prowadzi do nierównomiernego korzystania z usług i powielania kodu u klientów. Utrudnia to pielęgnację kodu klienta i zmniejsza elastyczność jego zastosowania.
2. **Rozdrobnienie komponentów na jedną usługę biznesową** prowadzi do wielu komunikacji w sieci z powodu zdalnych wywołań do rozdrobnionych komponentów

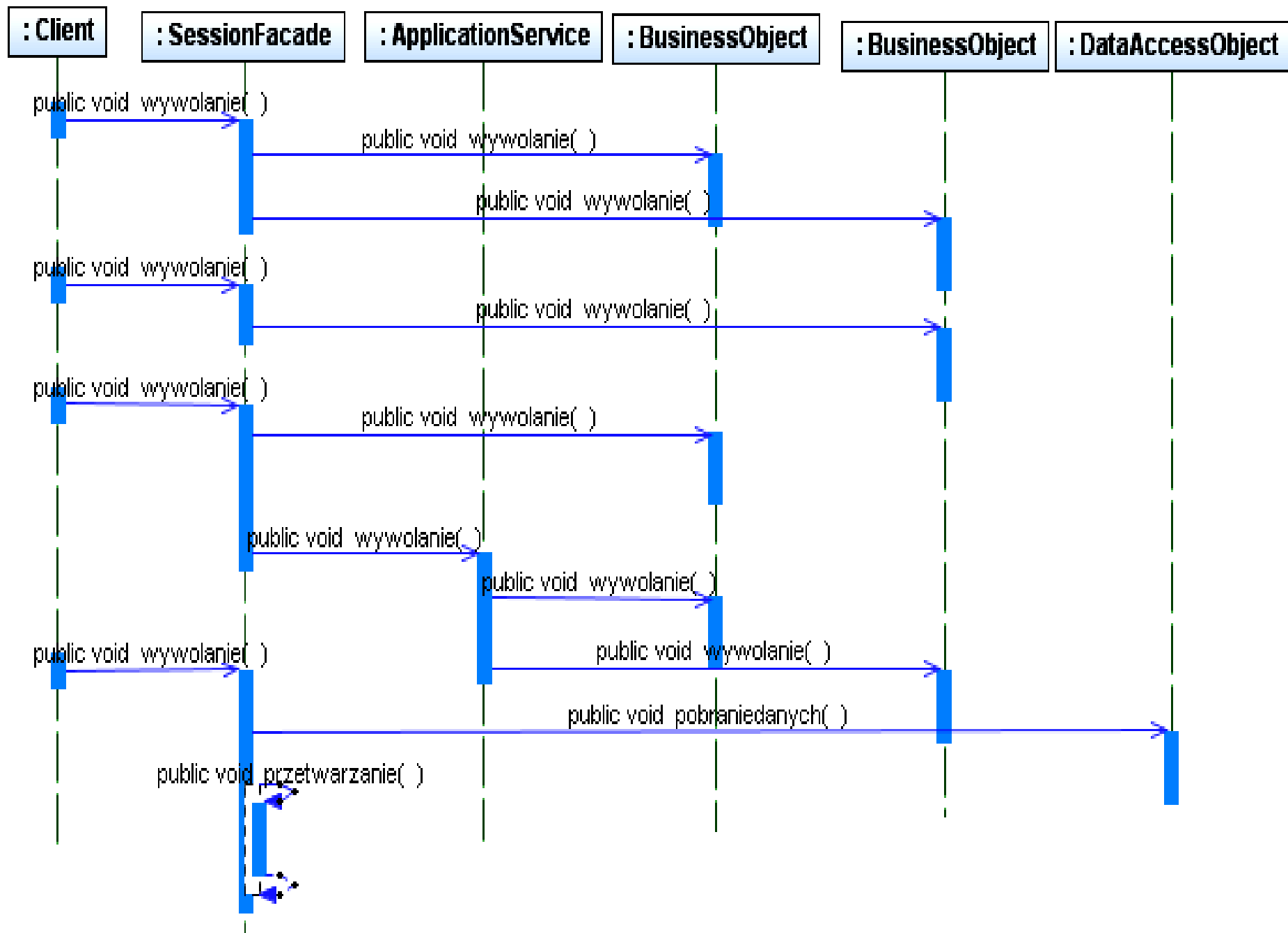
Wymagania:

1. Należy unikać bezpośredniego udostępnienia klientom komponentów warstwy biznesowej, aby nie dopuścić do powstania zbyt wielu zależności między klientami a warstwą biznesową
2. Należy zapewnić zdalną warstwę dostępu do obiektów **Business Object** lub innych obiektów warstwy biznesowej
3. Należy zgrupować i udostępnić zdalnym klientom usługi aplikacji (implementować wzorzec **Application Service**) oraz dowolne inne usługi
4. Należy scentralizować i połączyć całą logikę biznesową udostępnianą zdalnym klientom
5. Należy ukryć złożone interakcje i wzajemne zależności między komponentami i usługami biznesowymi w celu ułatwienia zarządzania, centralizacji logiki, zwiększenia elastyczności i ułatwienia wprowadzania zmian.

Wzorzec Session Facade (**fasada sesji**) – udostępnia klientom ogólne usługi biznesowe realizujące powiązane przypadki użycia, na przykład służące do obsługi rachunku bankowego

- **Client** – klient komponentu **Session Facade**, który chce uzyskać dostęp do usług biznesowych.
- **BusinessComponent** – uczestniczy w realizacji żądań klienta. Może nim być **Business Object** jako model obiektowy danych i zachowanie biznesowe lub jako **ApplicationService** lub jako **DataAccessObject**
- **ApplicationService** – obiekt ten korzysta z obiektów biznesowych i implementuje logikę biznesową. Komponent **Session Facade** może korzystać z wielu takich obiektów
- **DataAccessObject** – pośredniczy w dostępie do bazy danych w prostych aplikacjach, w których nie ma warstwy obiektów biznesowych tworzących model obiektowy danych





Problem 2 – centralizacja logiki biznesowej kilku komponentów i usług biznesowych

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

Uwagi:

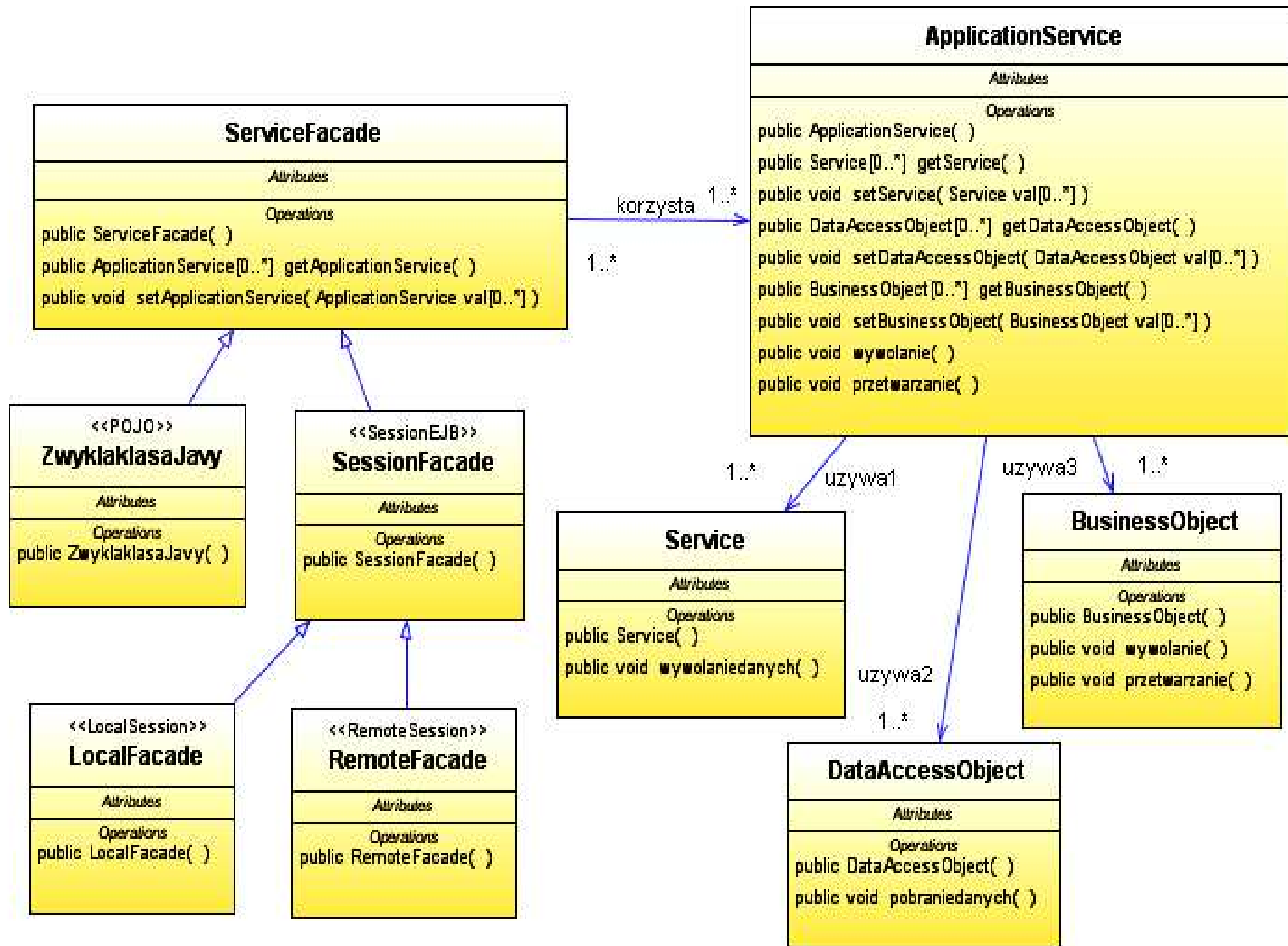
1. Fasady sesji, na przykład **Session Facade** lub fasady zwykłych obiektów zawierają niewiele logiki biznesowej i udostępniają prosty, nierozdrobniony interfejs.
2. Aplikacje implementują przypadki użycia koordynujące współpracę kilku obiektów biznesowych i usług. Nie należy jednak implementować logiki zarządzającej relacjami pomiędzy obiektami biznesowymi w samych obiektach, gdyż zwiększa to zależności pomiędzy nimi i zmniejsza elastyczność. Logiki nie należy umieszczać w fasadzie, gdyż mogłaby ona zostać powielona w wielu komponentach fasady.
3. Aplikacje, które nie stosują komponentów EJB, implementują komponenty warstwy biznesowej jako zwykłe obiekty. Ta logika biznesowa nie powinna być umieszczana ani w fasadzie sesji, ani w kodzie klienta.

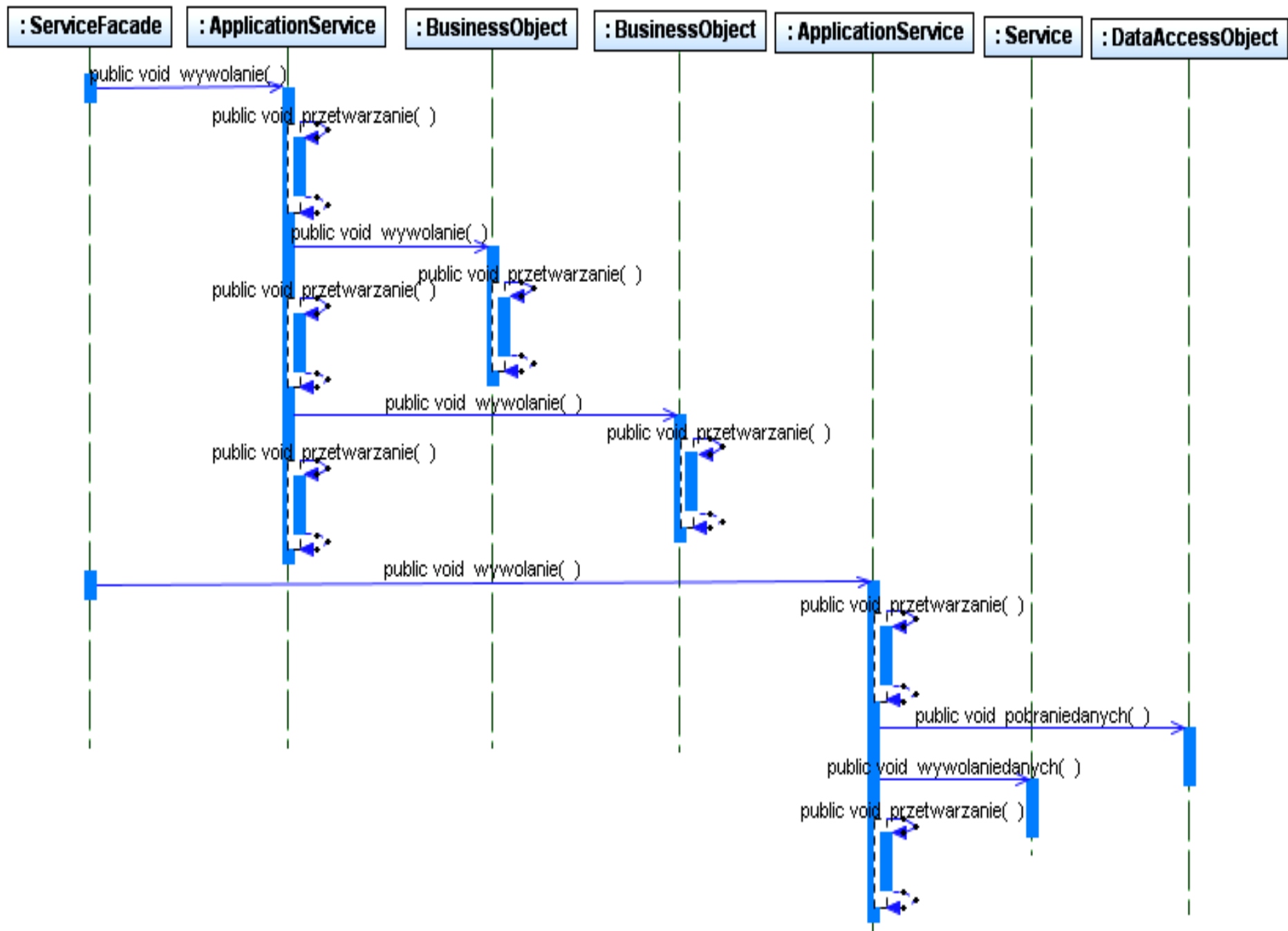
Wymagania:

1. Należy ograniczyć ilość logiki biznesowej w fasadach sesji.
2. Logika biznesowa operuje na kilku obiektach biznesowych lub usługach.
3. Należy utworzyć nierozdrobniony interfejs usług nad istniejącymi komponentami i usługami biznesowymi.
4. Należy umieścić logikę związaną z konkretnymi przypadkami użycia obiektami biznesowymi (głównie dotyczącą wzajemnych zależności pomiędzy obiektami biznesowymi) poza obiektami **Business Object**

Wzorzec **Application Service (fasada usług)** – zapewnia jednolitą warstwę usług i stanowi zaplecze fasad.

- **Client** – fasada typu **Session Facade**, obiekty zwykłych klas Javy, inny obiekt typu **Application Service**
- **Application Service** – pełni główną rolę, hermetyzuje logikę biznesową operującą na kilku obiektach biznesowych lub opartą na konkretnym przypadku użycia. Może on wywoływać metodę biznesową obiektu typu **BusinessObject** lub innego obiektu typu **ApplicationService** lub obiektu dostępu do danych **DataAccessObject**
- **BusinessObject** – kilka obiektów tego typu do realizacji żądania **ApplicationService**
- **Service** – komponent udostępniający dowolny rodzaj usługi
- **DataAccessObject** – obiekty dostępu do danych





4. Architektura typu wielu klientów ze wspólną warstwą biznesową istniejącą podczas sesji i wspólną warstwą integrującą z bazą danych,

Przykład projektu składającego się z wielu formularzy opartych na fragmentach stron internetowych

typu JSPF

(wg wykładu 5)

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

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](#)

File Explorer (WebWypożyczalnia2):

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

Navigator (jsp:directive.include:Menu.jspf - Navigator):


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

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%

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

The screenshot displays an IDE interface for a web application project named "WebWypożyczalnia2". The main design view shows a multi-layered application layout on a grid background. At the top left, there is a small image of a sunset over water. Below it, a navigation menu is visible with links: [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](#). To the right of the menu is a form titled "Dodaj tytuł" with input fields for "Tytuł", "Autor", "ISBN", "Wydawnictwo", and "Aktor". A dropdown menu is located below the "Aktor" field. Further right, there is a "System Messages" box containing a red bullet point and the text "List of all message summaries", and a "Dodaj tytuł" button.

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

The "Tytuly - Navigator" window at the bottom left shows the following structure:

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

At the bottom of the IDE, there is an "HTTP Monitor" window.

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 książkę 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%

Projekty formularza „Dodaj książki w aplikacji” (Ksiazki.jsp)


The screenshot displays an IDE interface for a web application project. The main workspace shows a design view of a form titled "Przykład wielowarstwowej aplikacji". The form contains a navigation menu with links: "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". A form section includes two input fields labeled "Numer" and "Termin", a "Dodaj książkę" button, and a dropdown menu. The left sidebar shows the project structure for "WebWypożyczalnia2", listing files such as "FormKsiazka.jspf" and "Ksiazki.jsp". The bottom left panel, "Ksiazki - Navigator", shows a tree view of the page structure, including "page1", "html1", "head1", "body1", and "Form1".

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

http://localhost:8080/WebWypożyczalnia2/faces/Ksiazki.j... Live Search

Norton™ Monitorowanie fałszywych witryn jest włączone 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

Numer:

Termin:

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

Dodaj książkę

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

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

Tytuł: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2 Numer: 1 termin: Tue May 06 00:00:00 CEST 2008

Tytuł: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2 Numer: 3

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

Gotowe Internet | Tryb chroniony: włączony 100%

Projekty formularza „Dodaj tytuł do bazy” (Baza_tytul.jsp)

The screenshot displays an IDE interface for a web application project named 'WebWypożyczalnia2'. The main design view shows a page titled 'Przykład wielowarstwowej aplikacji' (Example of a multi-layered application). The page contains a navigation menu, a form for adding a book title, a table of titles, and a system messages area.

Navigation Menu:

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

Add Title Form:

Tytuł:

Autor:

ISBN:

Wydawnictwo:

Aktor:

System Messages:

- List of all message summaries

Titles Table:

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

Project Structure (Left Panel):

- 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
 - Tytulybaza.jspf

Baza_tytul - Navigator (Bottom Left):


- Baza_tytul
 - page1
 - html1
 - head1
 - body1
 - form1
 - div
 - div
 - div
 - div
 - bazatytul:Zapisz tytuł do bazy
 - messageGroup1

HTTP Monitor (Bottom):

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

http://localhost:8080/W... Norton™ Monitorowanie fałszywych witryn jest włączone

Przykład wielowarstwowej aplikacji



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

Tytuł

Autor

ISBN

Wydawnictwo

Aktor

Zapisz tytuł 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%

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

The screenshot displays an IDE window for a web application project named "WebWypożyczalnia2". The main design view shows a page titled "Przykład wielowarstwowej aplikacji". The page layout includes a navigation menu with the following items:

- [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](#)

Below the menu is a button labeled "Zapisz tytuły do bazy". To the right of the button is 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 IDE interface also shows a file explorer on the left with the following files:

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

The "Baza_tytuly - Navigator" at the bottom left shows the following structure:


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

The IDE also shows a "RequestBean1" at the bottom left.

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%

Projekty formularza „Przepisz książki do bazy” (Baza_książki.jsp)

The screenshot shows an IDE interface for a web application project named "WebWypożyczalnia2". The main design view displays a page titled "Przykład wielowarstwowej aplikacji" (Example of a multi-layered application). The page contains a navigation menu with links: "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". A table is also present, showing a list of books with columns for id, numer, termin, and tytuł książki. The table contains three rows of data.

The "dodajksiążkabaza:Zapisz książki do bazy - Navigator" panel shows the project structure for the "Baza_książki" package. The structure is as follows:

- Baza_książki
 - page1
 - html1
 - head1
 - body1
 - form1
 - div
 - div
 - dodajksiążkabaza:Zapisz książki do bazy
 - div

The table data is as follows:


id	numer	termin	tytuł książki
123	123	Thu May 08 16:21:40 CEST 2008	Tytuł: abc Autor: abc ISBN: abc Wydawnictwo: abc
123	123	Thu May 08 16:21:40 CEST 2008	Tytuł: abc Autor: abc ISBN: abc Wydawnictwo: abc
123	123	Thu May 08 16:21:40 CEST 2008	Tytuł: abc Autor: abc ISBN: abc Wydawnictwo: abc

http://localhost:8080/WebWypożyczalnia2/faces/Baza_książki.jsp;jsessionid=8ea329a1ec81f0fdef19b - Windows Internet ...

http://localhost:8080/WebWypożyczalnia2/faces/Baza_książki.jsp; Live Search

http://localhost:8080/WebWypożyczalnia2/faces... Strona Narzędzia

Norton™ Monitorowanie fałszywych witryn jest włączone Opcje



Przykład wielowarstwowej aplikacji

Zapisz książki 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](#)

Table			
id	numer	termin	tytuł_książki
3	1	Tue May 06 00:00:00 CEST 2008	Tytuł: 1 Autor: 1 ISBN: 1 Wydawnictwo: 1
52	2		Tytuł: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2
102	1	Tue May 06 00:00:00 CEST 2008	Tytuł: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2
153	1		Tytuł: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2
202	3		Tytuł: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2
302	4		Tytuł: 2 Autor: 2 ISBN: 2 Wydawnictwo: 2 Aktor: 2
402	1	Fri May 09 00:00:00 CEST 2008	Tytuł: 4 Autor: 4 ISBN: 4 Wydawnictwo: 4

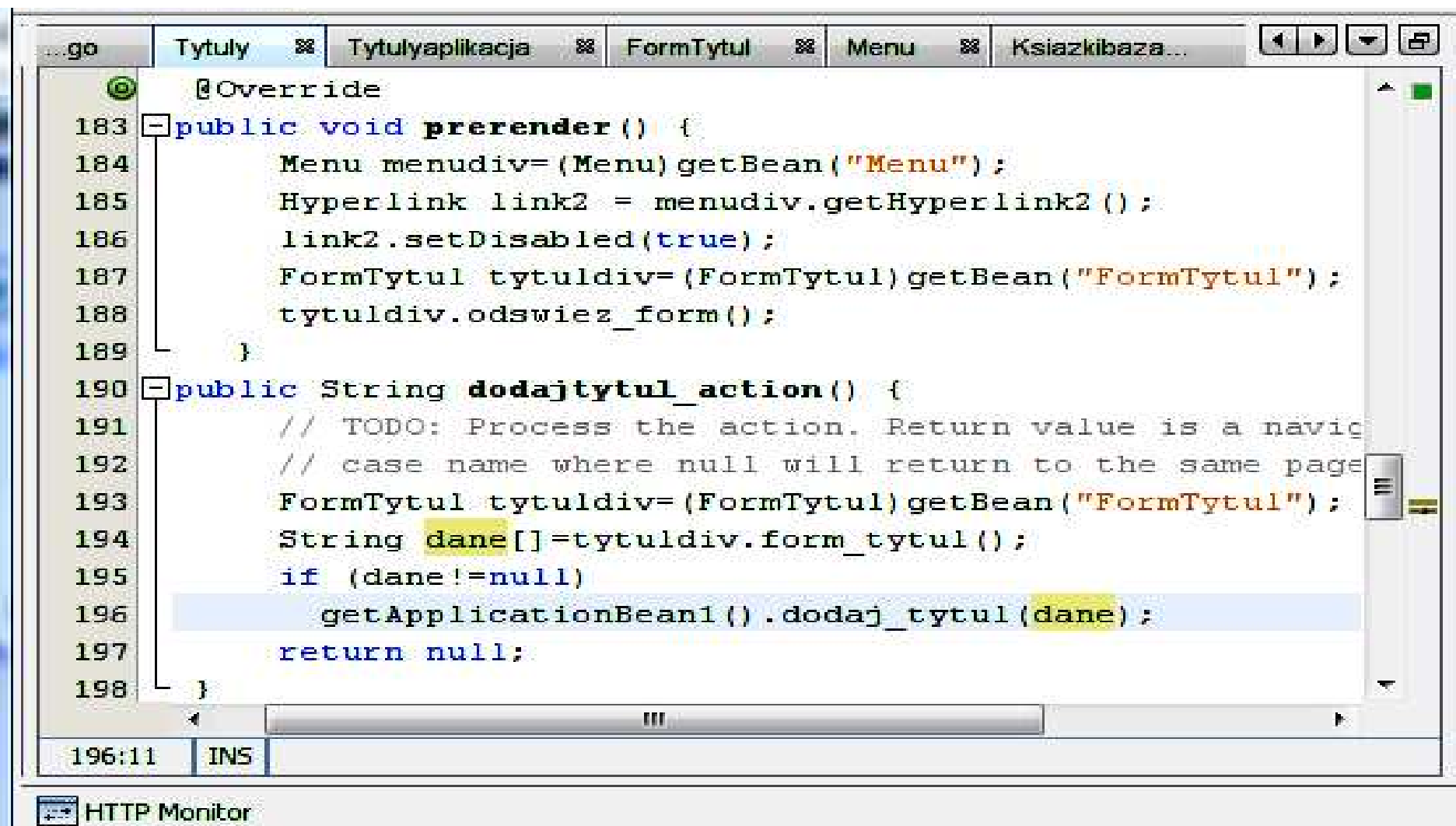
Internet | Tryb chroniony: włączony 100%

**5. Oprogramowanie systemu
typu wielu klientów ze
wspólną warstwą biznesową
istniejącą podczas sesji i
wspólną warstwą integrującą
z bazą danych,**

**Przykład projektu składającego się z
wielu formularzy opartych na
fragmentach stron internetowych
typu JSPF**

5.1. 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 w fazie Response przetwarzania strony 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 | HTTP Monitor
```

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)

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

212:6 | INS

HTTP Monitor

Save All finished.

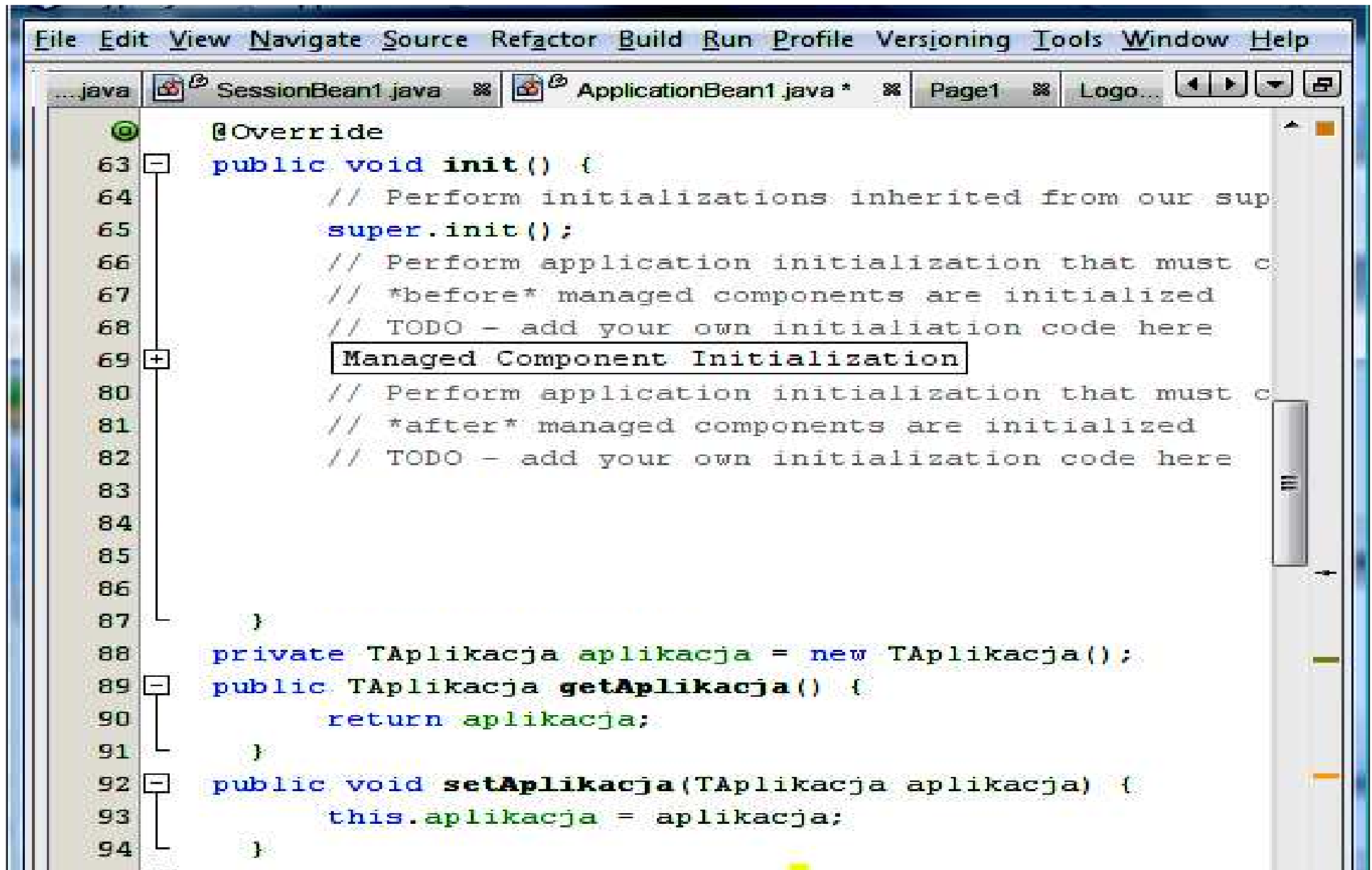
Definicje metod w klasie ApplicationBean1 związanych z zapisem (dodaj_tytul) i odczytem (przygotujtytuly) danych typu kolekcja obiektów TTytul_ksiazki i TTytul_ksiazki_na_kasecie w warstwie biznesowej – odczytane dane wstawiane są do tablicy tytuly_, która jest wyświetlana w komponencie typu DropDown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

```
...java  SessionBean1.java  ApplicationBean1.java  Page1  Logo  Tytuly...
135  public void  dodaj_tytul(String dane[])
136  {getAplikacja().dodaj_tytul(dane); //przypadek użycia "dodaj tytulu"
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_ksiazki> tytuly = aplikacja.getTytul_ksiazki();
148  int ile = tytuly.size();
149  if (ile > 0) {
150  Option pom[] = new Option[ile];
151  Iterator iterator = tytuly.iterator();
152  int i = 0;
153  while (iterator.hasNext()) {
154  pom[i++] =
155  new Option(Integer.toString(i), iterator.next().toString());
156  }
157  tytuly_ = pom;
158  }
159  }
```

159:5 INS

HTTP Monitor

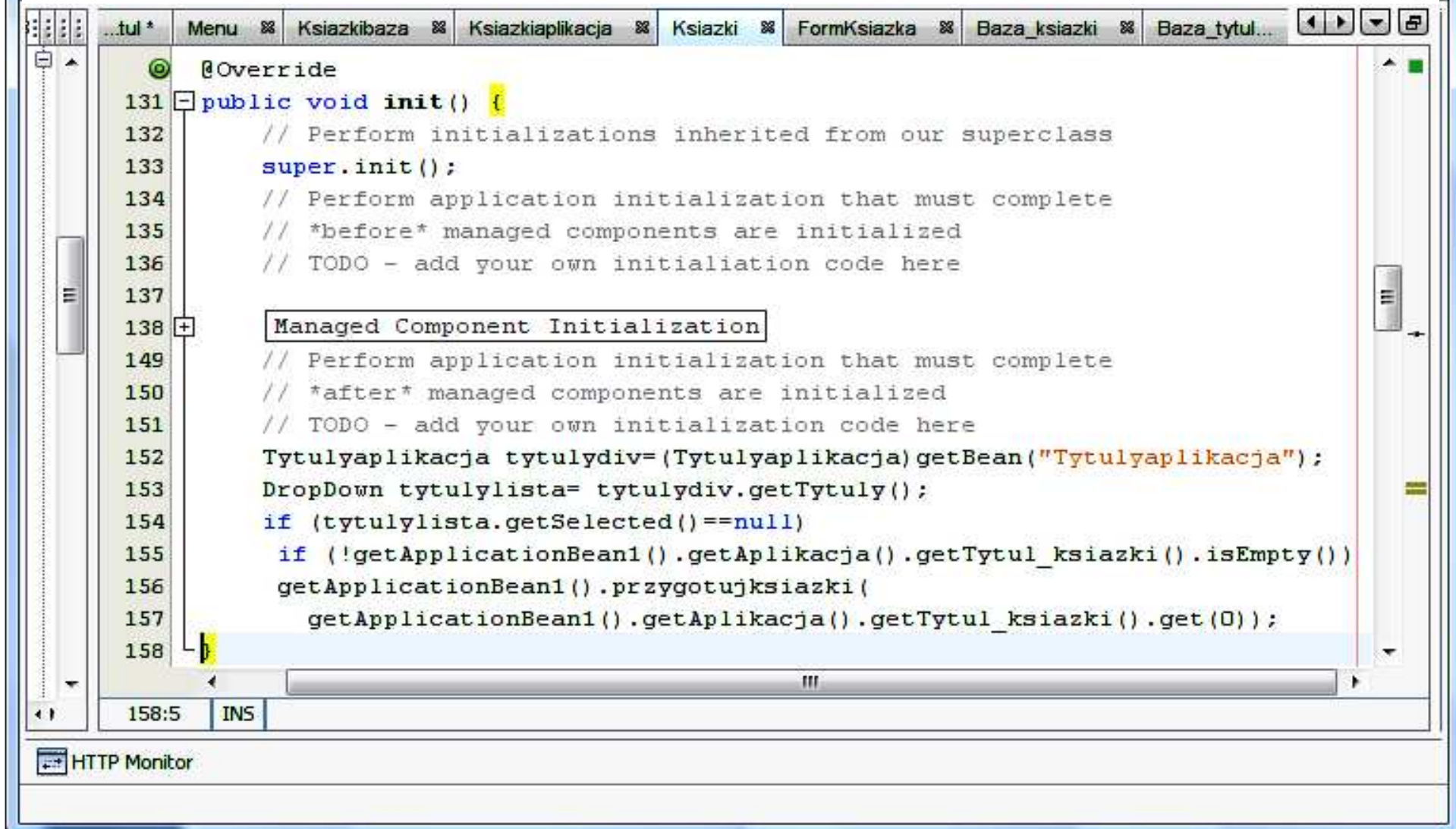
Utworzenie warstwy biznesowej oraz obiektu typu TApplikacja, który jest fasadą warstwy biznesowej w postaci zwykłego obiektu Javy



```
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 TApplikacja aplikacja = new TApplikacja();
89 public TApplikacja getApplikacja() {
90     return aplikacja;
91 }
92 public void setApplikacja(TApplikacja aplikacja) {
93     this.aplikacja = aplikacja;
94 }
```

5.2. Oprogramowanie dotyczące formularza Ksiazki.jsp

Definicje metody `init` w klasie `Ksiazki` dla stron typu JSP – zainicjowanie zawartości komponentu typu DropDown List na stronie `Ksiazkiaplikacja` typu JSPF informacjami o książkach (metoda `przygotujksiazki` opisana dalej) przez wybór pierwszego tytułu na stronie `Tytulyaplikacja` typu JSPF (o ile ten zbiór tytułów nie jest pusty)



```
...tul *  Menu  Ksiazkibaza  Ksiazkiaplikacja  Ksiazki  FormKsiazka  Baza_ksiazki  Baza_tytul...
@Override
131 public void init() {
132     // Perform initializations inherited from our superclass
133     super.init();
134     // Perform application initialization that must complete
135     // *before* managed components are initialized
136     // TODO - add your own initialization code here
137
138     Managed Component Initialization
139
140     // Perform application initialization that must complete
141     // *after* managed components are initialized
142     // TODO - add your own initialization code here
143
144     Tytulyaplikacja tytulydiv=(Tytulyaplikacja)getBean("Tytulyaplikacja");
145     DropDown tytulylista= tytulydiv.getTytuly();
146     if (tytulylista.getSelected()==null)
147         if (!getApplicationBean1().getAplikacja().getTytul_ksiazki().isEmpty())
148             getApplicationBean1().przygotujksiazki(
149                 getApplicationBean1().getAplikacja().getTytul_ksiazki().get(0));
150
151 }
```

158:5 INS

HTTP Monitor

Definicje metod w klasie Ksiazki dla strony typu JSP – do wstawiania nowej książki do warstwy biznesowej (obsługa zdarzenia dodajksiazke_action) oraz generowania widoku w fazie Response przetwarzania strony metoda prerender (wygaszanie linku do bieżącej strony w formularzu Menu typu JSPF i czyszczenie pól formularza FormKsiazka typu JSPF jego metodą odswiez_form)

```
...nu | Ksiazkibaza | Ksiazkiaplikacja | Ksiazki | FormKsiazka | Baza_ksiazki | Baza_tytul...
@Override
179 public void prerender() {
180     Menu menudiv=(Menu)getBean("Menu");
181     Hyperlink link3 = menudiv.getHyperlink3();
182     link3.setDisabled(true);
183     FormKsiazka ksiazkativ=(FormKsiazka)getBean("FormKsiazka");
184     ksiazkativ.odswiez_form();
185 }
+ /**...*/
194 public String dodajksiazke_action() {
195     // TODO: Process the action. Return value is a navigation
196     // case name where null will return to the same page.
197     Tytulyaplikacja tytulydiv=(Tytulyaplikacja)getBean("Tytulyaplikacja");
198     String dane1[]= tytulydiv.wybor_tytulu();
199     FormKsiazka ksiazkativ=(FormKsiazka)getBean("FormKsiazka");
200     String dane2[]=ksiazkativ.form_ksiazka();
201     if (dane1!=null&& dane2!=null)
202         getApplicationBean1().dodaj_ksiazke(dane1,dane2);
203     return null;
204 }
```

186:1 | INS

HTTP Monitor

Definicje metod w klasie FormKsiazka typu BackingBean dla stron typu JSPF – do pobierana danych o nowej książce (form_ksiazka): dane1 dla wyszukania tytułów wstawianych książek (metoda wybor_tytulu z ze strony Tytulyaplikacja typu JSPF) i dane2 z danymi wstawianej książki; oraz do czyszczenia pól formularza (odswiez_form)

```
158 public void odswiez_form()
159 {
160     ksiazkiaplikacjaPanel.setRendered(true);
161     numer.setText("");
162     termin.setText("");
163 }
164 public String[] form_ksiazka() {
165     // TODO: Process the action. Return value is a navigation
166     // case name where null will return to the same page.
167     String jaka;
168     if (numer.getText().equals(""))
169         return null;
170     if (termin.getText().equals(""))
171         { jaka = "0"; }
172     else
173         { jaka = "1"; } // Informacja dla fabryki - : jaki egzemplarz utworzyć c
174     String dane[] = {jaka, (String) numer.getText(), (String) termin.getText()};
175     return dane;
176 }
```

168:39 INS

HTTP Monitor

Definicje metod w klasie Tytulyaplikacja typu BackingBean dla stron typu JSPF – do pobierania danych o tytule dane1 do wyszukiwania (wybor_tytulu): ISBN lub ISBN i nazwisko aktora; oraz do zmiany zawartości komponentu typu DropDown List na stronie Ksiazkiaplikacja typu JSPF (metodą przygotujksiazki) za pomocą obsługi zdarzenia wyboru pozycji z listy metodą tytuly_processValueChange

```
120 public String [] wybor_tytulu()
121 { String jaki;
122   int nr = Integer.parseInt((String)tytuly.getSelected());
123   if (nr==0) return null;
124   Option pom[]=getApplicationBean1().getTytuly_();
125   String pom1=pom[nr-1].getLabel();
126   String pom2[]=pom1.split(" ");
127   if (pom2.length==8)//Informacja dla fabryki - jaki tytuł utworzyć do szukania
128     { pom1="";
129       jaki="0"; }
130   else
131     { pom1=pom2[9];
132       jaki="2";}
133   String dane1[]= {jaki, (String) pom2[5], pom1};
134   return dane1;
135 }
136 public void tytuly_processValueChange (ValueChangeEvent event) {
137   String dane[]=wybor_tytulu();
138   if (dane!=null)
139     getApplicationBean1().przygotujksiazki (
140     getApplicationBean1().getAplikacja().Wyszukaj_tytul(dane));
```

138:6 INS

HTTP Monitor

Save All finished.

Definicje metod w klasie ApplicationBean1 związanych z zapisem (dodaj_książke) i odczytem (przygotujksiazki) danych typu kolekcja obiektów TEgzemplarz i TEgzemplarz_termin w warstwie biznesowej – odczytane dane wstawiane są do tablicy ksiazki_, która jest wyświetlana w komponencie typu DropDown List na stronie Ksiazkiaplikacja typu JSPF

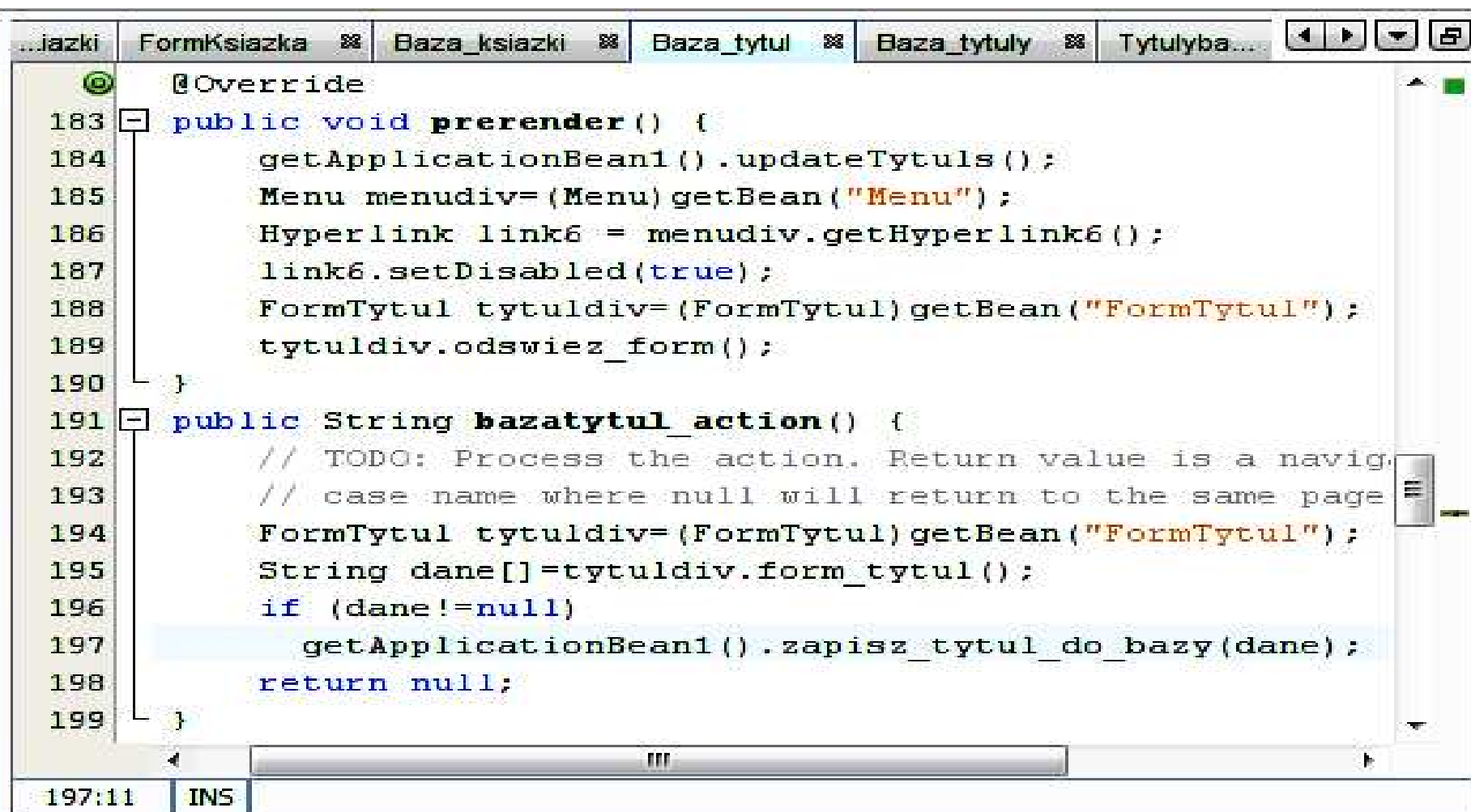
```
...t.java  SessionBean1.java  ApplicationBean1.java  Page1  Logo  Tytuly...
160     private Option ksiazki_[] = new Option[0];
161     public Option[] getKsiazki_() {
162         return ksiazki_;
163     }
164     public void setKsiazki_(Option[] ksiazki_) {
165         this.ksiazki_ = ksiazki_;
166     }
167     public void przygotujksiazki(TTytul_ksiazki tytul) {
168         if (tytul == null) {
169             return;
170         }
171         ArrayList<TEgzemplarz> ksiazki = tytul.getMKsiazka();
172         int ile = ksiazki.size();
173         if (ile > 0) {
174             Option pom[] = new Option[ile];
175             Iterator iterator = ksiazki.iterator();
176             int i = 0;
177             while (iterator.hasNext()) {
178                 pom[i++] =
179                     new Option(Integer.toString(i), iterator.next().toString());
180             }
181             ksiazki_ = pom;
182         }
183     }
184     public void dodaj_ksiazke(String dane1[], String dane2[])
185     { przygotujksiazki(getAplikacja().dodaj_ksiazke(dane1, dane2)); }
```

178:69 INS

HTTP Monitor

5.3. Oprogramowanie dotyczące formularza „Dodaj tytuł do bazy”

Definicje metod w klasie `Baza_tytul` dla strony typu JSP – do zapisu tytułów z warstwy biznesowej (obsługa zdarzenia `bazatytul_action`) oraz generowania widoku w fazie Response przetwarzania strony metoda `prerender` (wygaszanie linku do bieżącej strony w formularzu `Menu` typu JSPF, aktualizacja tablicy tytuły metodą `updateTytuls` w klasie `ApplicationBean1`, wyświetlanej w komponencie `Table` strony `Tytulybaza` typu JSPF, czyszczenie pól formularza `FormTytul` typu JSPF jego metodą `odswiez_form`)



```
...jazki | FormKsiazka | Baza_ksiazki | Baza_tytul | Baza_tytuly | Tytulyba...
@Override
183 public void prerender() {
184     getApplicationBean1().updateTytuls();
185     Menu menudiv=(Menu) getBean("Menu");
186     Hyperlink link6 = menudiv.getHyperlink6();
187     link6.setDisabled(true);
188     FormTytul tytuldiv=(FormTytul) getBean("FormTytul");
189     tytuldiv.odswiez_form();
190 }
191 public String bazatytul_action() {
192     // TODO: Process the action. Return value is a navig.
193     // case name where null will return to the same page
194     FormTytul tytuldiv=(FormTytul) getBean("FormTytul");
195     String dane[]=tytuldiv.form_tytul();
196     if (dane!=null)
197         getApplicationBean1().zapisz_tytul_do_bazy(dane);
198     return null;
199 }
```

197:11 | INS

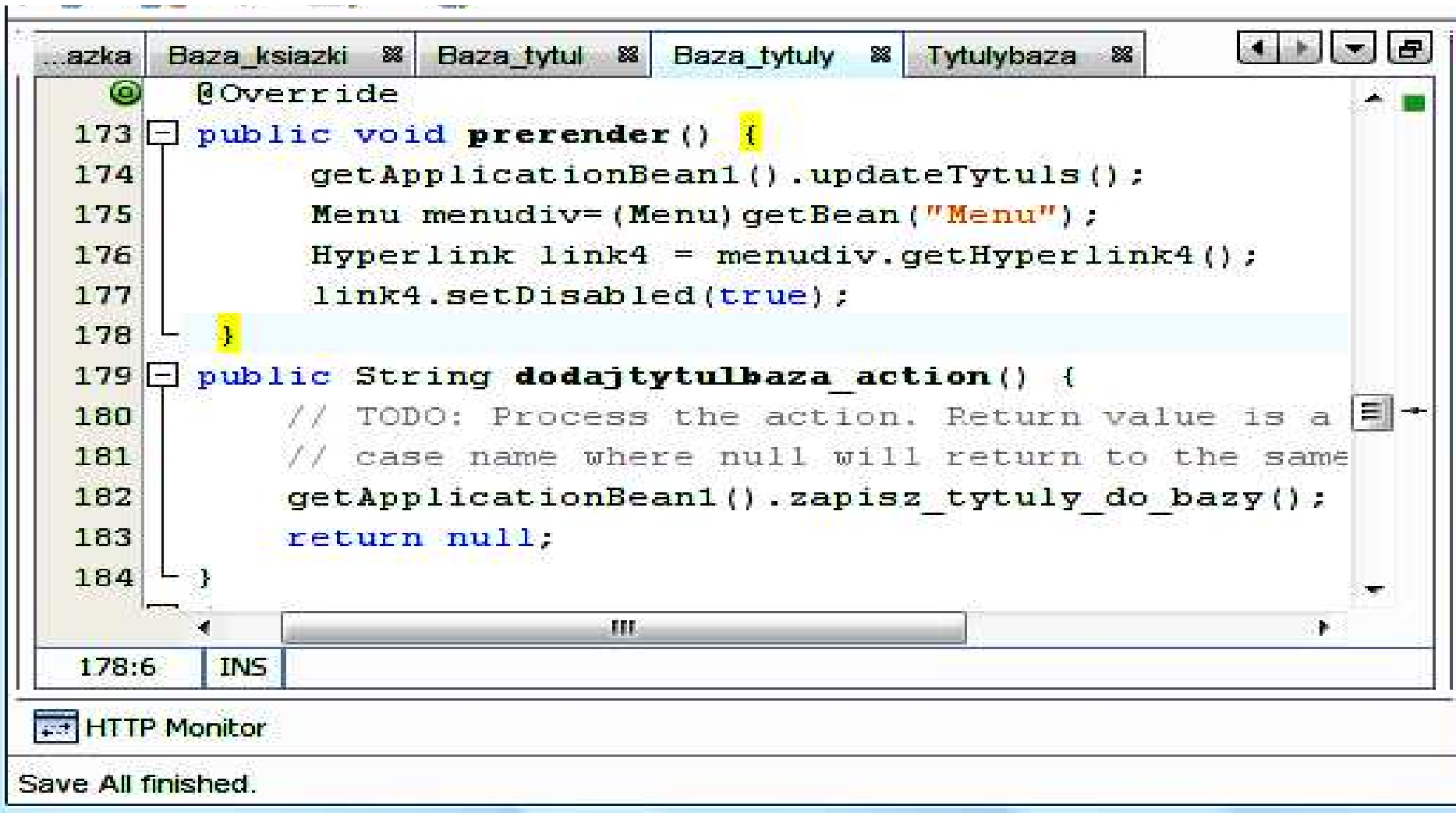
Definicje metod w klasie ApplicationBean1: odczytującą dane typu kolekcja obiektów TTytul_książki i TTytul_książki_na_kasecie z bazy danych (updateTytuls) oraz metoda zapisz_tytul_do_bazy zapisująca do bazy danych (addTTytul_książkis) i warstwy biznesowej (dodaj_tytul) pojedynczych danych typu TTytul_książki i TTytul_książki_na_kasecie oraz odświeżająca zawartość komponentu typu DropDownList na stronie Tytulyaplikacja typu JSPF (przygotujtytuly)

```
..t.java  SessionBean1.java  ApplicationBean1.java  Page1  Logo  Tytuly  Tytulyaplikacja...
89  private TTytul_książki tytuly[];
90  public TTytul_książki[] getTytuly() {
91      return tytuly;
92  }
93  public void setTytuly(TTytul_książki[] tytuly) {
94      this.tytuly = tytuly;
95  }
96  public void updateTytuls() {
97      TTytul_książkiController tytulController = new TTytul_książkiController();
98      tytuly = tytulController.getTTytul_książkis();
99  }
100 public void zapisz_tytul_do_bazy(String dane[]) {
101     // Add the new Entity to the database using UserController
102     TTytul_książkiController Tytul_książkiController =
103         new TTytul_książkiController();
104     TTytul_książki tytul=getAplikacja().dodaj_tytul(dane);//przypadek użycia ""
105     if (tytul!=null)
106     { przygotujtytuly();//wyswietlenie kolek
107       Tytul_książkiController.addTTytul_książkis(tytul); }
108 }
```

105:22 INS

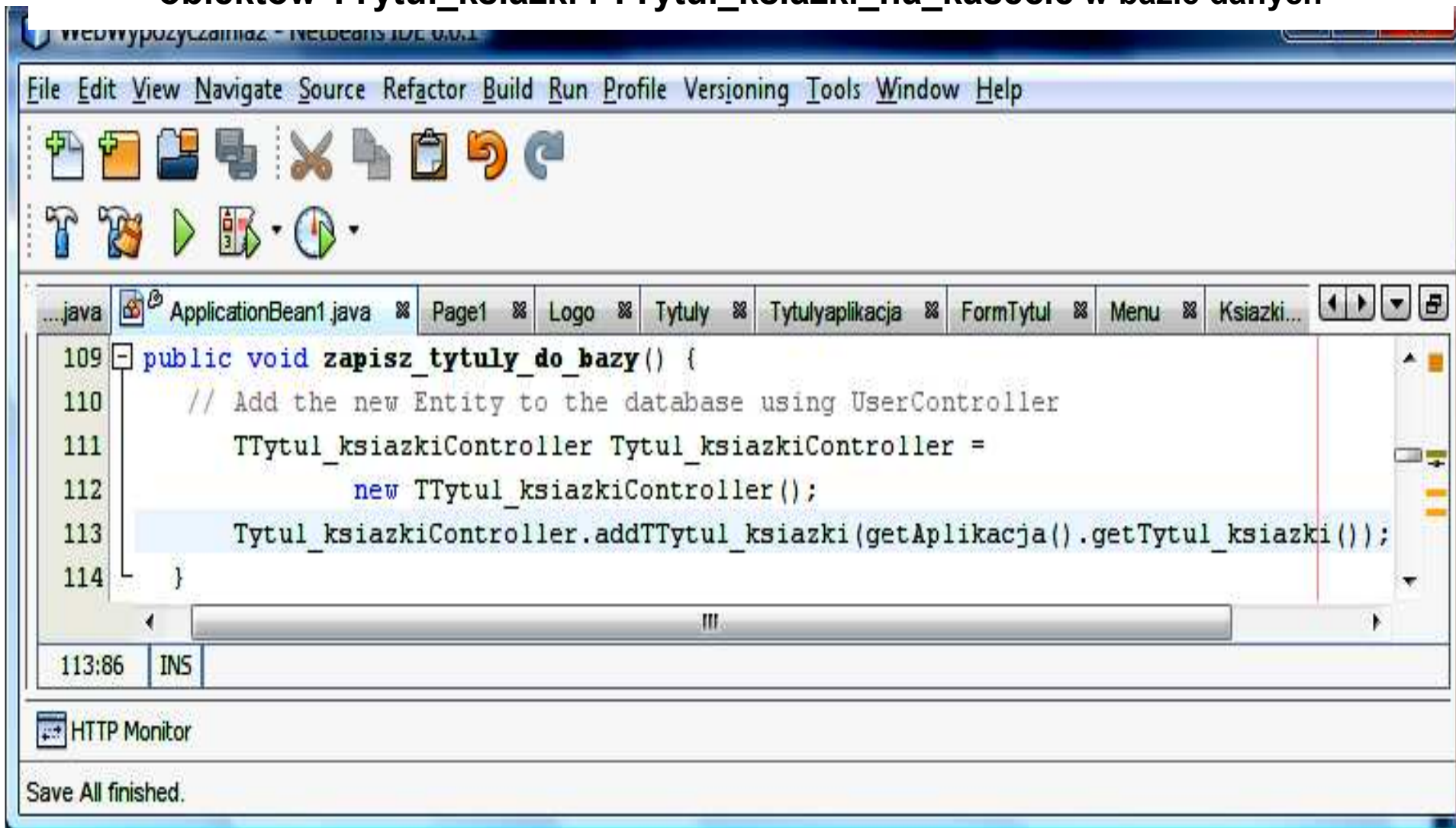
5.4. Oprogramowanie dotyczące formularza Baza_tytuly.jsp

Definicje metod w klasie Baza_tytuly dla strony typu JSP – do zapisu tytułów z warstwy biznesowej (obsługa zdarzenia dodajtytulbaza_action) oraz generowania widoku w fazie Response przetwarzania strony metoda prerender (wygaszanie linku do bieżącej strony w formularzu Menu typu JSPF i aktualizacja tablicy tytuły metodą updateTytuls w klasie ApplicationBean1 wyświetlanej w komponencie Table strony Tytulybaza typu JSPF)



```
...azka | Baza_książki | Baza_tytul | Baza_tytuly | Tytulybaza
@Override
173 public void prerender() {
174     getApplicationBean1().updateTytuls();
175     Menu menudiv = (Menu) getBean("Menu");
176     Hyperlink link4 = menudiv.getHyperlink4();
177     link4.setDisabled(true);
178 }
179 public String dodajtytulbaza_action() {
180     // TODO: Process the action. Return value is a
181     // case name where null will return to the same
182     getApplicationBean1().zapisz_tytuly_do_bazy();
183     return null;
184 }
178:6 | INS
HTTP Monitor
Save All finished.
```

Definicje metody w klasie ApplicationBean1 związanej z zapisem danych typu kolekcja obiektów TTytul_książki i TTytul_książki_na_kasce w bazie danych



The screenshot shows the NetBeans IDE interface with the following components:

- Menu Bar:** File, Edit, View, Navigate, Source, Refactor, Build, Run, Profile, Versioning, Tools, Window, Help.
- Toolbar:** Contains icons for file operations (new, open, save, delete, copy, paste), development (run, debug), and navigation (undo, redo).
- Project Explorer:** Shows a project structure with files: ApplicationBean1.java, Page1, Logo, Tytuly, Tytulyaplikacja, FormTytul, Menu, and Książki...
- Source Editor:** Displays the following Java code:

```
109 public void zapisz_tytuly_do_bazy() {  
110     // Add the new Entity to the database using UserController  
111     TTytul_książkiController Tytul_książkiController =  
112         new TTytul_książkiController();  
113     Tytul_książkiController.addTTytul_książki(getAplikacja().getTytul_książki());  
114 }
```
- Status Bar:** Shows the cursor position at line 113, column 86, with the text "INS".
- HTTP Monitor:** A panel at the bottom left, currently empty.
- Message Area:** Displays the message "Save All finished."

Definicja nowej metody `addTTYtul_książki` w klasie `TTYtul_książkiController` zapisującej kolekcję obiektów typu `TTYtul_książki` i `TTYtul_książki_na_kasie` do bazy danych – wersja 1

`id = new Long(-1)` w konstruktorze encji `TTYtul_książki`

```
...java  TTYtul_książkiController.java  Account.java  SessionBean1.java  Applicati...
65 public boolean addTTYtul_książki(ArrayList<TTYtul_książki> tytuly) {
66     EntityManager em = getEntityManager();
67     try {
68         Iterator it = tytuly.iterator();
69         while (it.hasNext()) {
70             TTYtul_książki newTTYtul_książki = (TTYtul_książki) it.next();
71             if (!findTTYtul_książkis(newTTYtul_książki))
72                 { em.getTransaction().begin();
73                   em.persist(newTTYtul_książki);
74                   em.getTransaction().commit();}
75         }
76     } finally {
77         em.close();
78         return false; }
79 }
```

HTTP Monitor

Save All finished.

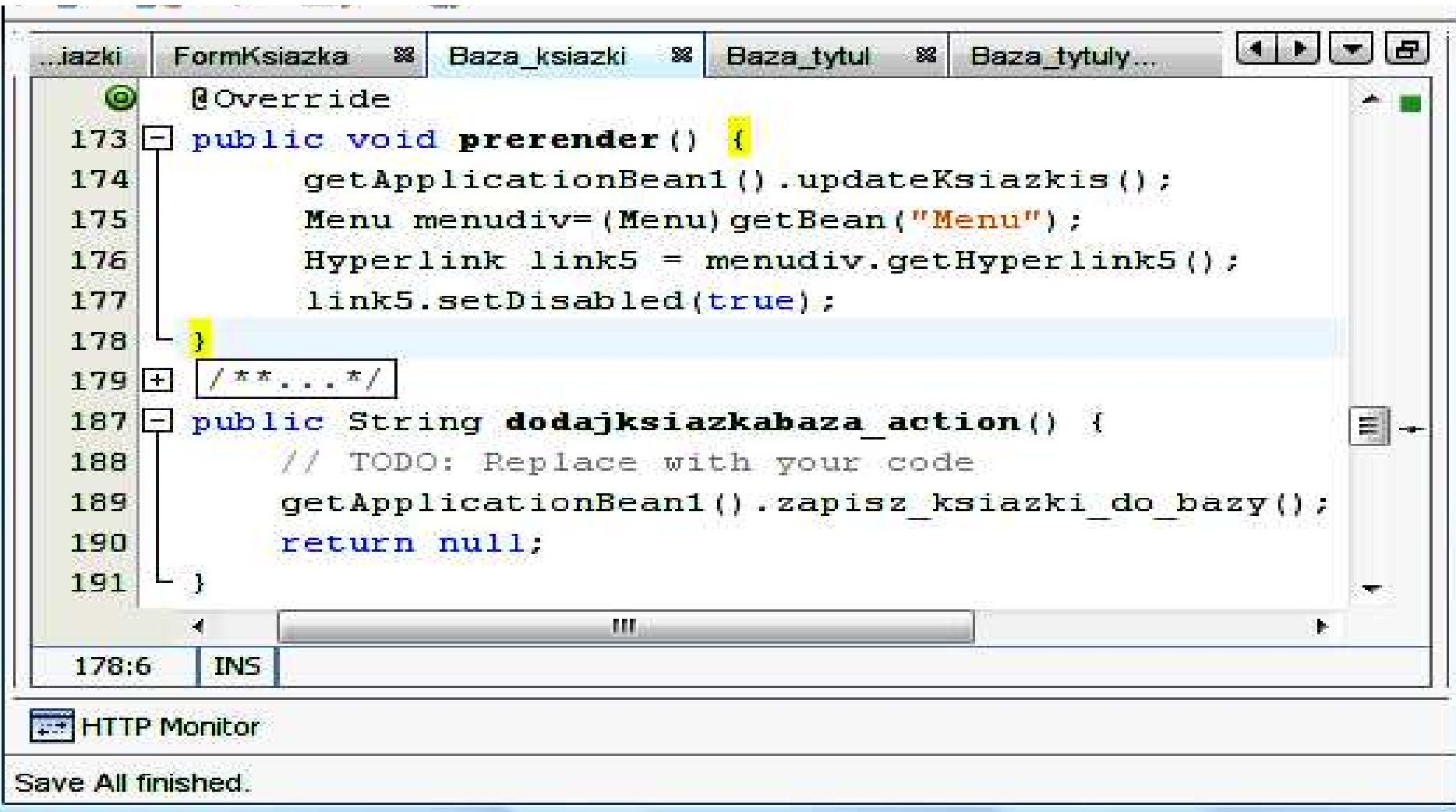
Definicja nowej metody addTTYtul_książki w klasie TTYtul_książkiController zapisującej kolekcję obiektów typu TTYtul_książki i TTYtul_książki_na_kasecie do bazy danych - wersja 2 (id = null w konstruktorze encji TTYtul_książki)



```
...a * | web.xml | Tytuly | TTYtul_książkiController.java | TApplikacja.java | b... |
| [Icons] |
- public boolean addTTYtul_książki (ArrayList<TTYtul_książki> tytuly) {
    EntityManager em = getEntityManager();
    TTYtul_książki newTTYtul_książki=null;
    try {
        Iterator it = tytuly.iterator();
        em.getTransaction().begin();
        while (it.hasNext()) {
            newTTYtul_książki = (TTYtul_książki) it.next();
            if (newTTYtul_książki.getId()==null)
                em.merge(newTTYtul_książki);
        }
        em.persist(newTTYtul_książki);
        em.getTransaction().commit();
    } finally {
        em.close();
        return false; }
}
```


5.5. Oprogramowanie dotyczące formularza Baza_książki.jsp

Definicje metod w klasie Baza_książki dla strony typu JSP – do zapisu książek z warstwy biznesowej (obsługa zdarzenia dodajksiążkabaza_action) oraz generowania widoku w fazie Response przetwarzania strony metoda prerender (wygaszanie linku do bieżącej strony w formularzu Menu typu JSPF i aktualizacja tablicy książki metodą updateKsiążkis w klasie ApplicationBean1 wyświetlanej w komponencie Table strony Książkibaza typu JSPF)



```
...iazki | FormKsiążka | Baza_książki | Baza_tytul | Baza_tytuly...
@Override
173 public void prerender() {
174     getApplicationBean1().updateKsiążkis();
175     Menu menudiv=(Menu) getBean("Menu");
176     Hyperlink link5 = menudiv.getHyperlink5();
177     link5.setDisabled(true);
178 }
179 /**...*/
187 public String dodajksiążkabaza_action() {
188     // TODO: Replace with your code
189     getApplicationBean1().zapisz_książki_do_bazy();
190     return null;
191 }
178:6 | INS
HTTP Monitor
Save All finished.
```

Definicje metod w klasie ApplicationBean1 związanych z zapisem (zapisz_książki_do_bazy) i odczytem (updateKsiążkis) danych typu kolekcja obiektów TEgzemplarz i TEgzemplarz_termin w bazie danych



```
...java ApplicationBean1.java Page1 Logo Tytuly Tytulyaplikacja FormTytul Menu...
186 private TEgzemplarz ksiazki[];
187 public TEgzemplarz[] getKsiazki() {
188     return ksiazki;
189 }
190 public void setKsiazki(TEgzemplarz[] ksiazki) {
191     this.ksiazki = ksiazki;
192 }
193 public void updateKsiazkis() {
194     TEgzemplarzController egzController = new TEgzemplarzController();
195     ksiazki = egzController.getTEgzemplarz();
196 }
197 public void zapisz_ksiazki_do_bazy(){
198     // Add the new Entity to the database using UserController
199     TEgzemplarzController egzemplarzController =
200         new TEgzemplarzController();
201     egzemplarzController.addTEgzemplarz(getAplikacja().getTytul_ksiazki());
202 }
```

186:3 INS

HTTP Monitor

Definicja nowej metody addTEgzemplarze w klasie TEgzemplarzController zapisującej kolekcję obiektów typu TEgzemplarz i TEgzemplarz_termin do bazy danych - wersja 1
id = new Long(-1) w konstruktorze encji TEgzemplarz

```
56 public boolean addTEgzemplarze (ArrayList<TTytul_książki> tytuly) {
57     EntityManager em = getEntityManager();
58     Iterator it = tytuly.iterator();
59     try {
60         while (it.hasNext()) {
61             TTytul_książki newTTytul_książki = (TTytul_książki) it.next();
62             Iterator it_ = newTTytul_książki.getMKsiążka().iterator();
63             while (it_.hasNext()) {
64                 TEgzemplarz newTEgzemplarz = (TEgzemplarz) it_.next();
65                 if (!findTEgzemplarzs(newTEgzemplarz))
66                 { em.getTransaction().begin();
67                   em.persist(newTEgzemplarz);
68                   em.getTransaction().commit();}
69             }
70         }
71     } finally {
72         em.close();
73         return false;
74     }
75 }
```

14:1

INS

HTTP Monitor

**Definicja nowej metody addTEgzemplarze w klasie TEgzemplarzController zapisującej kolekcję obiektów typu TEgzemplarz i TEgzemplarz_termin do bazy danych – wersja 2
id = null w konstruktorze encji TEgzemplarz**

```
...ava | TEGzemplarController.java * | TEGzemplar.java | TTYtul_ksiazki.java | TTYt...
public boolean addTEgzemplarze(ArrayList<TTYtul_ksiazki> tytuly) {
    EntityManager em = getEntityManager();
    TEgzemplarz newTEgzemplarz = null;
    Iterator it = tytuly.iterator();
    em.getTransaction().begin();
    try {
        while (it.hasNext()) {
            TTYtul_ksiazki newTTYtul_ksiazki = (TTYtul_ksiazki) it.next();
            if(newTTYtul_ksiazki.getId()==null) continue;
            Iterator it_ = newTTYtul_ksiazki.getMKsiazka().iterator();
            while (it_.hasNext()) {
                newTEgzemplarz = (TEgzemplarz) it_.next();
                if (newTEgzemplarz.getId()==null)
                    em.merge(newTEgzemplarz); }
            }
            em.persist(newTEgzemplarz);
            em.getTransaction().commit();
        } finally {
            em.close();
            return false; }
    }
```

5.6. Zsynchronizowanie zawartości bazy danych i warstwy biznesowej przy starcie aplikacji – zastosowanie metod update.. w obiekcie ApplicationBean1

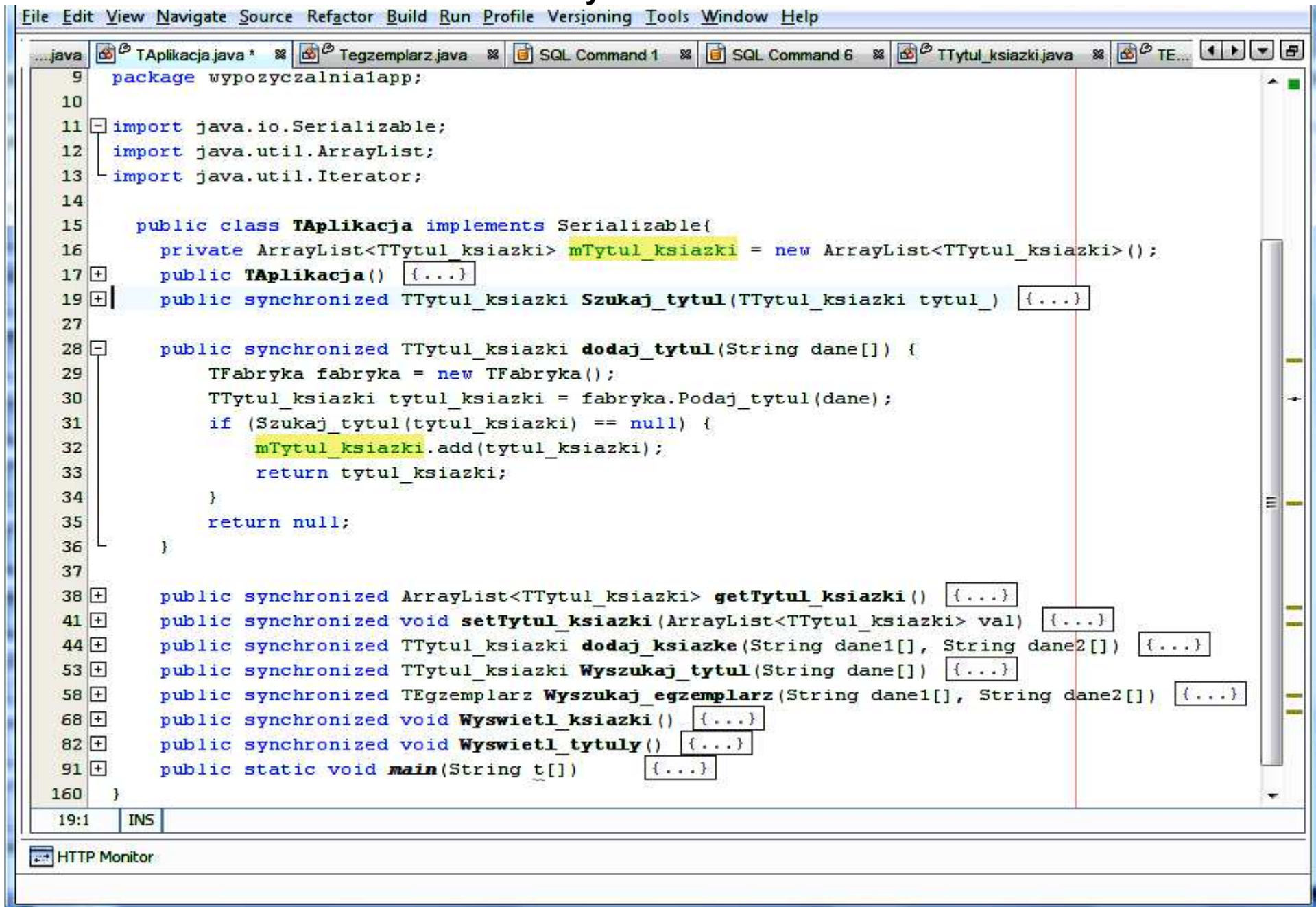
```
@Override
public void init() { // Perform initializations inherited from
    super.init();
    // Perform application initialization that must complete
    // *before* managed components are initialized
    // TODO - add your own initialization code here
    Managed Component Initialization
    // *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 < tytulj.length; i++)
        aplikacja.getTytul_ksiazki().add(tytulj[i]);
    Iterator it = aplikacja.getTytul_ksiazki().iterator();
    while (it.hasNext()) {
        T Tytul_ksiazki tytul = (T Tytul_ksiazki) it.next();
        for (int j = 0; j < ksiazki.length; j++) {
            T Tytul_ksiazki tytul1 = ksiazki[j].getMTytul_ksiazki();
            if (tytul1 != null)
                if (tytul1.equals(tytul))
                    { tytul.getMKsiazka().add(ksiazki[j]); }
        }
    }
}
```

5.7. Przystosowanie do pracy z wieloma wątkami warstwy biznesowej – metody typu synchronized



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

19:1 INS

HTTP Monitor