

Projekt INP002017

Instrukcja 2

Autor

Dr inż. Zofia Kruczkiewicz

I. Czynności wykonane zgodnie z harmonogramem grupy w tygodniach 1-15

Tabela 2. Przebieg realizacji każdego z projektów (tabela 1)

Opis realizacji projektu					
Nr tygodni Daty	Termin przekazania prac	Spotkanie	Uwagi dotyczące realizacji zadań przez każdą z dwóch podgrup zespołu	Ocena częściowa	Zadania Scrum Master
1-tydzień 23.02.18- 2.03.18	Wynik prac należy wysłać za pomocą poczty elektronicznej do dnia 7.03.2018.	Sprint planning meeting (45 min)	<ul style="list-style-type: none"> Zajęcia organizacyjne (podział na grupy i podgrupy, przydzielenie ról projektowych, uzyskanie dostępu do wymaganych narzędzi) User Stories - Opracowanie modelu biznesowego „świata rzeczywistego” systemu Sprint Backlog (formy pośrednie: Product Backlog), Sprint planining) - zdefiniowanie wymagań funkcjonalnych i нефункциональных wynikających z otrzymanego fragmentu opisu „świata rzeczywistego” w postaci diagramu wymagań wg instrukcji: 		Współdziałanie z wykonawcami
2-tydzień 2.03.18- 9.03.18	Wynik prac zostanie oceniony przez prowadzącego zajęcia. Uwaga: należy wysłać wersję końcową projektu UML z ewentualnie uzupełnionym diagramem wymagań		<p>Instrukcja 3 - definiowanie wymagań</p> <p>Podczas spotkania 9.03.2018 należy zaprezentować wyniki prac, czyli „Opis świata rzeczywistego” i diagram wymagań oprogramowania.</p>		

<p>3- tydzień 4- tydzień 9.03.2018- 23.03.2018</p>	<p>Konieczne wystanie projektu UML z dodanym diagramem przypadków użycia do dnia 19.03.18</p> <p>Diagram klas (wersja początkowa) powinien być dostarczony do dnia 22.03.18</p>	<p>Sprint planning meeting (45 min)</p>	<p>Należy wykonać:</p> <p>1) diagram przypadków użycia wg instrukcji: Instrukcja 4 - specyfikacja wymagań za pomocą diagramu przypadków użycia. ze specyfikacją poszczególnych przypadków użycia</p> <p>2) Diagramy klas i sekwencji wg instrukcji do lab5-7: Instrukcja - część 1 Instrukcja - część 2 Diagram klas UML zostanie przedstawiony na wykładzie w dniu 20.03.2018.</p>		<p>Eliminacja redundancji w projekcie. Udział w pracach projektowych</p>
<p>5- tydzień 6- tydzień 23.03.2018- 6.04.2018</p>	<p>Projekt UML oraz projekt z kodem Javy należy wysłać za pomocą poczty elektronicznej do dnia 2.04.2018.</p>	<p>Sprint planning meeting (45 min)</p>	<p>1) Projektowanie i implementacja oprogramowania w oparciu o wzorce projektowe – diagramy UML: rozwijanie diagramu klas, diagramy sekwencji i aktywności wybranych 3 prostych przypadków użycia reprezentujących projekt warstwy biznesowej projektu oraz implementacja projektu na platformie Java SE. Diagramy klas i sekwencji wg instrukcji do lab5-7: Instrukcja - część 1 Instrukcja - część 2</p>		<p>Eliminacja redundancji w projekcie. Udział w pracach projektowych i implementacji</p>

<p>7- tydzień 8- tydzień 6.04.2018- 20.04.2018</p>	<p>Kod programu w wersji SE i EE należy wysłać do dnia 16.04.18</p>	<p>Sprint planning meeting (45 min)</p>	<ol style="list-style-type: none"> 1) Tworzenie interfejsu graficznego użytkownika na platformie Java SE do programu zawierającego implementację warstwy biznesowej 2) Przekształcenie aplikacji na aplikację działającą na platformie Java EE, zawierającego warstwę klienta typu desktopowego 3) Instrukcja 1 - opisująca przebieg prac podczas 1- 8 tygodnia 	<p>Eliminacja redundancji w projekcie. Udział w pracach projektowych i implementacji</p>
---	---	---	--	--

<p>9-tydzień 10-tydzień 11- tydzień 12- tydzień</p> <p>20.04.2018- 18.05.2018</p>	<p>Kod programu wg 1 należy wysłać do dnia 4.05.2018, drugi wg p.2 do 14.05.2018.</p>	<p>Sprint planning meeting (45 min)</p>	<p>1) Rozwijanie oprogramowania: projektowanie i implementacja oprogramowania w oparciu o wzorce projektowe – diagramy UML: rozwijanie diagramu klas, diagramy sekwencji i aktywności wybranych 2 (minimum) złożonych przypadków użycia reprezentujących projekt warstwy biznesowej projektu oraz implementacja projektu na platformie Java EE</p> <p>2) Rozwijanie aplikacji desktopowej umożliwiającej dostęp do dodanej logiki biznesowej, działającej na platformie Java EE</p> <p><u>Uwaga: działania wg instrukcji dodanej do tego etapu projektu</u></p> <p>1) Sprawdzanie poprawności oprogramowania wg <u>Instrukcji do lab 10</u> oraz <u>Instrukcji do lab 11</u>.</p>		<p>Eliminacja redundancji w projekcie. Udział w pracach projektowych i implementacji</p>
<p>13- tydzień 14- tydzień 15- tydzień</p> <p>18.05.2018- 8.06.2018</p>	<p>Kod programu należy wysłać do dnia 1.05.2018, drugi wg p.2 do 4.06.2018</p>	<p>Sprint planning meeting (45 min)</p>	<p>1) Dodanie warstwy integracji z bazą danych opartą na technologii JPA (Java Persistence API) do programu oparta na automatycznym generowaniu kontrolerów typu Session Bean For Entity Class</p> <p>Uwaga: działania wg bieżącej instrukcji do tego etapu projektu</p> <p>1) Uzupełnienie aplikacji desktopowej do wywołanie operacji na bazie danych w technologii ORM.</p> <p>2) Zaliczenie projektu</p>		<p>Eliminacja redundancji w projekcie. Udział w pracach projektowych i implementacji</p> <p>-</p>

II. Kontynuacja programu wykonanego wg instrukcji Projekt_INP002017_1 (wg materiałów do wykładu 7 Wyklad_INP002017_7_2.pdf – p.6-8)

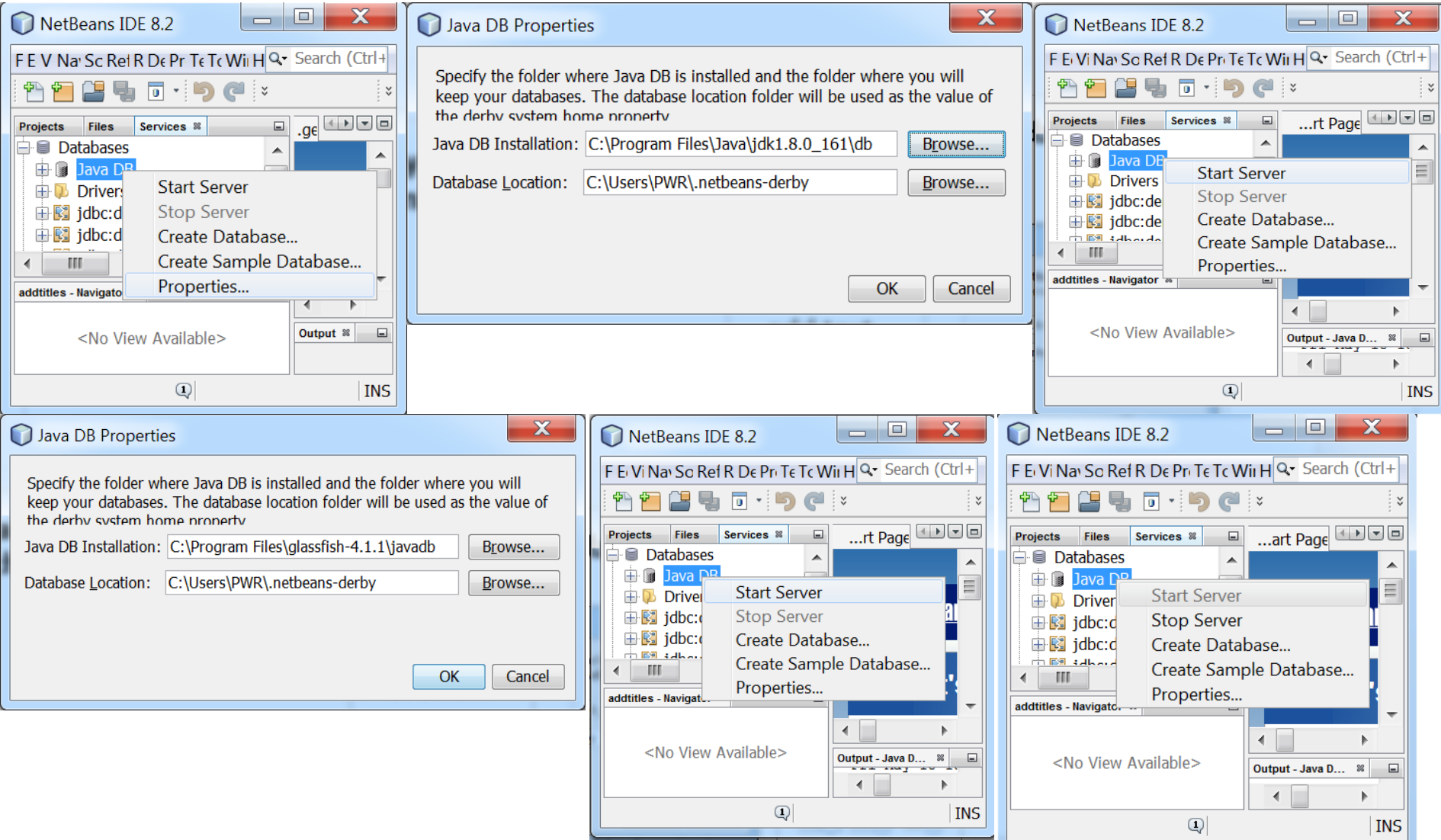
1. **Warstwa zasobów (EIS)** - baza danych w systemie baz danych Derby
2. **Utworzenie obiektowego modelu** danych do utrwalania ORM
3. **Warstwa integracji.** Zastosowanie wzorca projektowego typu *Domain Store* w technologii JPA (Java Persistence) na platformie Java EE

Zagadnienia

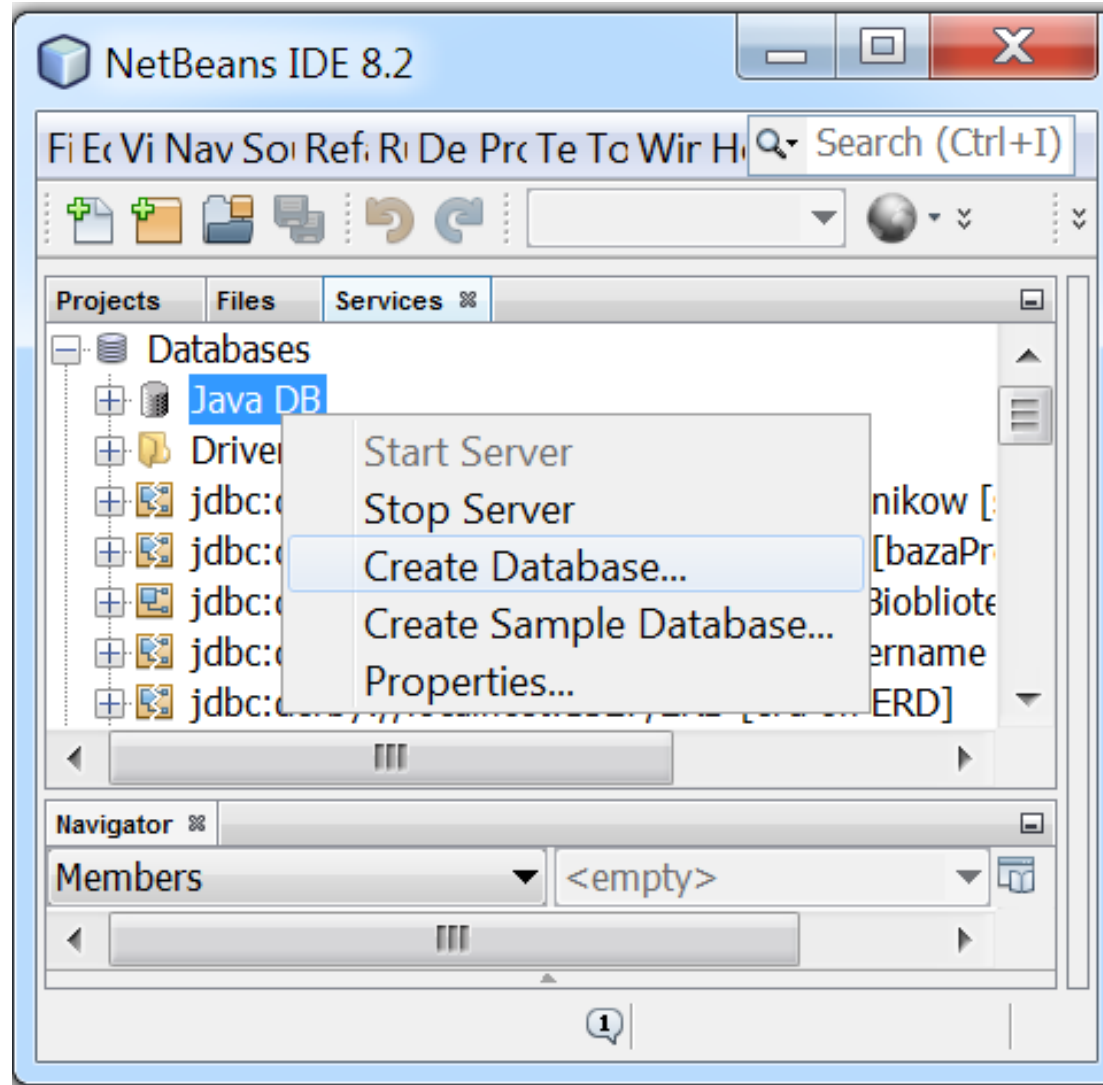
1. **Warstwa zasobów (EIS)- baza danych w systemie baz danych Derby**

Przed założeniem bazy danych należy sprawdzić, czy silnik bazy danych Java DB uruchamia się

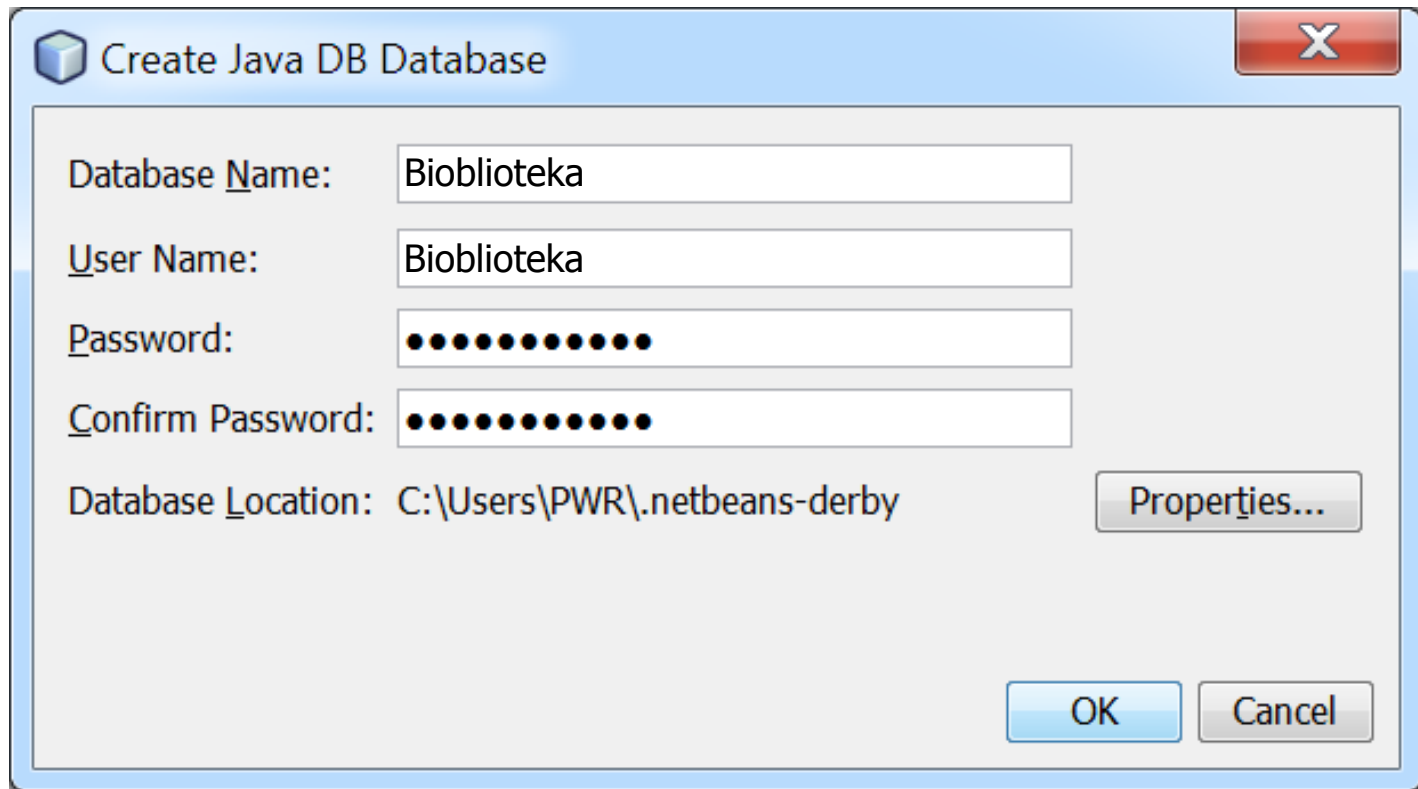
1. Należy w zakładce **Services** rozwinąć pozycję **Databases** i kliknąć na pozycję **Start Server**.
2. Jeżeli server **Java DB** nie wystartuje (przejdźcie do pozycji **Stop Server**), należy wybrać pozycję **Properties**, wybrać pole **Java DB Instalation** i nacisnąć na klawisz **Browse...**. Wtedy można wybrać instalację javadb w **C:\Program Files\glassfish-4.1.1\javadb** i ponowić uruchomienie silnika Java DB. Proba powinna być udana



1. Zakładanie pustej bazy danych dla systemu baz danych Derby



2. Zakładanie pustej bazy danych **Bioblioteka** w systemie baz danych Derby



Create Java DB Database

Database Name:

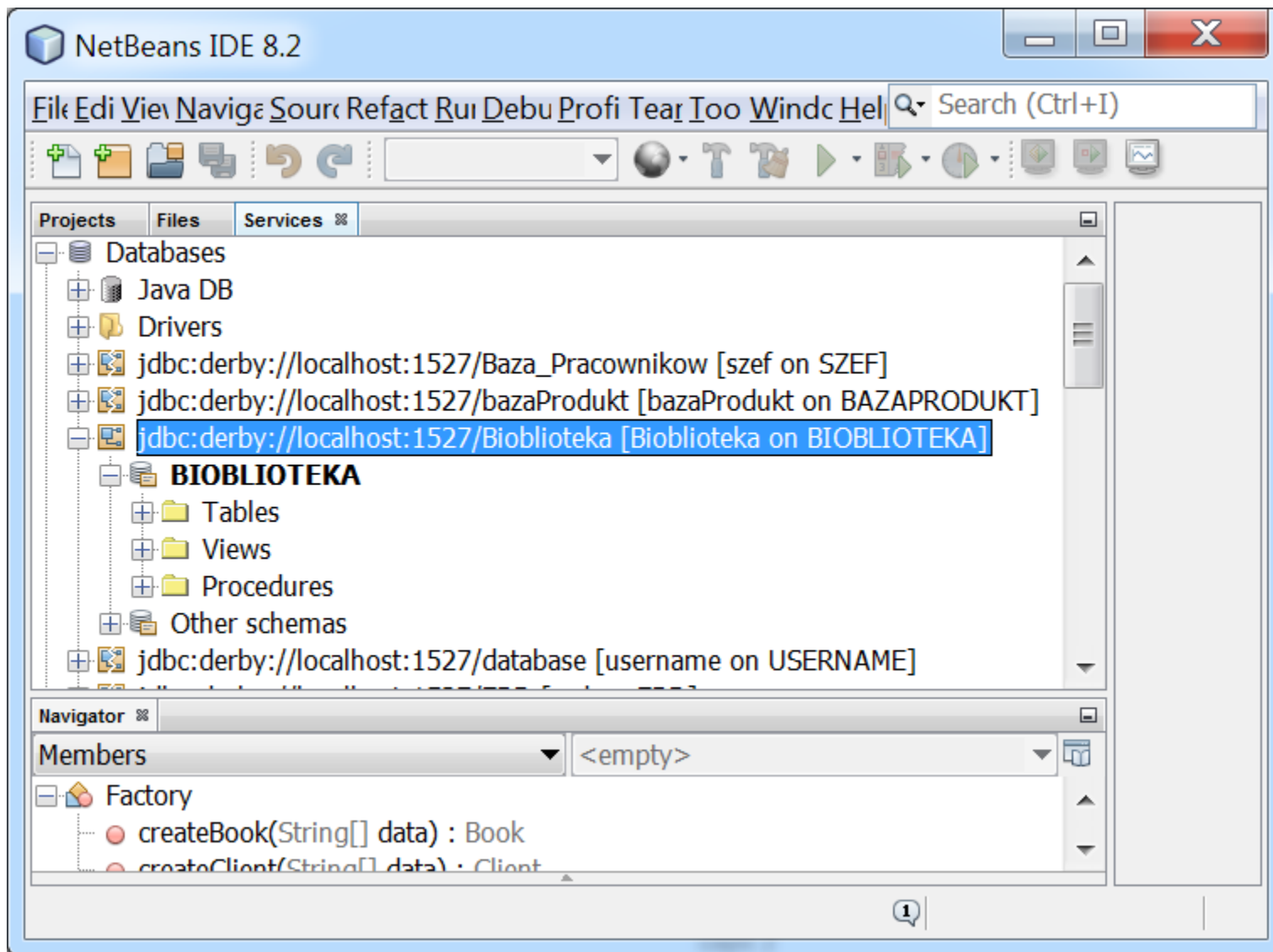
User Name:

Password:

Confirm Password:

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

3. Utworzona pusta baza danych. W celu połączenia z bazą danych należy kliknąć na nazwę bazy danych i wybrać pozycję **Connect...**



Zagadnienia

1. **Warstwa zasobów (EIS)- baza danych w systemie baz danych Derby**
2. **Utworzenie obiektowego modelu danych do utrwalania ORM**

Dodanie biblioteki **EclipseLink** **from GlassFish** do projektu Java SE z logiką biznesową – do folderu Libraries

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

<default confi... [Icons]

Projects | Files | Services

- Library_2IO
 - Source Packages
 - subbusinessstier
 - subbusinessstier.entities
 - Book.java
 - Client.java
 - Reservation.java
 - TitleBook.java
 - TitleBookRead.java
 - Test Packages
 - Libraries
 - EclipseLink from GlassFish - org.eclipse.persistence.ar
 - EclipseLink from GlassFish - org.eclipse.persistence.as
 - EclipseLink from GlassFish - org.eclipse.persistence.cc
 - EclipseLink from GlassFish - org.eclipse.persistence.dk
 - EclipseLink from GlassFish - org.eclipse.persistence.jp
 - EclipseLink from GlassFish - org.eclipse.persistence.jp
 - EclipseLink from GlassFish - org.eclipse.persistence.jp
 - EclipseLink from GlassFish - org.eclipse.persistence.m
 - EclipseLink from GlassFish - org.eclipse.persistence.or
 - EclipseLink from GlassFish - javax.persistence.jar
 - JDK 1.8 (Default)
 - Test Libraries

```

Source | History | [Icons]
13  import javax.persistence.Id;
14  import javax.persistence.OneToMany;
15  import javax.persistence.Transient;
16  import subbusinessstier.Factory;
17
18  @Entity
19  public class TitleBook implements Serializable {
20      private static final long serialVersionUID=1L;
21      private String publisher;
22      private String ISBN;
23      private String title;
24      private String author;
25
26      @Id
27      @GeneratedValue(strategy = GenerationType.AUTO)
28      private Long id;
29      public Long getId() {
30          return id;
31      }
32      public void setId(Long id) {
33          this.id = id;
34      }
35
36      @OneToMany(mappedBy = "titleBook", cascade=ALL)
37      List<Book> books;
38      public List<Book> getBooks() {
39          return books;
40      }
41      public void setBooks(List<Book> books) {
42          this.books = books;
43      }
44      public TitleBook() {
45          books = new ArrayList();
46      }
47  }

```

setBooks - Navigator

Members

- getISBN() : String
- getId() : Long
- getPublisher() : String
- getTitle() : String
- hashCode() : int ↑ Object
- searchBook(Book book) : Book
- searchFreeBook(LocalDate date) : boolean
- setActor(String val)
- setAuthor(String author)
- setBooks(List<Book> books)
- setISBN(String ISBN)
- setId(Long id)

subbusinessstier.entities.TitleBook > setBooks >

Output

Java DB Database Process | GlassFish Server | Library2_Client1_EE (run) | Libra

```
@Entity
```

```
public class TitleBook implements Serializable {  
    private static final long serialVersionUID=1L;  
    private String publisher;  
    private String ISBN;  
    private String title;  
    private String author;  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private Long id;  
    public Long getId() { ...3 lines }  
    public void setId(Long id) { ...3 lines }  
    @OneToMany(mappedBy = "titleBook", cascade=ALL)  
    List<Book> books;  
    public List<Book> getBooks() { ...3 lines }  
    public void setBooks(List<Book> books) { ...3 lines }  
    public TitleBook() { ...3 lines }  
  
    public String getPublisher() { ...3 lines }  
    public void setPublisher(String publisher) { ...3 lines }  
    public String getISBN() { ...3 lines }  
    public void setISBN(String ISBN) { ...3 lines }  
    public String getTitle() { ...3 lines }  
    public void setTitle(String title) { ...3 lines }  
    public String getAuthor() { ...3 lines }  
    public void setAuthor(String author) { ...3 lines }  
    public String getActor() { ...3 lines }  
    public void setActor(String val) { ...2 lines }
```

Zmiana typu klas danych na typ @Entity w projekcie warstwy biznesowej – dodano adnotacje, nowy atrybut Id. Należy zestandaryzować nazwy metod dostępu do składowych klasy typu Entity.

```
package subbusinessstier.entities;
```

```
+ import ...
```

```
@Entity
```

```
public class TitleBookRead extends TitleBook {
```

```
private static final long serialVersionUID=1L;
```

```
private String actor;
```

```
@Override
```

```
+ public String getActor() {...3 lines }
```

```
@Override
```

```
+ public void setActor(String actor) {...3 lines }
```

```
@Override
```

```
+ public String toString() {...5 lines }
```

```
}
```

```
@Entity
```

```
public class Book implements Serializable {
```

```
private static final long serialVersionUID=1L;
```

```
private int number;
```

```
@ManyToOne
```

```
private TitleBook titleBook;
```

```
// @OneToMany (mappedBy="book")
```

```
@Transient
```

```
private List<Reservation> reservations;
```

```
@Id
```

```
@GeneratedValue(strategy = GenerationType.AUTO)
```

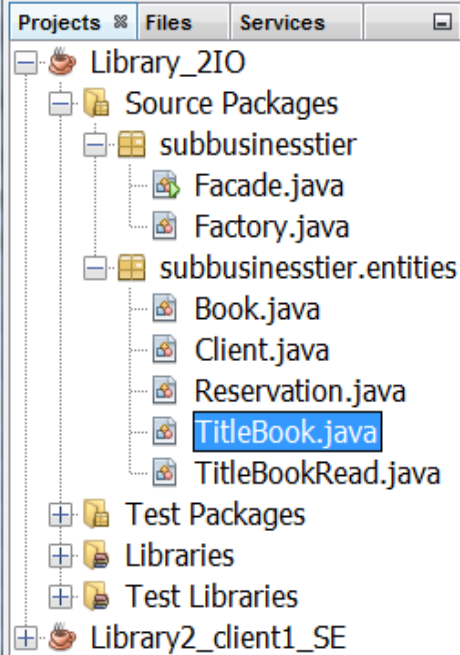
```
private Long id;
```

```
public Long getId() { ...3 lines }
```

```
public void setId(Long id) { ...3 lines }
```

```
public List<Reservation> getReservations() { ...3 lines }
```

```
public void setReservations(List<Reservation> reservations) { ...3 lines }
```

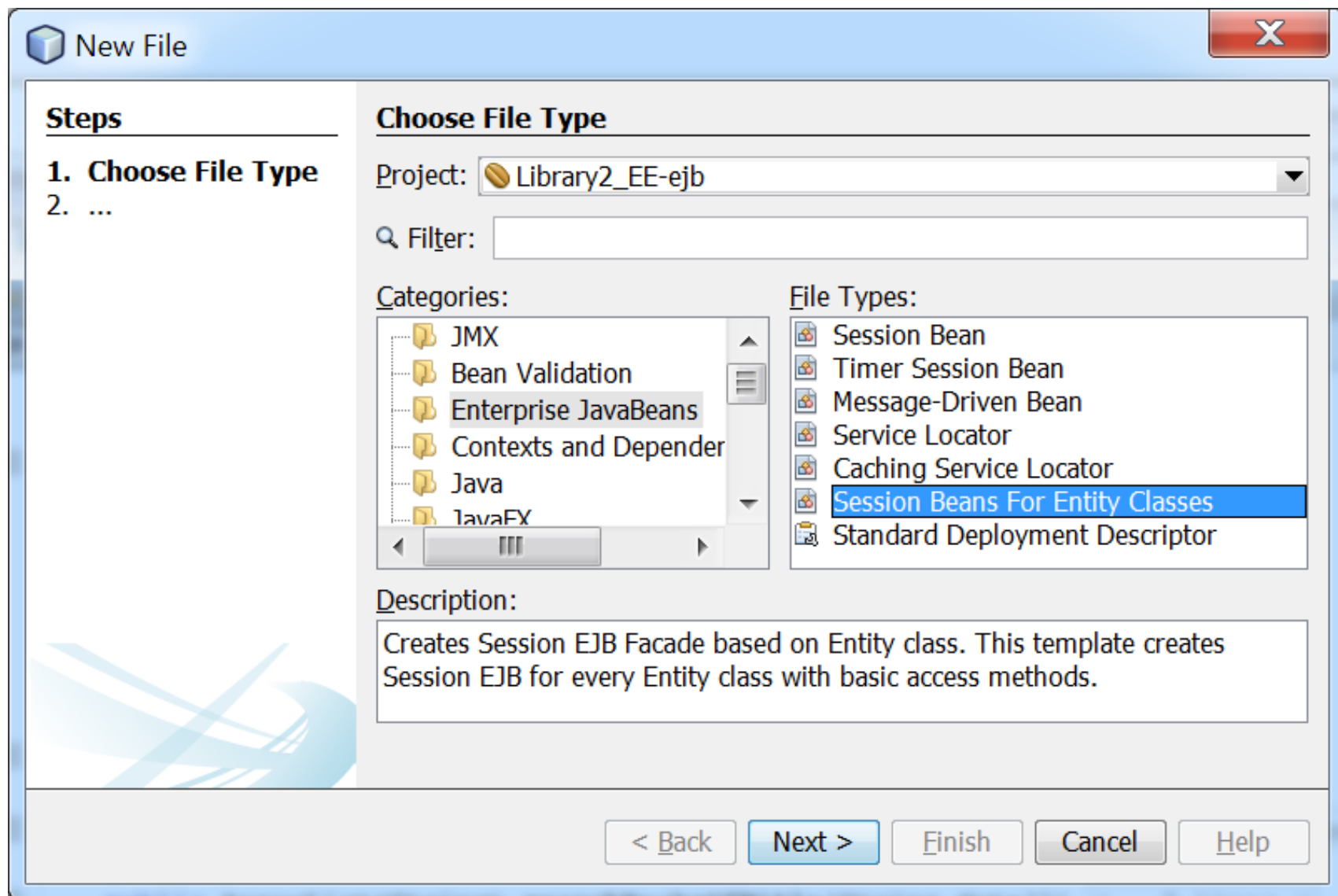



```
...va Book.java Factory.java Facade.java TitleBook.java
Source History
17
   @Entity
   public class TitleBook implements Serializable {
20     private static final long serialVersionUID=1L;
21     private String publisher;
22     private String ISBN;
23     private String title;
24     private String author;
25     @Id
26     @GeneratedValue(strategy = GenerationType.AUTO)
27     private Long id;
28     public Long getId() {...3 lines }
31     public void setId(Long id) {...3 lines }
34     @OneToMany(mappedBy = "titleBook", cascade=ALL)
35     List<Book> books;
36     public List<Book> getBooks() {...3 lines }
39     public void setBooks(List<Book> books) {...3 lin
42     public TitleBook() {...3 lines }
```

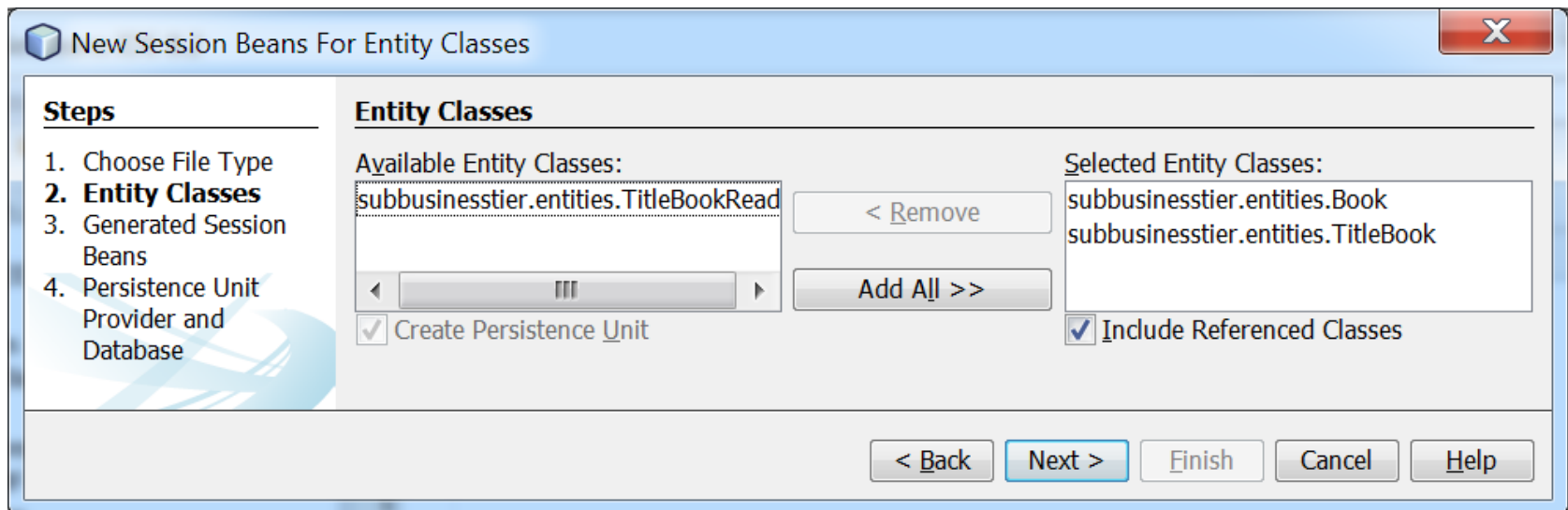
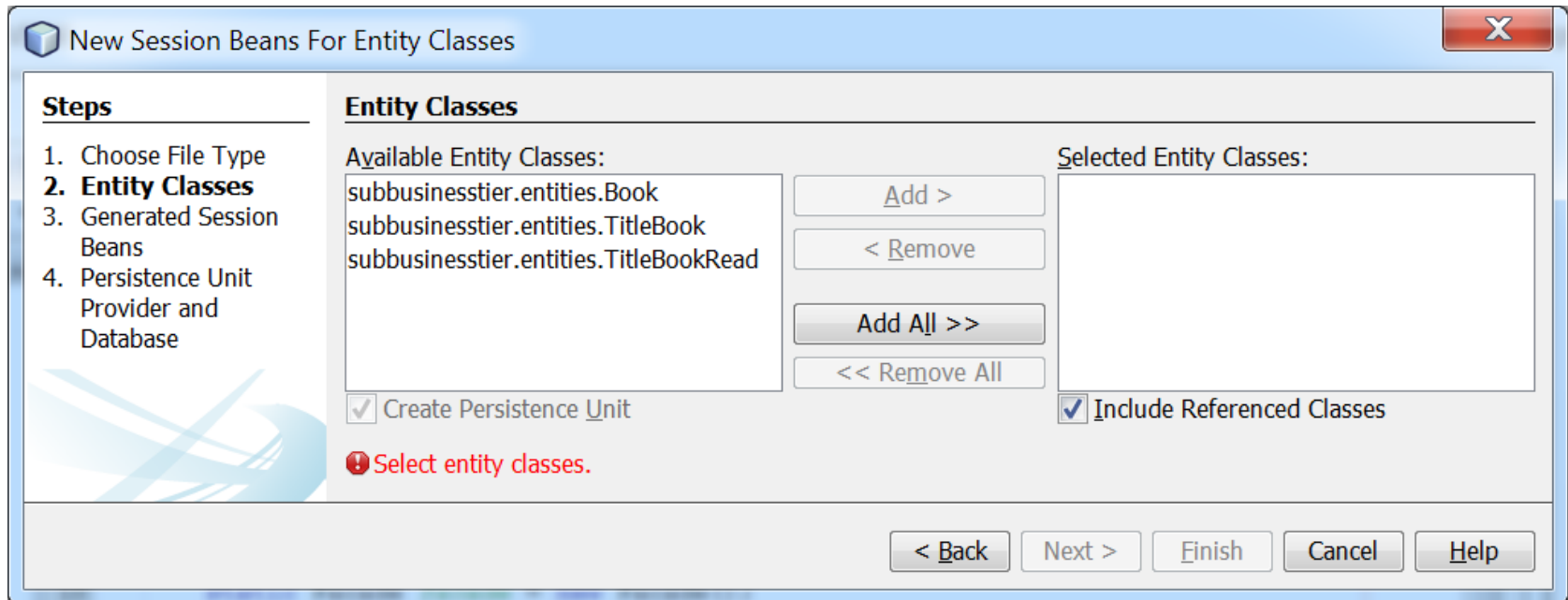
Zagadnienia

1. **Warstwa zasobów (EIS)**- baza danych w systemie baz danych Derby
2. **Utworzenie obiektowego modelu** danych do utrwalania ORM
3. **Warstwa integracji. Zastosowanie wzorca projektowego typu *Domain Store* w technologii JPA (Java Persistence) na platformie Java EE**

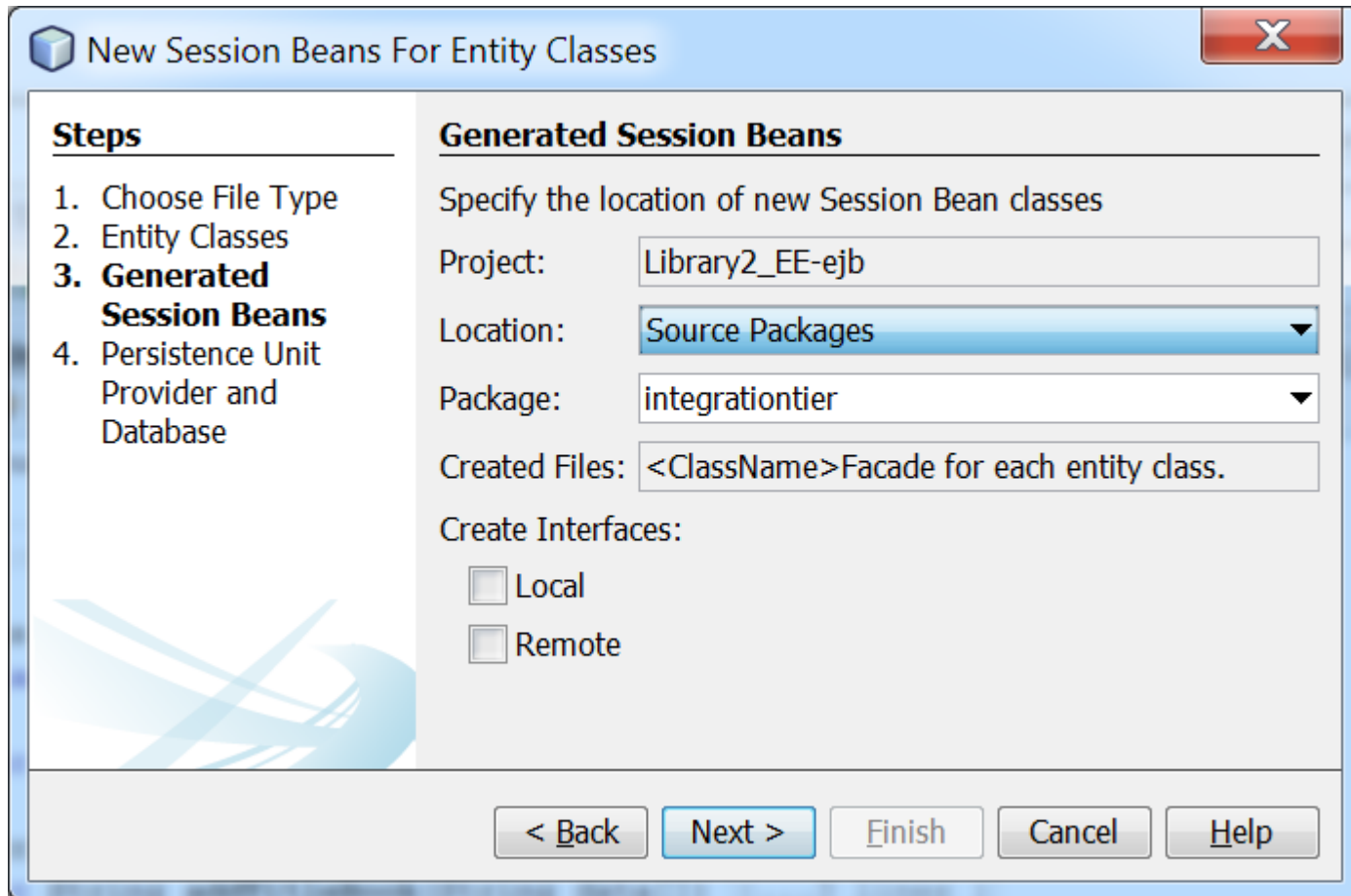
1. Należy w projekcie typu moduł EJB dodać komponenty typu Session Bean for Entity Class - po kliknięciu na nazwę projektu w zakładce Projects wybrać **New/Other./Enterprise JavaBeans/Session Beans For Entity Classes** nacisnąc na klawisz Next



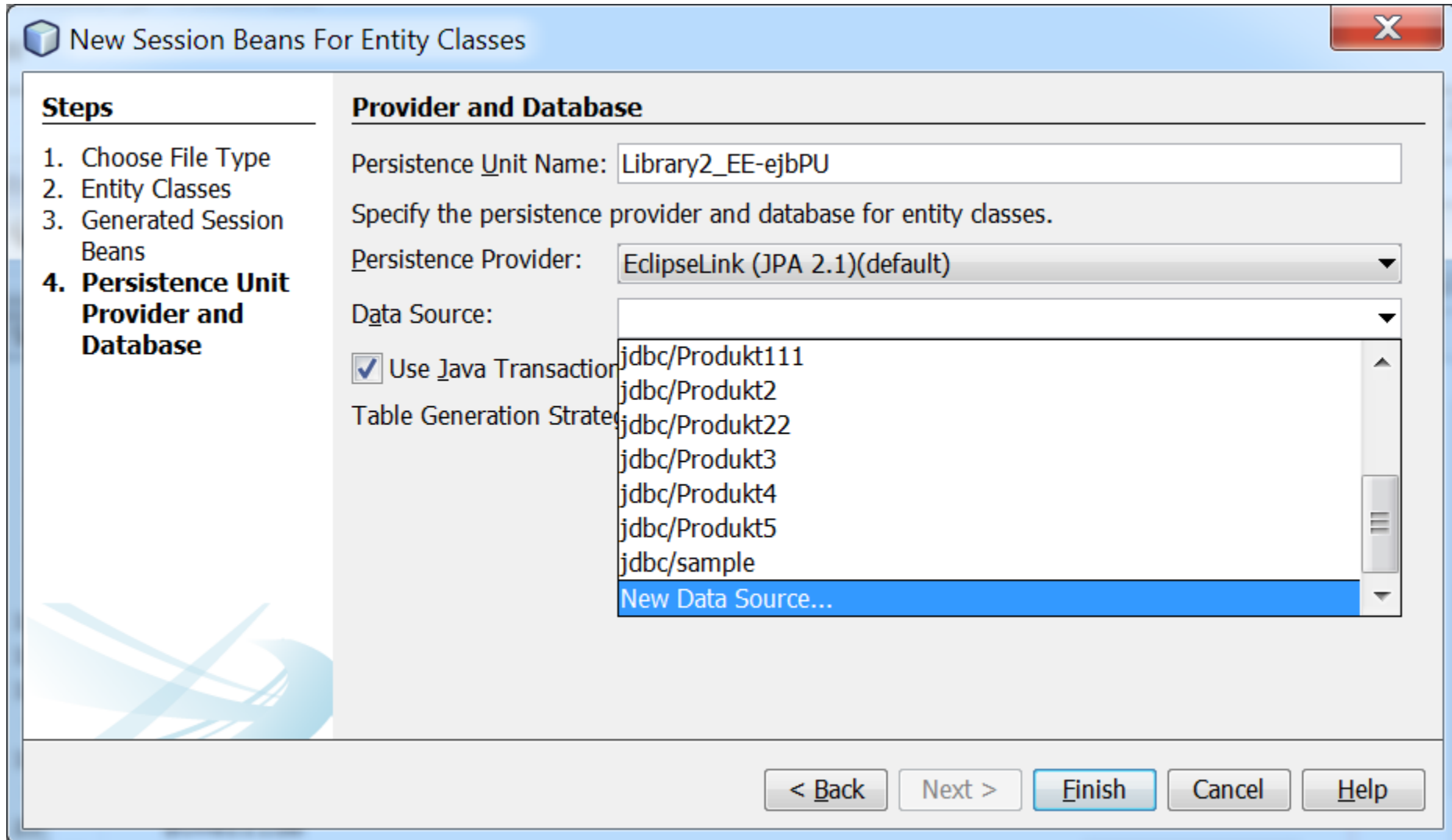
2. Należy wybrać z listy encje **TitleBook** i **Book** i kliknąć na klawisz Next



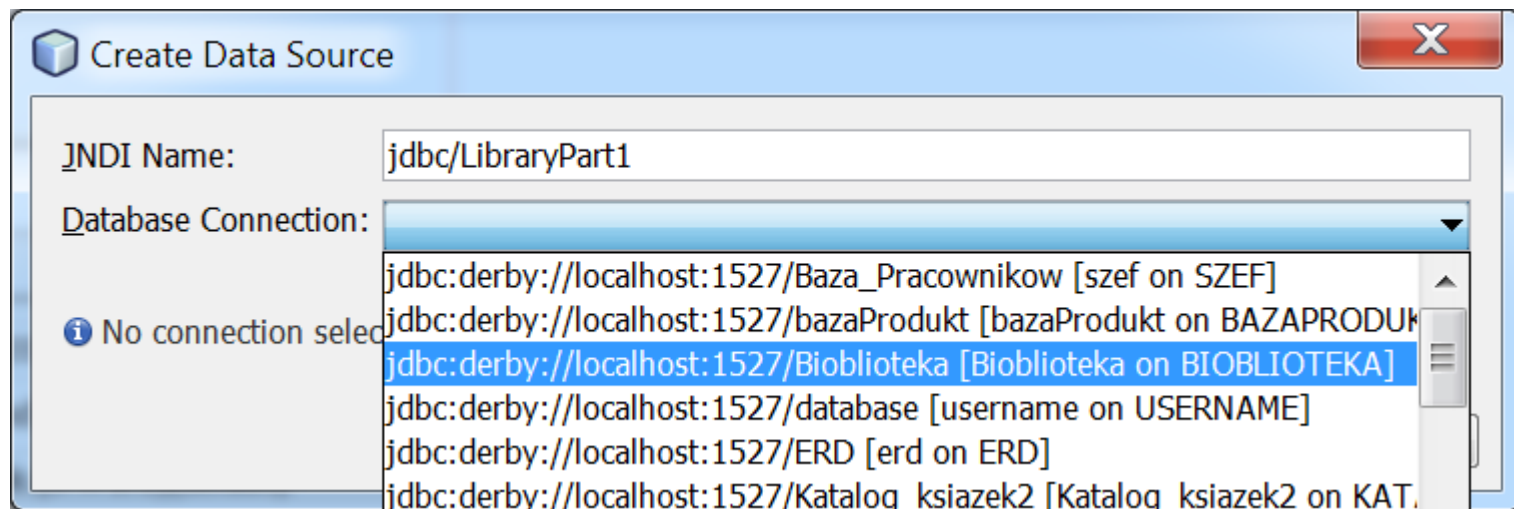
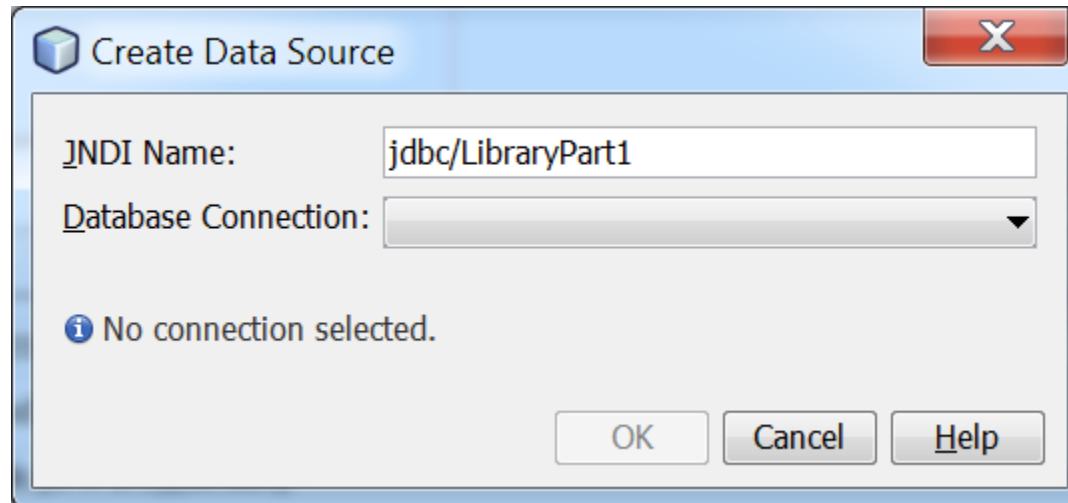
3. W nowym pakiecie np **integrationtier** zostaną wygenerowane komponenty po naciśnięciu na klawisz Next



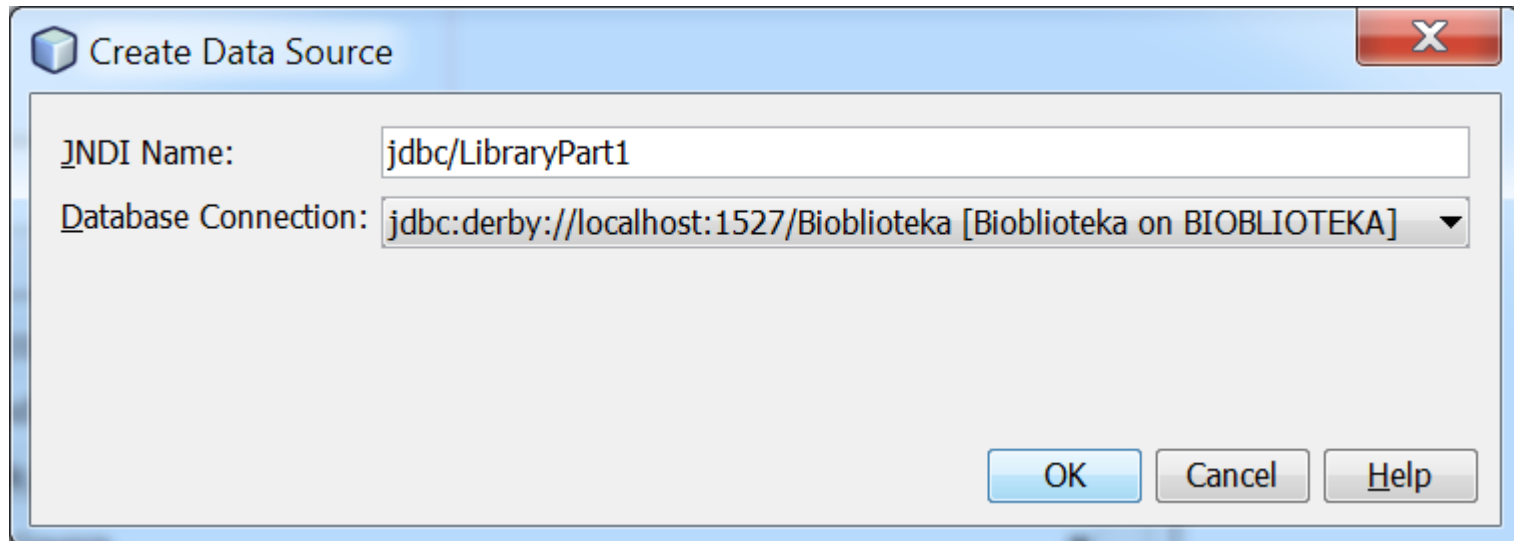
4. W kolejnym kroku należy utworzyć połączenie typu JNDI z bazą danych **Biblioteka** – należy z listy wybrać pozycję **New Data Source...**



5. Wypełnić pole **JNDI Name**, a z listy **Database Connection** należy wybrać bazę danych **Biблиотеka**, założoną wcześniej

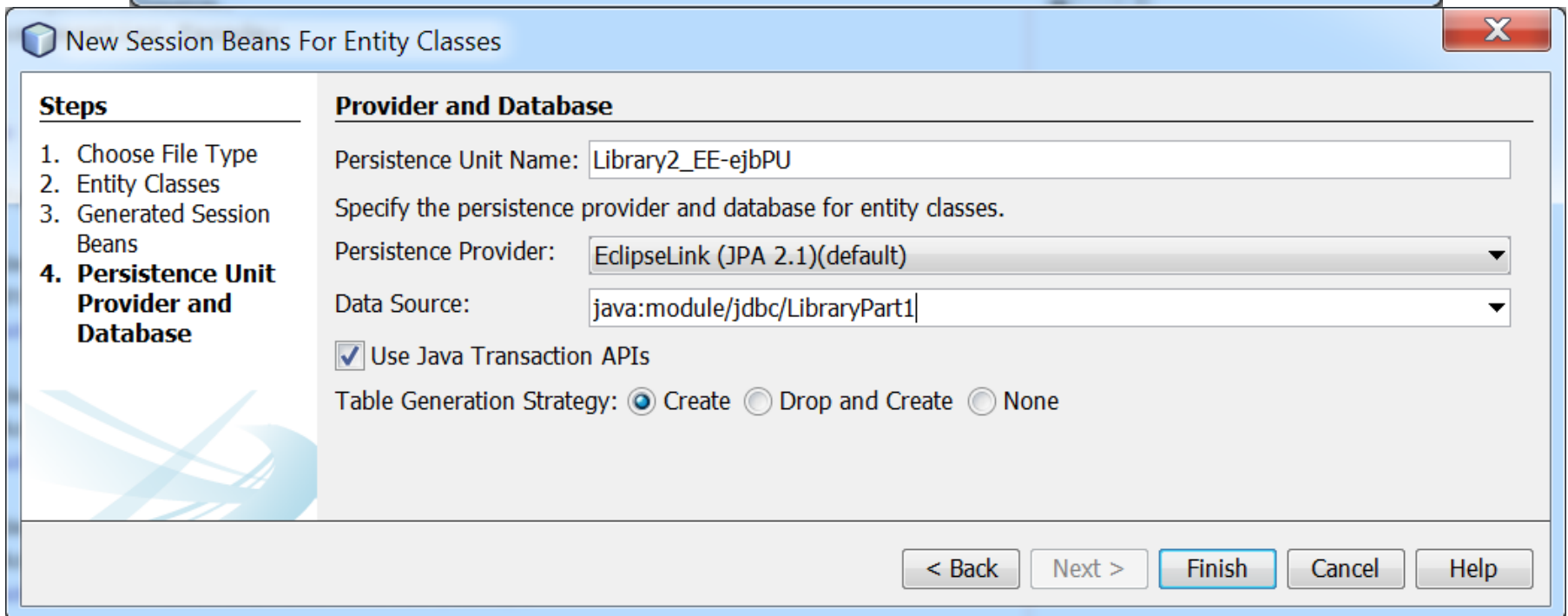


6. Widok po utworzeniu połączenia JNDI i kliknięcia na OK, w polu Data Source połączenie ma zasięg modułu – należy zmodyfikować to połączenie na globalne (verte)



The 'Create Data Source' dialog box is shown with the following fields and options:

- JNDI Name:** jdbc/LibraryPart1
- Database Connection:** jdbc:derby://localhost:1527/Bioblioteka [Bioblioteka on BIOBLIOTEKA]
- Buttons:** OK, Cancel, Help



The 'New Session Beans For Entity Classes' dialog box is shown with the following fields and options:

- Steps:**
 1. Choose File Type
 2. Entity Classes
 3. Generated Session Beans
 - 4. Persistence Unit Provider and Database**
- Provider and Database:**
 - Persistence Unit Name:** Library2_EE-ejbPU
 - Specify the persistence provider and database for entity classes.**
 - Persistence Provider:** EclipseLink (JPA 2.1)(default)
 - Data Source:** java:module/jdbc/LibraryPart1
 - Use Java Transaction APIs
 - Table Generation Strategy:** Create Drop and Create None
- Buttons:** < Back, Next >, Finish, Cancel, Help

7. Należy wybrać zakładkę **Source** i w kodzie źródłowym pliku persistence.xml zmodyfikować wartość znacznika **<jta-data-source>** na **jdbc/LibraryPart1**

The screenshot shows the NetBeans IDE 8.2 interface. The main editor window displays the `persistence.xml` file in the **Source** tab. The XML content is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
3   <persistence-unit name="Library2_EE-ejbPU" transaction-type="JTA">
4     <jta-data-source>java:module/jdbc/LibraryPart1</jta-data-source>
5     <exclude-unlisted-classes>false</exclude-unlisted-classes>
6     <properties>
7       <property name="javax.persistence.schema-generation.database.action" value="create"/>
8     </properties>
9   </persistence-unit>
10 </persistence>
```

A red arrow points from the instruction text to the `<jta-data-source>` tag in the XML code. The left sidebar shows the project structure for `Library2_EE-ejb`, including `AbstractFacade.java`, `BookFacade.java`, and `TitleBookFacade.java`. The bottom pane shows the **Output** window with the following text:

```
Java DB Database Process | GlassFish Server | Library2_Client1_EE (run)
Library2_interface.compile:
Library2_interface.jar:
deps-jar:
compile:
library-inclusion-in-archive:
Building jar: C:\Studia\LibraryJPAEJB\Library2_Client1_EE\dist\Library2_Client1_EE.jar
dist:
pre-run-deploy:
```

The status bar at the bottom right indicates the time is 11:1 and the user is INS.

8. Resultat

The screenshot displays the NetBeans IDE 8.2 interface. The main editor window shows the `persistence.xml` file with the following content:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
3   <persistence-unit name="Library2_EE-ejbPU" transaction-type="JTA">
4     <jta-data-source>jdbc/LibraryPart1</jta-data-source>
5     <exclude-unlisted-classes>>false</exclude-unlisted-classes>
6     <properties>
7       <property name="javax.persistence.schema-generation.database.action" value="create"/>
8     </properties>
9   </persistence-unit>
10 </persistence>
```

The left sidebar shows the project structure for `Library2_EE-ejb`, including `AbstractFacade.java`, `BookFacade.java`, and `TitleBookFacade.java`. The bottom panel shows the `Output` window with the following log messages:

```
Java DB Database Process
GlassFish Server
Library2_Client1_EE (run)
Library2_interface.compile:
Library2_interface.jar:
deps-jar:
compile:
library-inclusion-in-archive:
Building jar: C:\Studia\LibraryJPAEJB\Library2_Client1_EE\dist\Library2_Client1_EE.jar
dist:
pre-run-deploy:
```

A red arrow points from the text "8. Resultat" to the `<jta-data-source>jdbc/LibraryPart1</jta-data-source>` line in the XML file.

9. Widok pliku **persistence.xml** po przejściu do widoku **Design**

The screenshot displays the NetBeans IDE 8.2 interface. The main window is titled "Library2_EE - NetBeans IDE 8.2". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. A search bar is located in the top right corner.

The left sidebar shows the "Projects" view with the following structure:

- Library_2IO
- Library2_Client1_EE
- Library2_EE (selected)
- Java EE Modules
 - Library2_EE-ejb.jar
- Configuration Files
 - META-INF
 - MANIFEST.MF
- Server Resources
- Library2_EE-ejb
 - Source Packages
 - business-tier
 - integration-tier
 - AbstractFacade.java
 - BookFacade.java
 - TitleBookFacade.java

The "persistence.xml - Navigator" view shows the following XML content:

```
<?xml version="1.0" encoding="UTF-8" ?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd" >
  <persistence-unit name="Library2_EE-ejbPU" >
```

The main editor area is in "Design" view, showing the "Persistence Units" configuration for "Library2_EE-ejbPU". The "General" tab is active, displaying the following settings:

- Persistence Unit Name: Library2_EE-ejbPU
- Persistence Provider: EclipseLink (JPA 2.1)(default)
- Data Source: jdbc/LibraryPart1
- Use Java Transaction APIs
- Table Generation Strategy: Create Drop and Create None
- Validation Strategy: Auto Callback None
- Shared Cache Mode: All None Enable Selective Disable Selective Unspecified

The "Output" view at the bottom shows the following log messages:

```
pre-run-deploy:
Initial deploying Library2_EE to C:\Studia\LibraryJPAEJB\Library2_EE\dist\gfdploy\Library2_EE
Completed initial distribution of Library2_EE
post-run-deploy:
run-deploy:
BUILD SUCCESSFUL (total time: 6 seconds)
```

The status bar at the bottom right shows the time 13:46 and the keyboard indicator "INS".

10. Należy usunąć zaznaczenie z pola **Include All Entity Classes...** i wybrać z listy **Add Class...** encje **TitleBook** i **Book**. Spowoduje to w kolejnych krokach wygenerowanie jedynie dwóch tabel w bazie danych: **TitleBook** i **Book**. Tabela **TitleBook** będzie zawierała dane obiektów typu **TitleBook** i **TitleBookRead**

The screenshot displays the NetBeans IDE 8.2 interface. The left sidebar shows a project named "BIBLIOTEKA" with a "Tables" folder selected. The main editor area shows the "General" tab of the persistence.xml configuration. The "Include All Entity Classes in 'Library2_EE-ejb' Module" checkbox is unchecked. The "Include Entity Classes" list contains "subbusinesstier.entities.Book" and "subbusinesstier.entities.TitleBook". The "Add Class..." button is visible. The bottom right pane shows the "Output" window with the following text:

```
Java DB Database Process GlassFish Server Library2_EE (run-deploy)
pre-run-deploy:
Initial deploying Library2_EE to C:\Studia\LibraryJPAEJB\Library2_EE\dist\gfdeploy\Li
Completed initial distribution of Library2_EE
post-run-deploy:
run-deploy:
BUILD SUCCESSFUL (total time: 6 seconds)
```

The bottom right corner of the IDE shows the system time as 4:22.

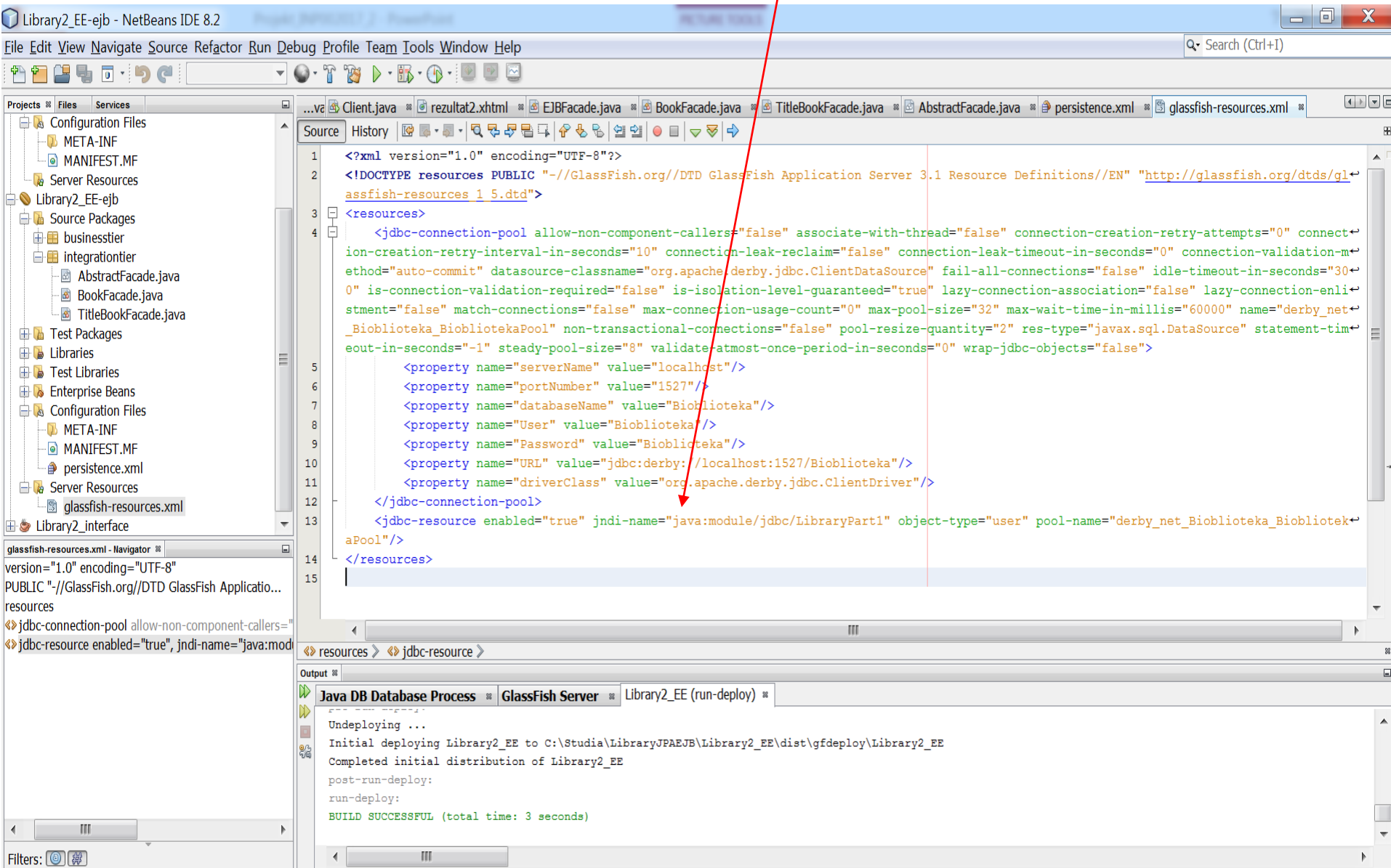
11. Należy „przesunąć” plik `glassfish-resources.xml` z folderu **Configuration Files/Meta-INF** do folderu **Server Resources**. Przed wykonaniem należy w zakładce Files sprawdzić, czy istnieje folder **setup** w module **Library2_EE-ejb**. Jeśli nie, należy go utworzyć (po zaznaczeniu projektu należy wybrać z listy pozycję **New/Folder** i założyć folder o nazwie **setup**).

The screenshot displays the NetBeans IDE interface with the 'Library2_EE-ejb' project selected in the Projects view. The 'New Folder' dialog box is open, showing the following details:

- Steps:**
 1. Choose File Type
 2. Name and Location
- Name and Location:**
 - Folder Name: `folder`
 - Project: `Library2_EE-ejb`
 - Parent Folder: `C:\Studia\LibraryJPAEJB\Library2_EE\Library2_EE-ejb`
 - Created Folder: `C:\Studia\LibraryJPAEJB\Library2_EE\Library2_EE-ejb\folder`

The dialog box also features navigation buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

12. Należy „przesunąć” plik **glassfish-resotces.xml** z folderu **Configuration Files/Meta-INF** do folderu **Server Resources**. Następnie, należy zmienić atrybut **jndi-name** w znaczniku **<jdbc-resources** na **jdbc/LibraryPart1**



The screenshot shows the NetBeans IDE 8.2 interface. The left sidebar displays the project structure for 'Library2_EE-ejb', with 'Server Resources' selected. The main editor window shows the 'glassfish-resotces.xml' file. The code is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE resources PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 Resource Definitions//EN" "http://glassfish.org/dtds/glassfish-resources_1_5.dtd">
3 <resources>
4   <jdbc-connection-pool allow-non-component-callers="false" associate-with-thread="false" connection-creation-retry-attempts="0" connection-creation-retry-interval-in-seconds="10" connection-leak-reclaim="false" connection-leak-timeout-in-seconds="0" connection-validation-method="auto-commit" datasource-classname="org.apache.derby.jdbc.ClientDataSource" fail-all-connections="false" idle-timeout-in-seconds="300" is-connection-validation-required="false" is-isolation-level-guaranteed="true" lazy-connection-association="false" lazy-connection-enlistment="false" match-connections="false" max-connection-usage-count="0" max-pool-size="32" max-wait-time-in-millis="60000" name="derby_net_Biblioteka_BibliotekaPool" non-transactional-connections="false" pool-resize-quantity="2" res-type="javax.sql.DataSource" statement-timeout-in-seconds="-1" steady-pool-size="8" validate-atmost-once-period-in-seconds="0" wrap-jdbc-objects="false">
5     <property name="serverName" value="localhost"/>
6     <property name="portNumber" value="1527"/>
7     <property name="databaseName" value="Bioblioteka"/>
8     <property name="User" value="Bioblioteka"/>
9     <property name="Password" value="Bioblioteka"/>
10    <property name="URL" value="jdbc:derby://localhost:1527/Bioblioteka"/>
11    <property name="driverClass" value="org.apache.derby.jdbc.ClientDriver"/>
12  </jdbc-connection-pool>
13  <jdbc-resource enabled="true" jndi-name="java:module/jdbc/LibraryPart1" object-type="user" pool-name="derby_net_Biblioteka_BibliotekaPool"/>
14 </resources>
15
```

The 'jndi-name' attribute in the 'jdbc-resource' tag is highlighted with a red arrow, indicating the change from its previous value to 'java:module/jdbc/LibraryPart1'. The bottom status bar shows the output of the deployment process, which was successful.

13. Wynik po zmianie

The screenshot shows the NetBeans IDE 8.2 interface. The main editor displays the `glassfish-resources.xml` file with the following XML content:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 Resource Definitions//EN" "http://glassfish.org/dtds/glassfish-resources_1_5.dtd">
<resources>
  <jdbc-connection-pool allow-non-component-callers="false" associate-with-thread="false" connection-creation-retry-attempts="0" connection-creation-retry-interval-in-seconds="10" connection-leak-reclaim="false" connection-leak-timeout-in-seconds="0" connection-validation-method="auto-commit" datasource-classname="org.apache.derby.jdbc.ClientDataSource" fail-all-connections="false" idle-timeout-in-seconds="300" is-connection-validation-required="false" is-isolation-level-guaranteed="true" lazy-connection-association="false" lazy-connection-enlistment="false" match-connections="false" max-connection-usage-count="0" max-pool-size="32" max-wait-time-in-millis="60000" name="derby_net_Biblioteka_BibliotekaPool" non-transactional-connections="false" pool-resize-quantity="2" res-type="javax.sql.DataSource" statement-timeout-in-seconds="-1" steady-pool-size="8" validate-atmost-once-period-in-seconds="0" wrap-jdbc-objects="false">
    <property name="serverName" value="localhost"/>
    <property name="portNumber" value="1527"/>
    <property name="databaseName" value="Biblioteka"/>
    <property name="User" value="Biblioteka"/>
    <property name="Password" value="Biblioteka"/>
    <property name="URL" value="jdbc:derby://localhost:1527/Biblioteka"/>
    <property name="driverClass" value="org.apache.derby.jdbc.ClientDriver"/>
  </jdbc-connection-pool>
  <jdbc-resource enabled="true" jndi-name="jdbc/LibraryPart1" object-type="user" pool-name="derby_net_Biblioteka_BibliotekaPool"/>
</resources>
```

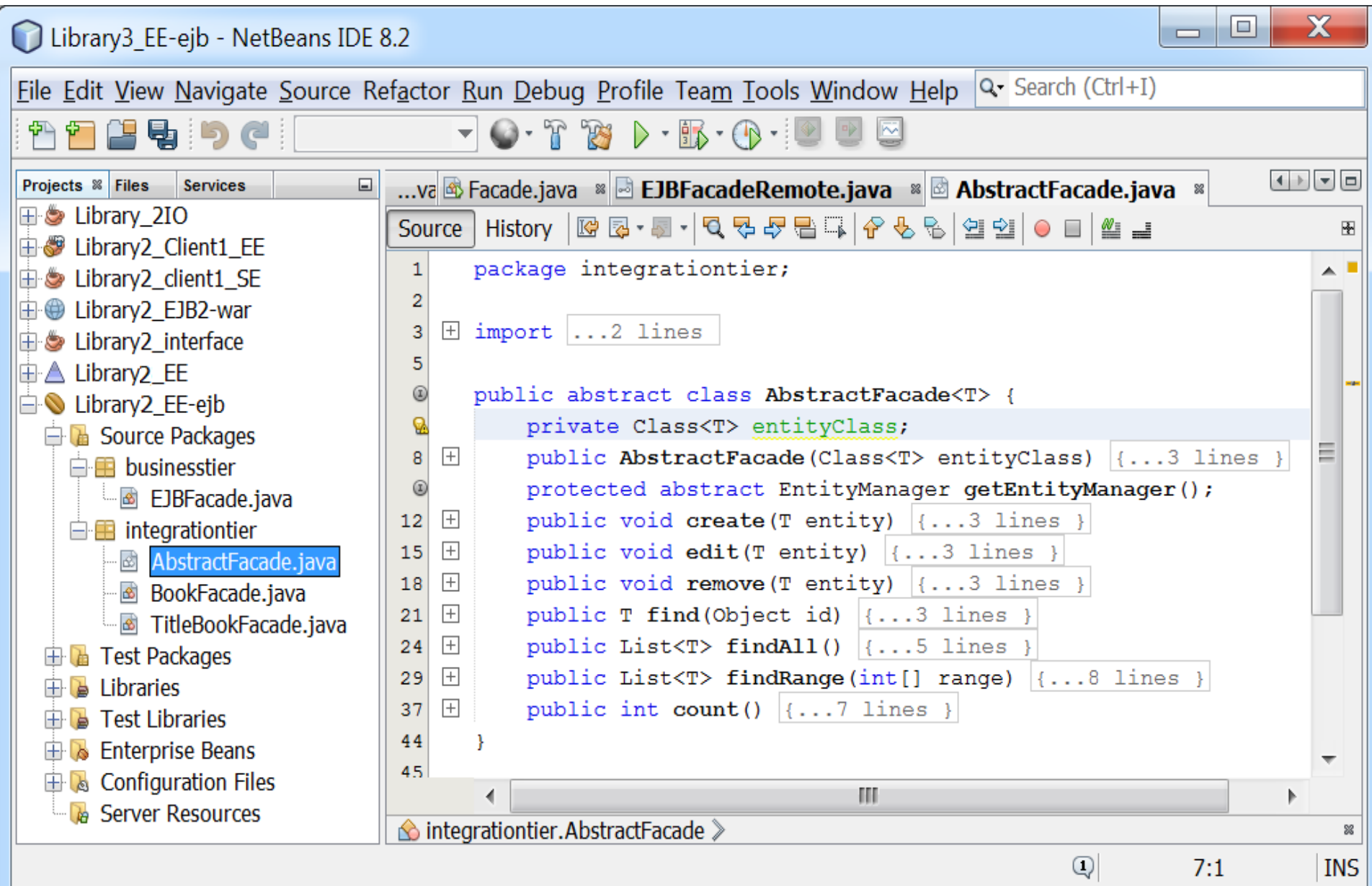
The output window shows the deployment process:

```
Java DB Database Process | GlassFish Server | Library2_EE (run-deploy)
Undeploying ...
Initial deploying Library2_EE to C:\Studia\LibraryJPAEJB\Library2_EE\dist\gfdeploy\Library2_EE
Completed initial distribution of Library2_EE
post-run-deploy:
run-deploy:
BUILD SUCCESSFUL (total time: 3 seconds)
```

Filters: [Icons]

Library2_EE deployed.

14. Klasa abstrakcyjna generyczna **AbstractFacade** z definicją uniwersalnych metod obsługujących transakcje JPA - parametr T może być zastąpiony każdą z klas typu Entity



The screenshot displays the NetBeans IDE 8.2 interface. The left sidebar shows a project tree for 'Library3_EE-ejb' with the 'integrationtier' package selected, containing the 'AbstractFacade.java' file. The main editor window shows the source code of 'AbstractFacade.java' with the following content:

```
1 package integrationtier;
2
3 import ...2 lines
4
5
6 public abstract class AbstractFacade<T> {
7     private Class<T> entityClass;
8     public AbstractFacade(Class<T> entityClass) {...3 lines }
9     protected abstract EntityManager getEntityManager();
10
11
12     public void create(T entity) {...3 lines }
13
14     public void edit(T entity) {...3 lines }
15
16     public void remove(T entity) {...3 lines }
17
18     public T find(Object id) {...3 lines }
19
20     public List<T> findAll() {...5 lines }
21
22     public List<T> findRange(int[] range) {...8 lines }
23
24     public int count() {...7 lines }
25
26 }
```

The status bar at the bottom indicates the current file is 'integrationtier.AbstractFacade' and the cursor is at line 7:1.

15. Klasa **TitleBookFacade** implementująca klasę **AbstractFacade** - kontroler typu Session Bean do utrwalania obiektów typu **TitleBook** i **TitleBookRead**. Dodano metodę **addTitleBooks**.

The screenshot shows the NetBeans IDE 8.2 interface. The main editor displays the source code for `BookFacade.java` in the `integrationtier` package. The code implements the `TitleBookFacade` class, which extends `AbstractFacade<TitleBook>`. It is annotated with `@Stateless` and `@PersistenceContext(unitName = "Library2_EE-ejbPU")`. A private `EntityManager em` is declared. The `getEntityManager()` method is overridden to return `em`. The `addTitleBooks(List<TitleBook> titles)` method iterates over the list and persists each title using `em.persist(title)` if its ID is null.

```
1 package integrationtier;
2
3 import java.util.List;
4 import javax.ejb.Stateless;
5 import javax.persistence.EntityManager;
6 import javax.persistence.PersistenceContext;
7 import subbusinessstier.entities.TitleBook;
8
9 @Stateless
10 public class TitleBookFacade extends AbstractFacade<TitleBook> {
11     @PersistenceContext(unitName = "Library2_EE-ejbPU")
12     private EntityManager em;
13
14     @Override
15     protected EntityManager getEntityManager() {return em;}
16
17     public TitleBookFacade() {
18         super(TitleBook.class);
19     }
20     public void addTitleBooks(List<TitleBook> titles){
21         for(TitleBook title: titles)
22             if (title.getId() == null)
23                 getEntityManager().persist(title);
24     }
25 }
26
```

The left sidebar shows the project structure for `Library2_EE-ejb`, including packages `businessstier` and `integrationtier`, and files `AbstractFacade.java`, `BookFacade.java`, and `TitleBookFacade.java`. The `em - Navigator` window shows the members of `TitleBookFacade`: `TitleBookFacade()`, `addTitleBooks(List<TitleBook> titles)`, `getEntityManager() : EntityManager`, and `em : EntityManager`. The status bar at the bottom indicates the current context is `integrationtier.TitleBookFacade > em`.

16. Klasa **BookFacade** implementująca klasę **AbstractFacade**

- kontroler typu Session Bean do utrwalania obiektów typu **Book**. Dodano metodę **addBooks**

Library2_EE-ejb - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Search (Ctrl+I)

Projects Files Services

- META-INF
- MANIFEST.MF
- Server Resources
- Library2_EE-ejb
 - Source Packages
 - business-tier
 - integration-tier
 - AbstractFacade.java
 - BookFacade.java
 - TitleBookFacade.java
 - Test Packages
 - Libraries
 - Test Libraries
 - Enterprise Beans
 - Configuration Files
 - META-INF
 - MANIFEST.MF
 - persistence.xml
 - Server Resources

em - Navigator

Members

- BookFacade :: AbstractFacade<Book>
 - BookFacade()
 - addBooks(List<TitleBook> titles)
 - getEntityManager() : EntityManager
 - em : EntityManager

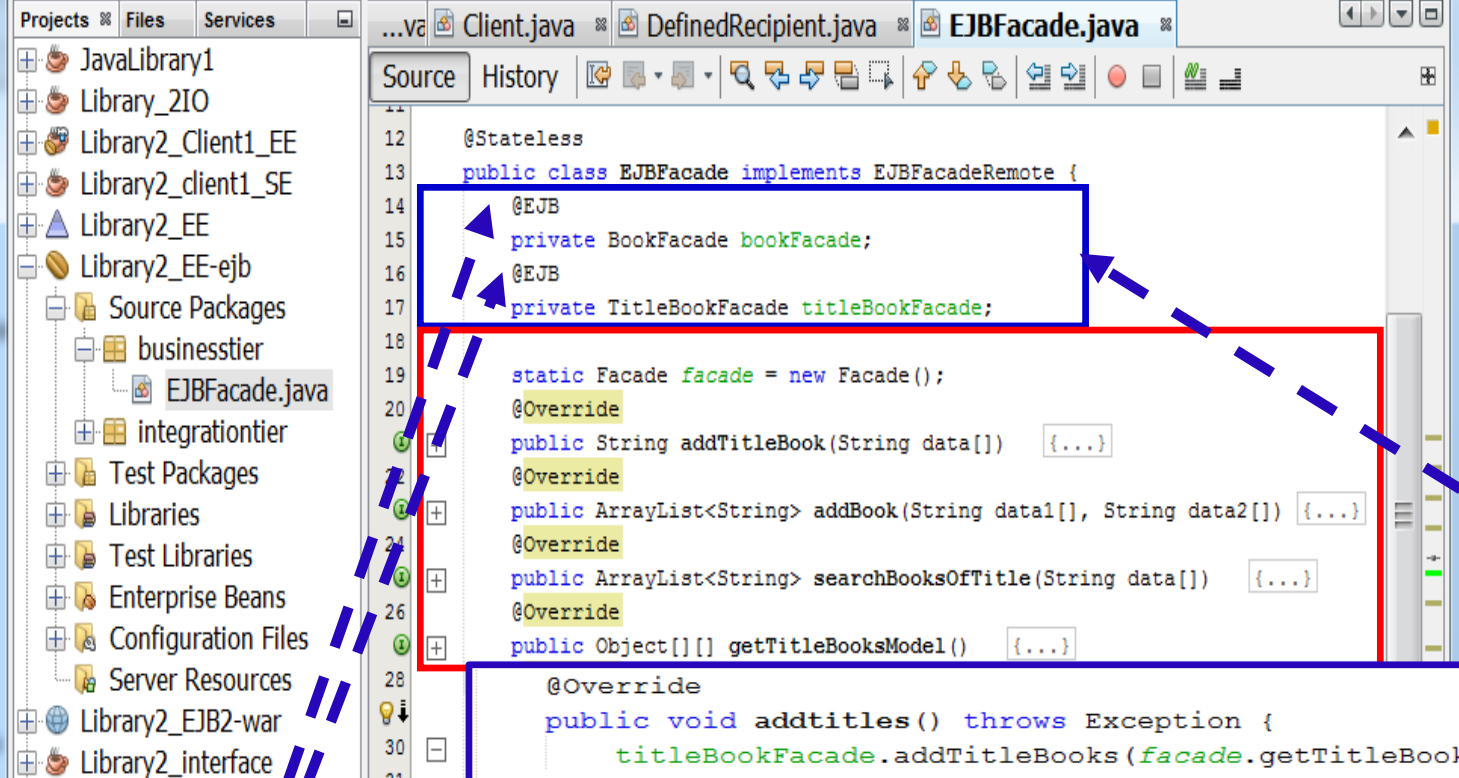
```
1 package integrationtier;
2
3 import java.util.List;
4 import javax.ejb.Stateless;
5 import javax.persistence.EntityManager;
6 import javax.persistence.PersistenceContext;
7 import subbusiness-tier.entities.Book;
8 import subbusiness-tier.entities.TitleBook;
9
10 @Stateless
11 public class BookFacade extends AbstractFacade<Book> {
12     @PersistenceContext(unitName = "Library2_EE-ejbPU")
13     private EntityManager em;
14
15     @Override
16     protected EntityManager getEntityManager() { return em; }
17     public BookFacade() {
18         super(Book.class);
19     }
20     public void addBooks(List<TitleBook> titles) {
21         for (TitleBook title:titles) {
22             if (title.getId() == null)
23                 continue;
24             for(Book book:title.getBooks())
25                 if (book.getId() == null)
26                     getEntityManager().persist(book);
27         }
28     }
29 }
```

integrationtier.BookFacade > em >

Output

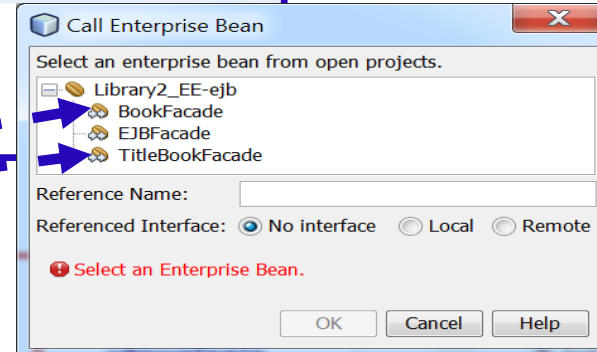
Java DB Database Process GlassFish Server SQL 8 execution Lib

13:30 INS



17. Klasa EJBFacade
- główny komponent warstwy biznesowej udostępnia **metody logiki biznesowej** i zarządza komponentami do utrwalania obiektów typu Entity.

Referencje @EJB
dodano w
następujący sposób:
należy kliknąć na
powierzchnię
edytora klasy i
wybrać pozycję
Insert Code.../Call
Enterprise Bean... i
wybrać z listy
kolejno



18. Uzupełnienie deklaracji metod o zdalnym dostępie do wybranych metod komponentów do utrwalania obiektów typu Entity w interfejsie **EJBFacadeRemote** komponentu **Session Bean** typu **EJBFacade**

The screenshot displays the NetBeans IDE 8.2 interface. The main editor window shows the source code for `EJBFacadeRemote.java`. The code defines a remote interface with several methods for interacting with a database. The left sidebar shows the project structure, with `EJBFacadeRemote.java` selected under the `business-tier` package. The bottom status bar indicates the current session bean is `Library2_EE-ejb (clean, dist)`.

```
13  * @author PWR
14  */
15  @Remote
16  public interface EJBFacadeRemote {
17      public String addTitleBook(String data[]);
18      public ArrayList<String> addBook(String data1[], String data2[]);
19      public ArrayList<String> searchBooksOfTitle(String data[]);
20  //model ComboBox
21      public Object[][] getTitleBooksModel();
22  //model JTable
23      public void addtitles() throws Exception;
24  //zapis tytuł,Åłw do bazy danych
25      public void addbooks() throws Exception;
26  //zapis ksiÄ...łEk do bazy danych
27      public ArrayList<String> titles() throws Exception;
28  //model JComboBox
29  }
30
```

19. Uruchomienie części serwerowej aplikacji typu EE – deploy (po wcześniejszym przeprowadzonym procesie **Build** na projektach składowych)

The screenshot displays the NetBeans IDE 8.2 interface. The title bar reads "Library2_EE - NetBeans IDE 8.2". The menu bar includes "File", "Edit", "View", "Navigate", "Source", "Refactor", "Run", "Debug", "Profile", "Team", "Tools", "Window", and "Help". The toolbar contains various icons for file operations and development actions. The "Projects" window on the left shows a tree structure with "Library2_EE" selected. A context menu is open over "Library2_EE", with "Deploy" highlighted. The main editor shows the source code of "EJBFacade.java" with the following methods:

```
public ArrayList<String> addBook(String data1[], String data2[↵
    return facade.addBook(data1, data2); }
@Override
public ArrayList<String> searchBooksOfTitle(String data[]) {
    return facade.searchBooksOfTitle(data); }
@Override
public Object[][] getTitleBooksModel() {
    return facade.getTitleBooksModel(); }
@Override
public void addtitles() throws Exception {
    titleBookFacade.addTitleBooks(facade.getTitleBooks()); }
@Override
public void addbooks() throws Exception {
    bookFacade.addBooks(facade.getTitleBooks()); }
@Override
public ArrayList<String> titles() throws Exception {
    List<TitleBook> help1 = titleBookFacade.findAll();
    ArrayList<String> help2 = new ArrayList();
    for (TitleBook t : help1) {
        ArrayList<String> help3 = t.getBooksModel();
        help2.addAll(help3); }
    return help2; }
```

The Output window at the bottom shows the deployment progress:

```
Java DB Database Process
GlassFish Server
SQL 8 execution
SQL 9 execution
```

The status bar at the bottom right shows the time "44:57" and the text "INS".

20. Wygenerowanie pustych tabel w bazie danych w wyniku operacji deploy.

The screenshot displays the NetBeans IDE 8.2 interface. On the left, the 'Projects' pane shows a database schema for 'BIOBLIOTEKA' on 'localhost:1527/Biblioteka'. The schema includes tables like 'BOOK' and 'TITLEBOOK', each with columns such as 'ID', 'NUMBER', 'ISBN', and 'AUTHOR'. The 'Tables' folder is highlighted. On the right, the 'Source' editor shows a Java file named 'Client.java' with a package declaration 'package client;'. Below the source editor, the 'Output' window displays the 'Java DB Database Process' log, which includes the following text:

```
pre-run-deploy:  
Initial deploying I  
Completed initial c  
post-run-deploy:  
run-deploy:  
BUILD SUCCESSFUL (t
```

At the bottom of the IDE, the 'Navigator' pane shows '<No View Available>' and the status bar indicates '1:1' and 'INS'.

Projects: Test Packages, Libraries, Test Libraries, Library2_Client1_EE, Source Packages, client_tier, Client.java, library, Book_form.java, Card0.java, Loan_form.java, Panel_util.java, Title_form.java, Test Packages, Libraries, Test Libraries, Configuration Files, Server Resources, Library2_EE, Java EE Modules, Library2_EE-ejb.jar, Configuration Files, META-INF

actionPerformed - Navigator: Members, Book_form :: JPanel : Action, Book_form(), actionPerformed(ActionEvent), init(), list_content(ArrayList<String>), paintComponent(Graphics), print_books(), table_content(), title(): String[], add_book: JButton, add_database: JButton, books: JComboBox

```

117     @Override
118     public void actionPerformed(ActionEvent event) {
119         if (event.getSource() == clear_selection) {
120             table.getSelectionModel().clearSelection();
121             return;
122         }
123         if (!table.getSelectionModel().isSelectionEmpty()) {
124             String what_book_type;
125             if (number.getText().equals("")) {
126                 JOptionPane.showMessageDialog(this, "required value");
127                 return;
128             }
129             what_book_type = "0";
130             String data2[] = {what_book_type, (String) number.getText()};
131             ArrayList<String> help3 = Client.getFacade().addBook(title(), data2);
132             if (help3 != null) {
133                 list_content(help3, books);
134             }
135             table.getSelectionModel().clearSelection();
136         }
137         if (event.getSource() == get_database) {
138             try {
139                 list_content(Client.getFacade().titles(), books);
140             } catch (Exception e) {
141                 books.removeAllItems();
142                 books.addItem("Errors");
143             }
144         }
145         if (event.getSource() == add_database) {
146             try {
147                 Client.getFacade().addtitles();
148             } catch (Exception e) {
149                 books.removeAllItems();
150                 books.addItem("Errors");
151             }
152         }
153     }

```

library.Book_form > actionPerformed >

Output: ver, Library2_Client1_EE (run), Library2_Client1_EE (run) #2, SQL 19 execution, SQL 20 execution

148:6 INS

21. Uzupełnienie formularza **Book_form** o uruchamianie operacji wyświetlania tych pobranych z bazy danych i utrwalania danych tytułów i książek w bazie danych. Jest to tymczasowy sposób prezentacji z GUI opracji na bazie danych

22. Jedna z aplikacji klienckich (po uruchomieniu za pomocą operacji **Run**) po wprowadzeniu danych tytułów i książek – dane 1-go tytułu i jego książek przechowywane w pamięci programu pokazane w formularzu **Book_form**.

MenuDemo

A Menu Another Menu

Publisher	ISBN	Title	Author	Actor
Wydawca1	12345	Tytul1	Autor1	
Wydawca1	12345	Tytul1	Autor1	Aktor1

Number of a book

1

Add book

Clear selection

Persist data

Show database

Books

Title: Tytul1 Author: Autor1 ISBN: 12345 Publisher: Wydawca1 Number: 1

23. Druga z aplikacji klienckich (po uruchomieniu za pomocą operacji **Run**) po wprowadzeniu danych tytułów i książek – dane 2-go tytułu i jego książek przechowywane w pamięci programu pokazane w formularzu **Book_form**

MenuDemo

A Menu Another Menu

Publisher	ISBN	Title	Author	Actor
Wydawca1	12345	Tytul1	Autor1	
Wydawca1	12345	Tytul1	Autor1	Aktor1

Number of a book

11

Add book

Clear selection

Persist data

Show database

Books

Title: Tytul1 Author: Autor1 ISBN: 12345 Publisher: Wydawca1 Actor: Aktor1 Number: 11

Title: Tytul1 Author: Autor1 ISBN: 12345 Publisher: Wydawca1 Actor: Aktor1 Number: 11

Title: Tytul1 Author: Autor1 ISBN: 12345 Publisher: Wydawca1 Actor: Aktor1 Number: 1

24. Dane tytułów zapisane w bazie danych po naciśnięciu klawisza **Persist data** na formularzu **Book_form**, zaprezentowane po kliknięciu na nazwę tabeli **TitleBook** i wybór pozycji **View Data...**

The screenshot shows the NetBeans IDE 8.2 interface. The left sidebar displays a project tree for 'jdbc:derby://localhost:1527/Biblioteka', with the 'TITLEBOOK' table selected. The main window shows a SQL editor with the query: `SELECT * FROM BIOBLIOTEKA.TITLEBOOK FETCH FIRST 100 ROWS ONLY;`. Below the editor, the results are displayed in a table format. The table has columns: #, ID, DTYPE, ISBN, AUTHOR, PUBLISHER, TITLE, and ACTOR. Two rows are visible: row 1 with ID 1, DTYPE TitleBook, ISBN 12345, AUTHOR Autor1, PUBLISHER Wydawca1, TITLE Tytul1, and ACTOR <NULL>; row 2 with ID 3, DTYPE TitleBookRead, ISBN 12345, AUTHOR Autor1, PUBLISHER Wydawca1, TITLE Tytul1, and ACTOR Aktor1. The bottom status bar shows 'Library2_Client1_EE (run) #2' with a 'running...' indicator.

#	ID	DTYPE	ISBN	AUTHOR	PUBLISHER	TITLE	ACTOR
1	1	TitleBook	12345	Autor1	Wydawca1	Tytul1	<NULL>
2	3	TitleBookRead	12345	Autor1	Wydawca1	Tytul1	Aktor1

25. Dane książek zapisane w bazie danych podczas zapisania tytułów (@OneToMany(mappedBy = "titleBook", cascade=ALL)) zaprezentowane po kliknięciu na nazwę tabeli **Book** i wybór pozycji **View Data...**

NetBeans IDE 8.2

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

Search (Ctrl+I)

Projects Files Services

jdbc:derby://localhost:1527/Biblioteka

BIOBLIOTEKA

- Tables
 - BOOK
 - SEQUENCE
 - TITLEBOOK
- Views
- Procedures
- Other schemas

jdbc:derby://localhost:1527/database [t

jdbc:derby://localhost:1527/ERD [erd o

jdbc:derby://localhost:1527/Katalog_ks

jdbc:derby://localhost:1527/Katalogksia

jdbc:derby://localhost:1527/Library201

Navigator

Members <empty>

Book_form :: JPanel : ActionListener

- Book_form()
- actionPerformed(ActionEvent event)
- init()

.....] SQL 19 [jdbc:derby://localhost:15...] SQL 20 [jdbc:derby://localhost:15...]

Connection: jdbc:derby://localhost:1527/Biblioteka [Biblioteka...]

```
1 SELECT * FROM BIOBLIOTEKA.BOOK FETCH FIRST 100 ROWS ONLY;
2
```

SELECT * FROM BIOBLIOTEKA...

Max. rows: 100 | Fetched Rows: 3 | Matching Rows:

#	ID	NUMBER	TITLEBOOK_ID
1	5	1	3
2	4	11	3
3	2	1	1

Output

ient1_EE (run) Library2_Client1_EE (run) #2 SQL 19 execution SQL 20 execution

Library2_Client1_EE (run) #2 running... (1 more...) 1:1 INS

26. Dane tytułów i książek pobrane z bazy danych i wyświetlone w liście rozwijanej (przycisk **Show database**).

The screenshot shows a Java Swing application window titled "MenuDemo". The window contains a menu with two items: "A Menu" and "Another Menu". Below the menu is a table with the following data:

Publisher	ISBN	Title	Author	Actor
Wydawca1	12345	Tytul1	Autor1	
Wydawca1	12345	Tytul1	Autor1	Aktor1

Below the table is a text field labeled "Number of a book" containing the value "1". To the right of the text field is a menu with four buttons: "Add book", "Clear selection", "Persist data", and "Show database". Below the menu is a list box labeled "Books" containing the following items:

- Title: Tytul1 Author: Autor1 ISBN: 12345 Publisher: Wydawca1 Number: 1
- Title: Tytul1 Author: Autor1 ISBN: 12345 Publisher: Wydawca1 Number: 1
- Title: Tytul1 Author: Autor1 ISBN: 12345 Publisher: Wydawca1 Actor: Aktor1 Number: 11
- Title: Tytul1 Author: Autor1 ISBN: 12345 Publisher: Wydawca1 Actor: Aktor1 Number: 1

Two red arrows point from the "Show database" button to the list box, indicating that the button triggers the display of the data in the list box.