

Instrukcja 11

Laboratorium 14-15 Testy funkcjonalne

Cel laboratorium:**Nabycie umiejętności tworzenia testów funkcjonalnych za pomocą narzędzia Selenium IDE.**

- Wg wskazówek podanych w p.1 **Dodatku 2**, należy zainstalować narzędzie **Selenium IDE** w przeglądarce **FireFox** firmy **Mozilla**.
- Na podstawie przykładu z **Dodatku 1**, należy w środowisku **NetBeans 8.1**, z zainstalowaną platformą **Java EE 7.0** (instrukcja do lab1), wykonać aplikację wygenerowaną na podstawie schematu relacyjnej bazy danych wg zasady „database-first”. Wykonanie pliku zawierającego skrypt SQL z instrukcjami tworzącymi tabele należy oprzeć na projektach w ramach przedmiotu **Bazy danych 2**. Dalszy ciąg działań wykonać zgodnie z tutorialem z **Dodatku 1**. Należy zdefiniować przykładowe scenariusze testowanych przypadków użycia aplikacji w zakresie dodawania, edycji i usuwania danych (p.2, **Dodatek 2**)
- Należy wykonać testy typu **Test Case** aplikacji typu **Java Web Application**, wykonanej w p.2. W tabelce poniżej podano informacje dotyczące wyboru stron takiej aplikacji do testowania oraz przykładów rozwiązań typu **Test Case**.

W **Dodatku 2** umieszczone zostały przykłady testów funkcjonalnych, które umożliwiają sprawdzenie działania wygenerowanej aplikacji w przykładzie z **Dodatku 1** w zakresie budowy stron oraz kontrolę poprawnego działania aplikacji za pomocą poleceń z grupy assert... oraz verify...

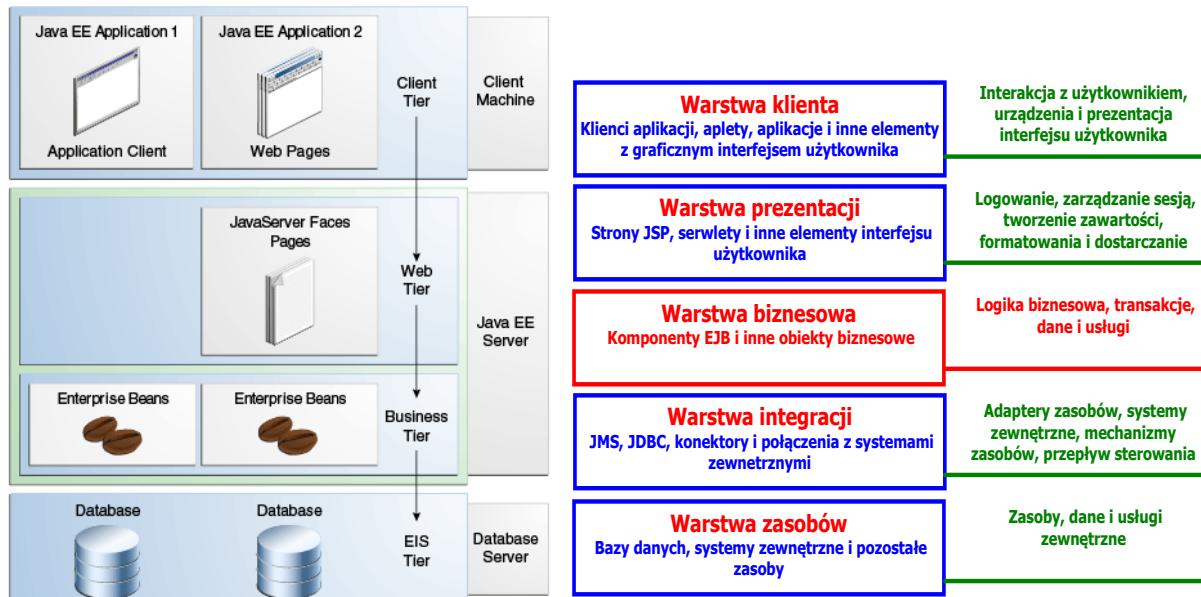
Grupa Liczba osób	Liczba stron do testowania	Przykłady stron			
		1-a para stron		2-a para stron	
Create.html	List.html	Edit.html (na podstawie Create.html)	View.html		
Przykłady rozwiązań					
<i>TestSelenium_KatalogKsiazek1_1 (str. 24-25 instrukcji)</i>					
<i>TestSelenium_KatalogKsiazek1_2</i>					
Tabela 3, Tabela 1	Weryfikacja- Tabela 2, Tabela 4	Tabela 3, Tabela 1	Weryfikacja- Tabela 2, Tabela 4		
1	1 para stron	Nagranie obsługi dodawana nowej danej i /lub ręczne zdefiniowanie obsługi dodawania nowej danej. Weryfikacja elementów UI strony	Nagranie obsługi wyświetlania danych i /lub ręczne zdefiniowanie obsługi wyświetlania. Dokonanie weryfikacji zawartości wyświetlanych danych	Nagranie obsługi edykcji wybranej danej i /lub ręczne zdefiniowanie obsługi edycji wybranej danej. Weryfikacja elementów UI strony	Nagranie obsługi wyświetlania wybranej danej i /lub ręczne zdefiniowanie obsługi wyświetlania takiej danej. Dokonanie weryfikacji zawartości wyświetlonej danej
2	2 pary stron				

- Wykonanie zestawu testów typu **Test Suite**, który powinien zawierać testy typu **Test Case** wykonane w p.3. – na podstawie przykładu budowy zestawu testów **TestSelenium_KatalogKsiazek1** w **Dodatku 2**.

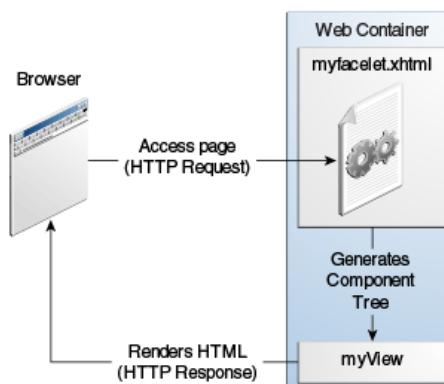
Dodatek 1

Przykład tworzenia aplikacji internetowej wg zasady „database-first development”.

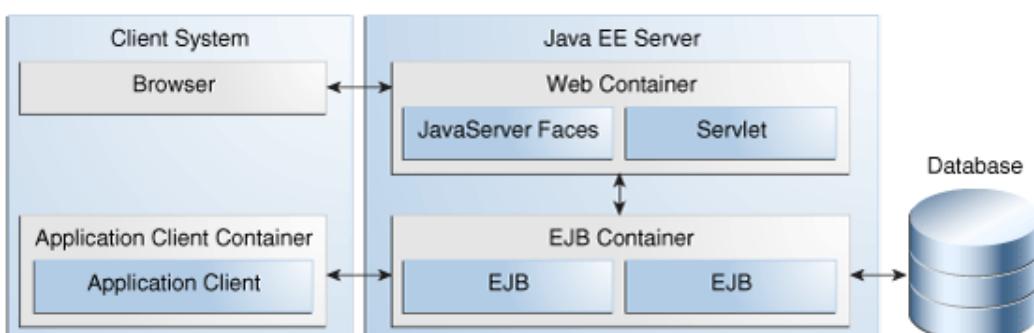
1. Krótka informacja dotycząca wielowarstwowej aplikacji internetowej na podstawie tutorialu Java EE 7 ze strony <https://docs.oracle.com/javaee/7/tutorial>.



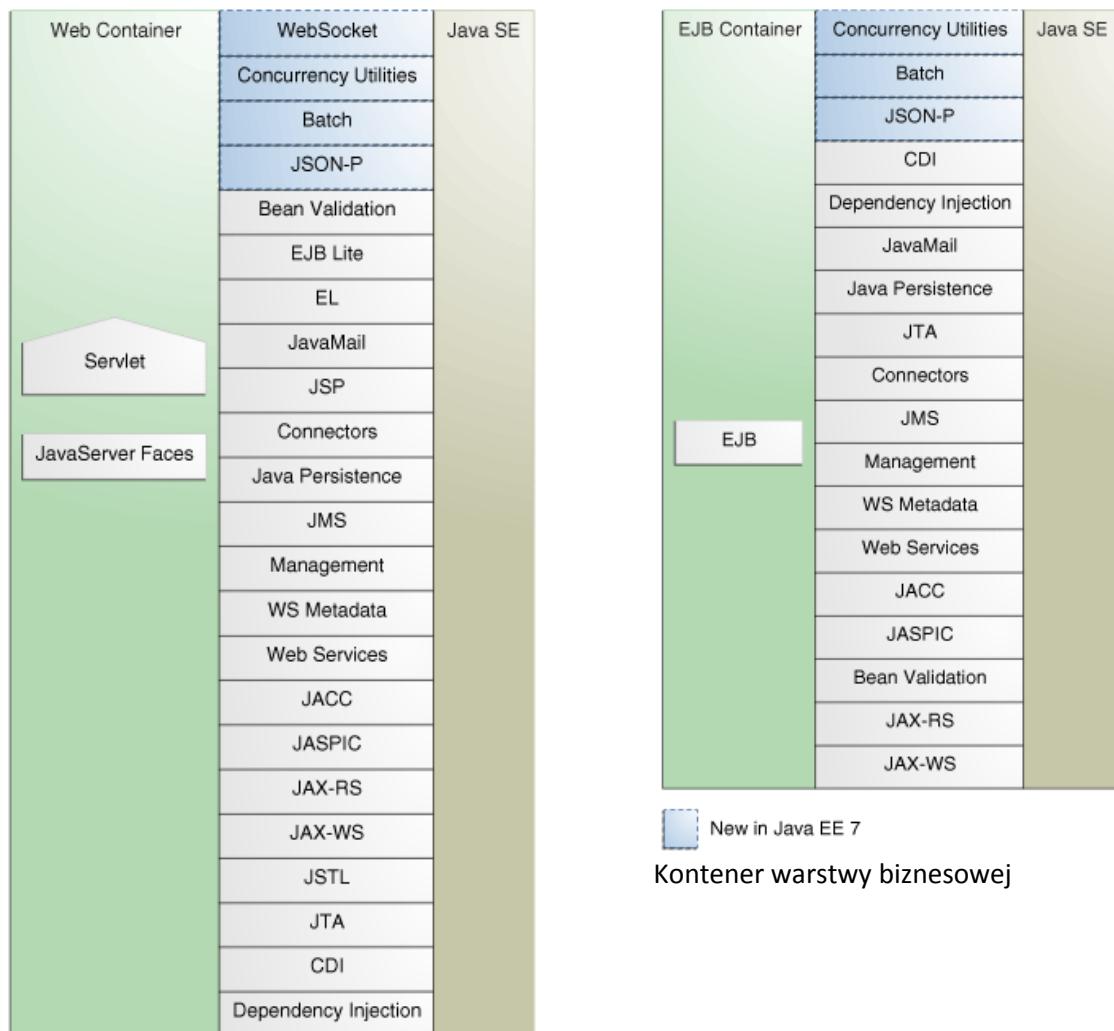
Model wielowarstwowej aplikacji na platformie **Java EE**, gdzie każda z warstw jest zbudowana z różnych typów komponentów.



Komunikacja “klient – strony **JavaServer Faces**” aplikacji internetowej typu **JSF** platformy **Java EE**.

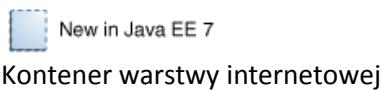


Kontenery na platformie **Java EE** umożliwiające obsługę różnego typu komponentów przez serwer aplikacji **Java EE Server**



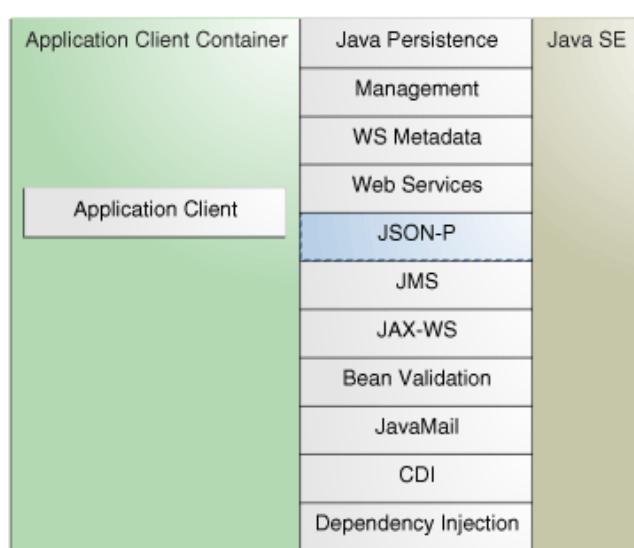
New in Java EE 7

Kontener warstwy biznesowej



New in Java EE 7

Kontener warstwy internetowej

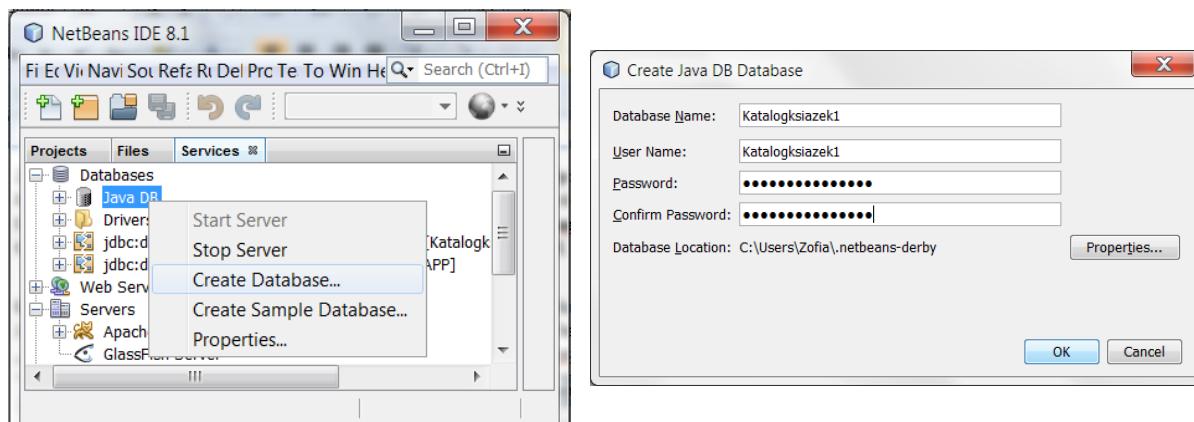


New in Java EE 7

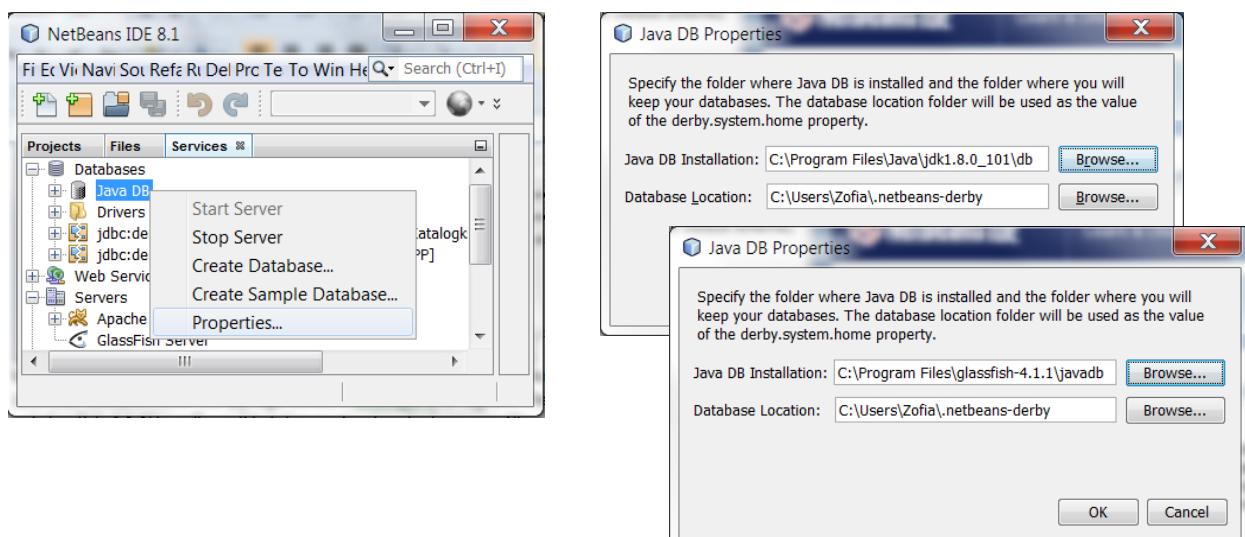
Kontener warstwy klienta

2. Przebieg tworzenia wielowarstwowej aplikacji internetowej w technologii **JavaServer Faces** w środowisku NetBeans 8.1 (wg <http://netbeans.org/kb/docs/web/jsf20-crud.html>)

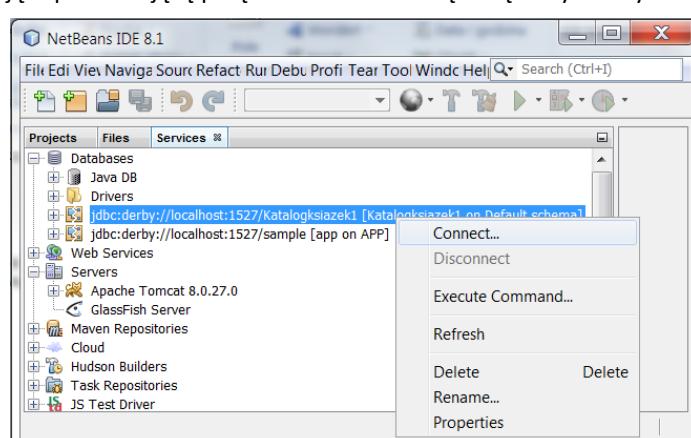
- 2.1. Zakładanie pustej bazy danych w systemie baz danych **Derby**: w zakładce **Services** w pozycji **Databases** kliknąć prawym klawiszem myszy na pozycję **Java DB** i następnie wybrać pozycję **Create Database**. W formularzu **Create Java DB Database** należy podać: **Database Name**: Katalogksiazek1, **User Name**: Katalogksiazek1, **Password**: Katalogksiazek1 i kliknąć na przycisk **OK**.



Uwaga: Jeśli wystąpią problemy z wykonaniem bazy danych z powodu braku połączenia z silnikiem bazy danych Java DB, pokazano poniżej, jak zmienić wersję tego silnika pochodzącego z podkatalogu **db** katalogu Javy SE (...\\jdk1.8.0_101\\db) na podkatalog **javadb** katalogu serwera GlassFish (...\\glassfish-4.1.1\\javadb).



Należy połączyć się z wykonaną, pustą bazą danych (poniżej): w zakładce **Services/Databases** kliknąć prawym klawiszem myszy na pozycję reprezentującą połączenie z utworzoną bazą danych i wybrać pozycję **Connect**.



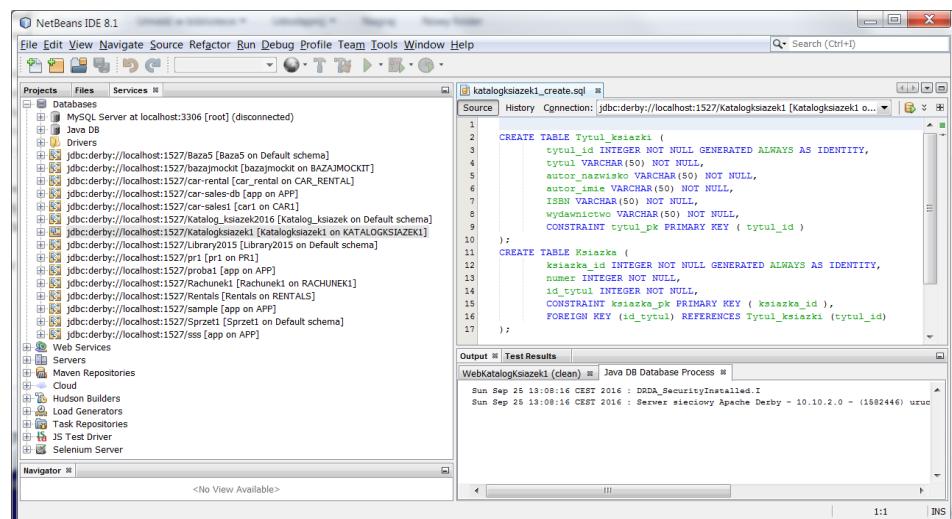
2.2. W celu utworzenia tabel relacyjnej bazy danych zdefiniowano skrypt w języku SQL **katalogksiazek1_create.sql** – w przykładzie skrypt utworzy dwie tabele: **Tytul_ksiazki** oraz **Ksiazka** powiązane relacją 1..*. Przykład ten nawiązuje do przykładu z Instrukcji 1 do lab1 (diagram klas z p.5.2 instrukcji).

```

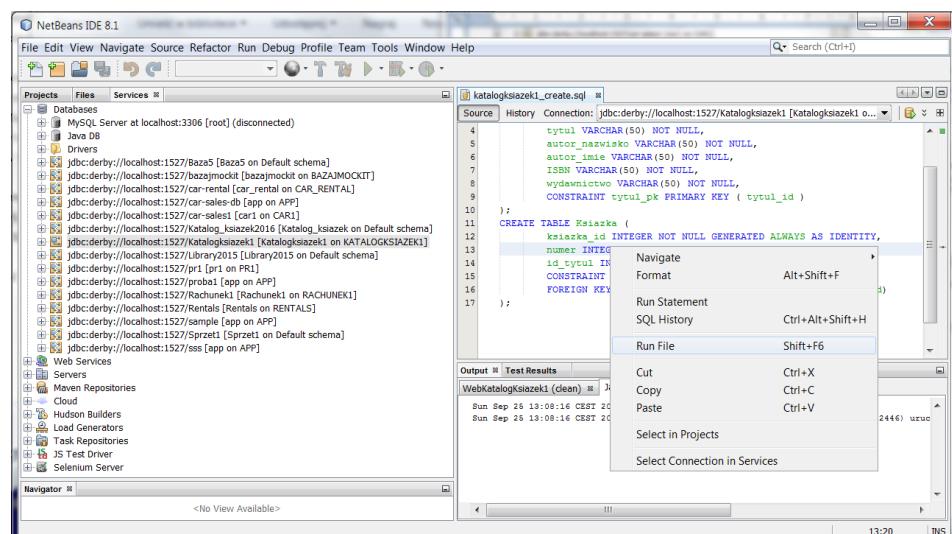
CREATE TABLE Tytul_ksiazki (
    tytul_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY,
    tytul VARCHAR(50) NOT NULL,
    autor_nazwisko VARCHAR(50) NOT NULL,
    autor_imie VARCHAR(50) NOT NULL,
    ISBN VARCHAR(50) NOT NULL,
    wydawnictwo VARCHAR(50) NOT NULL,
    CONSTRAINT tytul_pk PRIMARY KEY (tytul_id)
);
CREATE TABLE Ksiazka (
    ksiazka_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY,
    numer INTEGER NOT NULL,
    id_tytul INTEGER NOT NULL,
    CONSTRAINT ksiazka_id PRIMARY KEY (ksiazka_id),
    FOREIGN KEY (id_tytul) REFERENCES Tytul_ksiazki (tytul_id)
);

```

Utworzenie tabel relacyjnej bazy danych - utworzony skrypt należy uruchomić w następujący sposób: w **Menu Bar** należy kliknąć na **File**, a następnie na **Open File....** wybrać utworzony wcześniej plik w formacie **.sql**. Następnie, w oknie edytora otworzonego pliku należy w zakładce **Connection** wybrać z listy połączeń połączenie do utworzonej bazy danych **Katalogksiazek1**.



Następnie, należy kliknąć prawym klawiszem myszy na powierzchnię okna edytora pliku i następnie kliknąć na pozycję **RunFile** w celu uruchomienia skryptu **katalogksiazek1_create.sql**.



Poniżej pokazano utworzone dwie tabele: **KSIAZKA** oraz **TYTUL_KSIAZKI** w bazie danych **Katalogksiazek1**.

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

- Projects:** Shows the connection to `jdbc:derby://localhost:1527/Katalogksiazek1` and the schema **KATALOGKSIAZEK1**.
- Tables:** Displays two tables: **KSIAZKA** and **TYTUL_KSIAZKI**.
 - KSIAZKA:** Contains columns `KSIAZKA_ID`, `NUMER`, and `ID_TYTUL`. It has indexes `SQL160925131620100` and `SQL160925131620101`, and a foreign key `SQL160925131620100`.
 - TYTUL_KSIAZKI:** Contains columns `TYTUL_ID`, `TYTUL`, `AUTOR_NAZWISKO`, `AUTOR_IMIE`, and `ISBN`. It has indexes `SQL160925131619880` and a foreign key `SQL160925131619880`.
- Source:** Editor window containing the `katalogksiazek1_create.sql` script.
- Output:** Shows the execution results of the create statements.

```

CREATE TABLE Tytul_ksiazki (
    tytul_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY,
    tytul VARCHAR(50) NOT NULL,
    autor_nazwisko VARCHAR(50) NOT NULL,
    autor_imie VARCHAR(50) NOT NULL,
    ISBN VARCHAR(50) NOT NULL,
    wydawnictwo VARCHAR(50) NOT NULL,
    CONSTRAINT tytul_pk PRIMARY KEY (tytul_id)
);

CREATE TABLE Ksiazka (
    ksiazka_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY,
    numer INTEGER NOT NULL,
    id_tytul INTEGER NOT NULL,
    CONSTRAINT ksiazka_pk PRIMARY KEY (ksiazka_id)
);
  
```

Execution results:

- Line 2, column 1: Executed successfully in 0,062 s, 0 rows affected.
- Line 11, column 1: Executed successfully in 0,047 s, 0 rows affected.
- Execution finished after 0,109 s, 0 error(s) occurred.

Poniżej opisano utworzone tabele oraz możliwe do utworzenia klasy typu **Entity**.

Tabela	Entity	Opis
Tytul_ksiazki	Tytul_ksiazki	Przechowują dane tytułu książki; relacja 1 do wiele z tabelą (encją) Ksiazka
Ksiazka	Ksiazka	Przechowują numer książki Relacja wiele do jeden z tabelą (encją) Tytul_ksiazki

Poniżej pokazano w oknie edytora pliku wczytany drugi skrypt **katalogksiazek1_insert.sql**, podobnie jak skrypt **katalogksiazek1.sql**. Należy go uruchomić w taki sam sposób jak skrypt **katalogksiazek1.sql** w celu wstawienia po jednej krotce do każdej z dwóch tabel.

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

- Projects:** Shows the connection to `jdbc:derby://localhost:1527/Katalogksiazek1` and the schema **KATALOGKSIAZEK1**.
- Tables:** Displays the same tables as the previous screenshot: **KSIAZKA** and **TYTUL_KSIAZKI**.
- Source:** Editor window containing the `katalogksiazek1_insert.sql` script.
- Output:** Shows the execution results of the insert statements.

```

SET SCHEMA Katalogksiazek1;

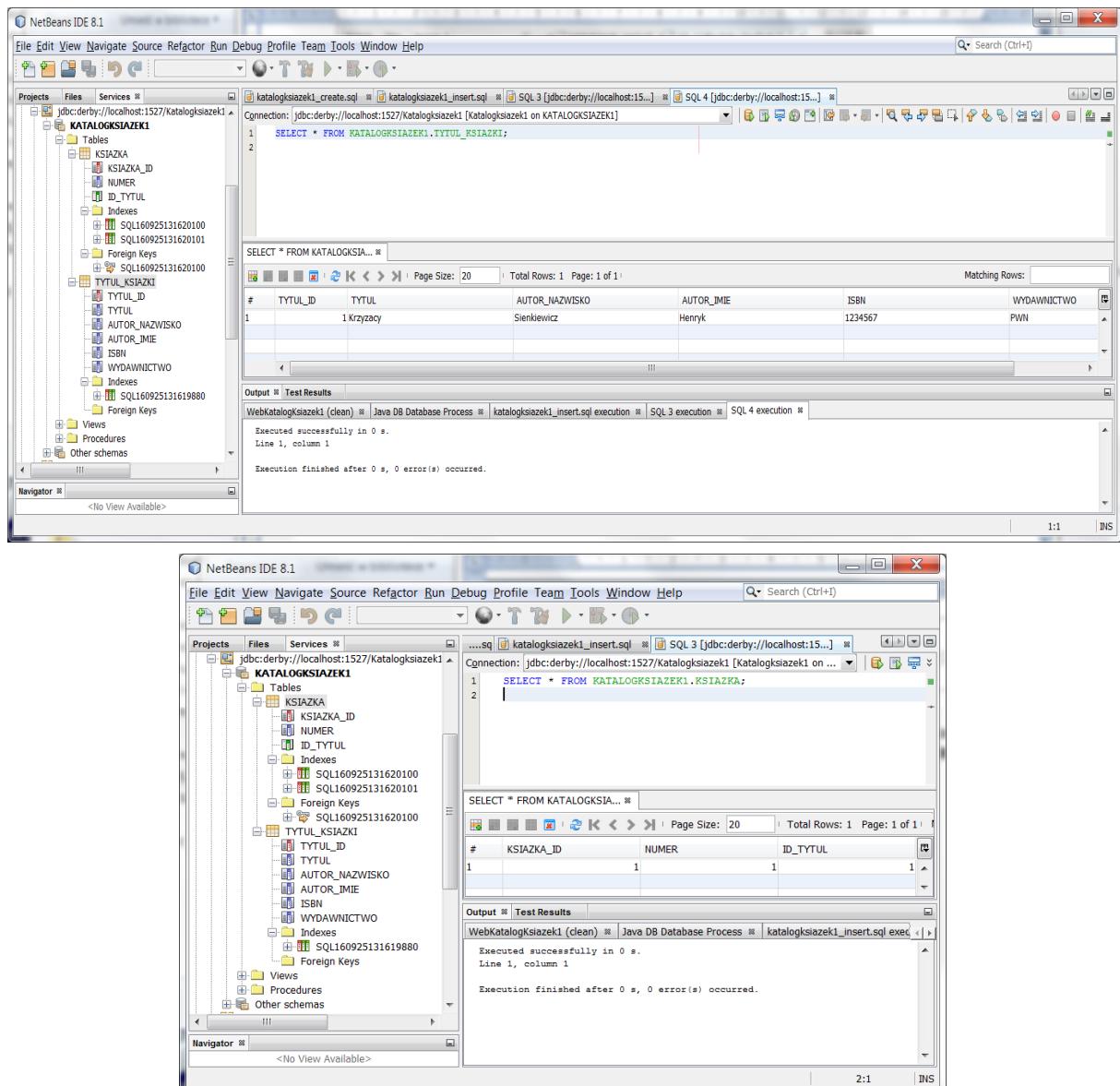
INSERT INTO Tytul_ksiazki (tytul, autor_nazwisko, autor_imie, ISBN, wydawnictwo)
VALUES ('Krzyzacy', 'Sienkiewicz', 'Henryk', '1234567', 'PWN');

INSERT INTO Ksiazka (numer,id_tytul) VALUES (1,1);
  
```

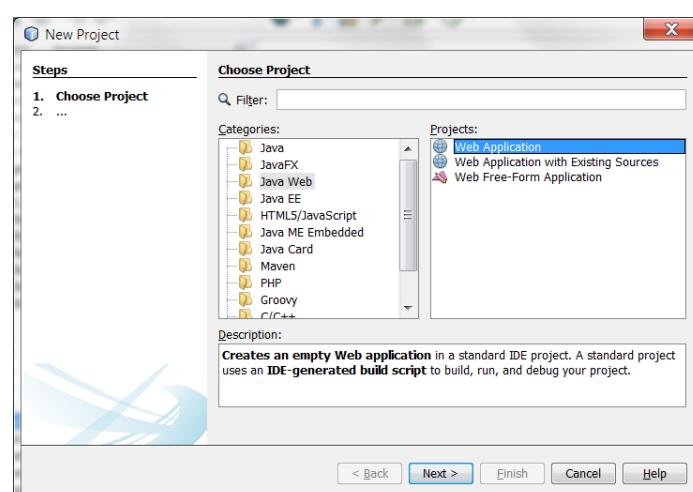
Execution results:

- Line 1, column 1: Executed successfully in 0 s, 0 rows affected.
- Line 3, column 1: Executed successfully in 0,047 s, 1 rows affected.
- Line 6, column 1: Executed successfully in 0,015 s, 1 rows affected.
- Execution finished after 0,062 s, 0 error(s) occurred.

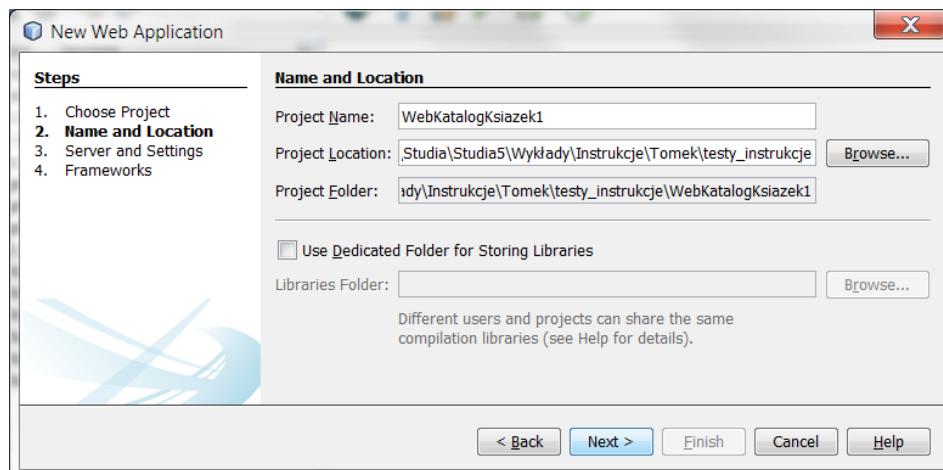
W celu odczytania zawartości tabel należy w oknie zakładki **Service**(jak poniżej pokazano) kliknąć prawym klawiszem myszy kolejno na każdą z nich i wybrać pozycję **View Data...**, co uruchamiania polecenie **SELECT * FROM** i pozwala wyświetlić zawartość tabeli.



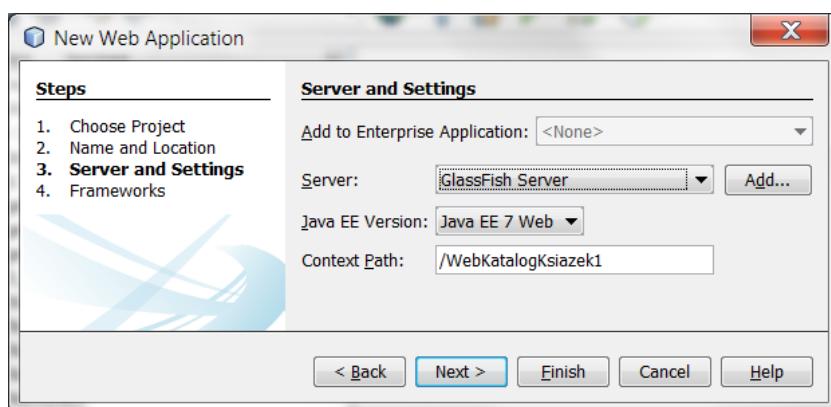
2.3. Tworzenie aplikacji typu **Web Application** – wybrać kolejno **File/New Project/Java Web/Web Application/Next**.



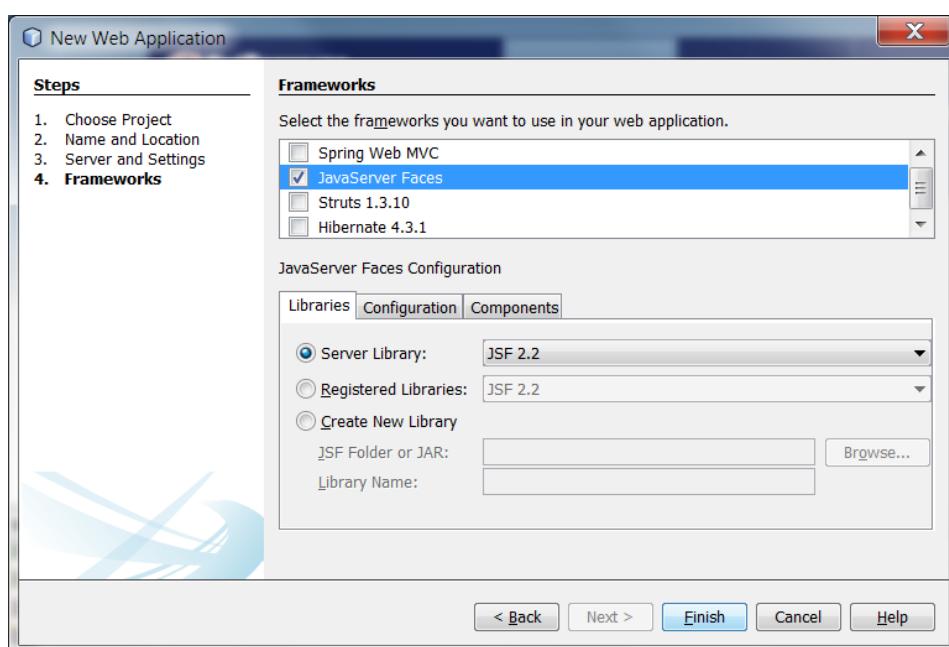
Poniżej pokazano nadanie nazwy projektowi **WebKatalogKsiazek1** oraz wybór położenia projektu. W celu kontynuacji należy kliknąć na przycisk **Next**.



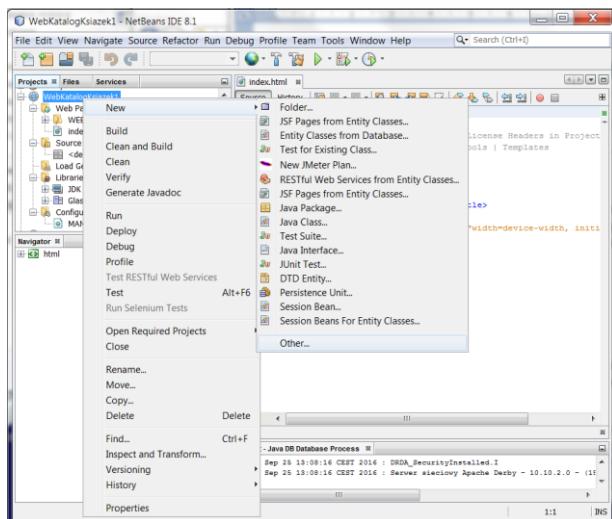
Poniżej pokazano wybór serwera aplikacji, platformy **Java EE 7 Web** oraz względnej ścieżki do projektu. W celu kontynuacji należy kliknąć na przycisk **Next**.



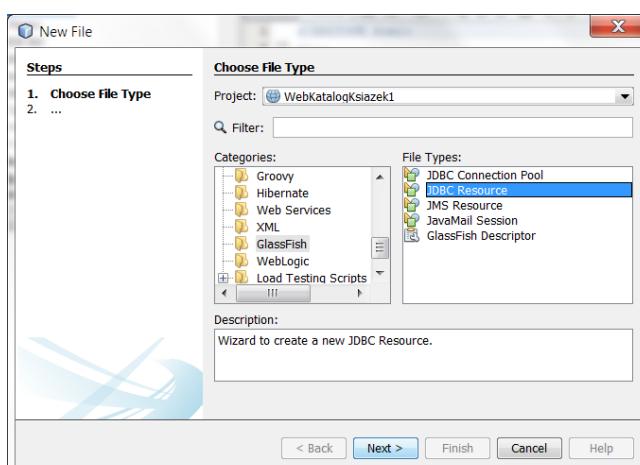
Poniżej pokazano wybór technologii **Java Server Faces**. Po kliknięciu na przycisk **Finish** zostanie wykonany projekt **Java Web Application**.



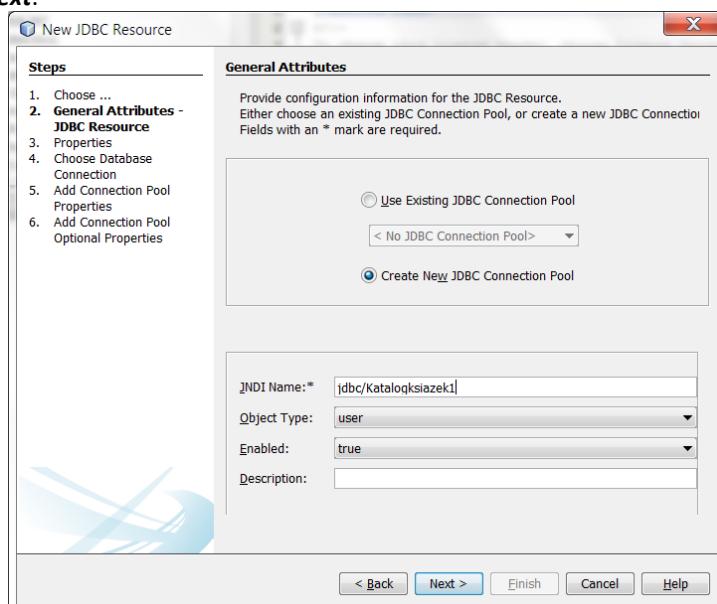
2.4. Wykonanie odniesienia typu **JNDI** do bazy danych **Katalogksiazek1** oraz puli połączeń do bazy danych: wybór **New** z listy po kliknięciu prawym klawiszem myszy na nazwę projektu wykonanego w p.2.5 w zakładce **Projects**, następnie wybór z kolejnej listy pozycji **Other**.



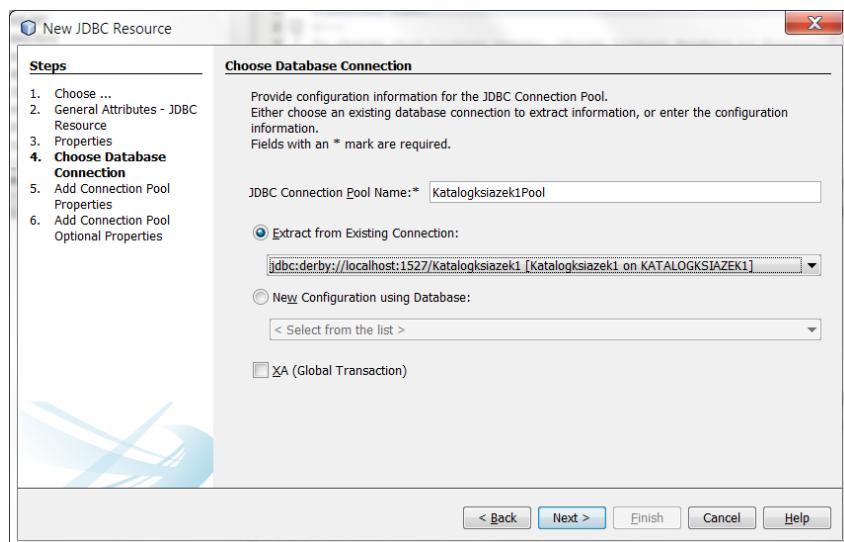
W formularzu **New File** należy wybrać z listy **Categories** pozycję **GlassFish**, następnie z listy **FileTypes** pozycję **JDBC Resource** i kliknąć na przycisk **Next**.



W kolejnych krokach należy zaznaczyć **Radio Button: Create New JDBC Connection Pool** oraz podać **JNDI Name** i kliknąć na przycisk **Next**.

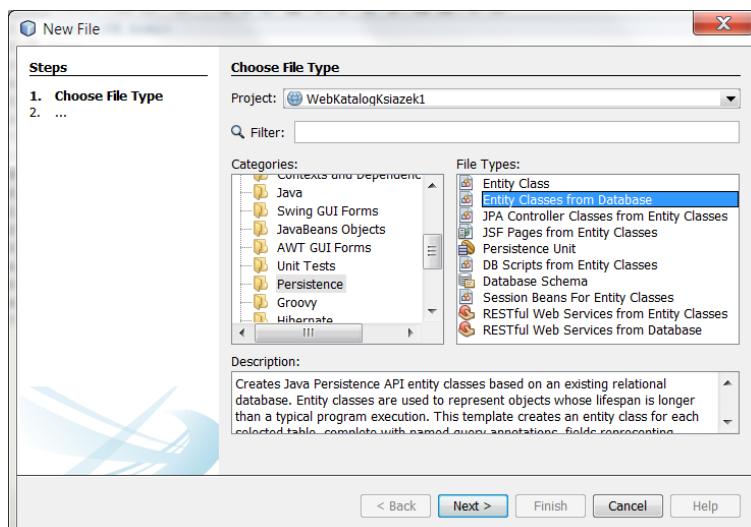


Na kolejnym formularzu należy wybrać połączenie do wykonanej w poprzednich krokach bazy danych w liście **Extract from Existing Connection**, a w polu **JDBC Connection Pool Name** należy podać nazwę tzw. puli połączeń do bazy danych, która poprawia wydajność dostępu do bazy danych i następnie kliknąć na przycisk **Next**.

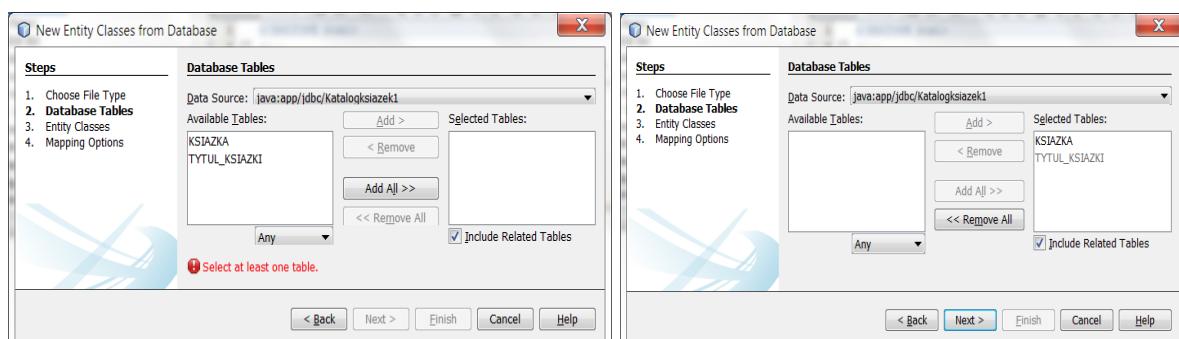


Po przejściu na kolejny formularz za pomocą kliknięcia na przycisk **Next** należy kliknąć na klawisz **Finish** bez wypełniania tego kolejnego formularza.

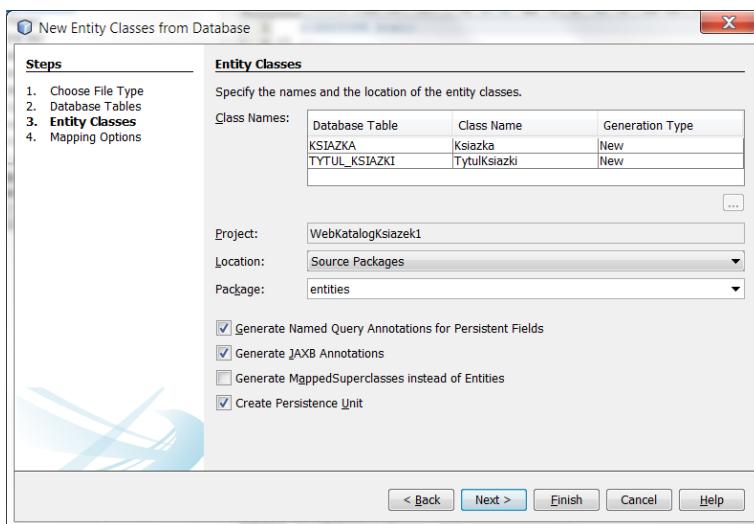
2.5. Wykonanie klas typu **Entity** z bazy danych **Katalogksiazek1** – w tym celu należy wybrać pozycję **New** z listy po kliknięciu prawym klawiszem myszy na nazwę projektu w zakładce **Projects**, następnie wybór z kolejnej listy pozycji **Other**. W formularzu **New File** (poniżej) należy wybrać: **Persistence/Entity Classes from Database**.



Wykonanie klas typu **Entity** z bazy danych – wybór odniesienia typu **JNDI** do bazy danych **Katalogksiazek1**: **java:app/jdbc/Katalogksiazek1** w liście **Data Source** i z listy **Available Tables** wybrać obie tabele **KSIAZKA** oraz **TYTUL_KSIAZKI** za pomocą przycisku **Add All>>**. Następnie należy kliknąć na klawisz **Next**.



Poniżej kolejny formularz prezentuje wykonane klas typu **Entity** na podstawie wybranych wcześniej tabel z bazy danych. Należy w polu **Package** wpisać nazwę pakietu, w którym będą przechowywane utworzone klasy. Na koniec należy kliknąć na przycisk **Finish**.



Kod wygenerowanej klas typu **Entity**: **TytulKsiazki** oraz **Ksiazka**

```
package entities;
import java.io.Serializable;
import java.util.Collection;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistenceNamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;

@Entity
@Table(name = "TYTUL_KSIAZKI")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "TytulKsiazki.findAll", query = "SELECT t FROM TytulKsiazki t"),
    @NamedQuery(name = "TytulKsiazki.findByTytulId", query = "SELECT t FROM TytulKsiazki t WHERE t.tytulId = :tytulId"),
    @NamedQuery(name = "TytulKsiazki.findByTytul", query = "SELECT t FROM TytulKsiazki t WHERE t.tytul = :tytul"),
    @NamedQuery(name = "TytulKsiazki.findByAutorNazwisko", query = "SELECT t FROM TytulKsiazki t WHERE t.autorNazwisko = :autorNazwisko"),
    @NamedQuery(name = "TytulKsiazki.findByAutorImie", query = "SELECT t FROM TytulKsiazki t WHERE t.autorImie = :autorImie"),
    @NamedQuery(name = "TytulKsiazki.findByIsbn", query = "SELECT t FROM TytulKsiazki t WHERE t.isbn = :isbn"),
    @NamedQuery(name = "TytulKsiazki.findByWydawnictwo", query = "SELECT t FROM TytulKsiazki t WHERE t.wydawnictwo = :wydawnictwo"))
public class TytulKsiazki implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "TYTUL_ID")
    private Integer tytulId;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 50)
    @Column(name = "TYTUL")
    private String tytul;
```

```

@Basic(optional = false)
@NotNull
@Size(min = 1, max = 50)
@Column(name = "AUTOR_NAZWISKO")
    private String autorNazwisko;
@Basic(optional = false)
@NotNull
@Size(min = 1, max = 50)
    private String autorImie;
@Basic(optional = false)
@NotNull
@Column(name = "AUTOR_IMIE")
@Size(min = 1, max = 50)
@Column(name = "ISBN")
    private String isbn;

@Basic(optional = false)
@NotNull
@Size(min = 1, max = 50)
@Column(name = "WYDAWNICTWO")
    private String wydawnictwo;
@OneToMany ksiazkaCollection; (cascade = CascadeType.ALL, mappedBy = "idTytul")
    private Collection<Ksiazka>

public TytulKsiazki() { }

public TytulKsiazki(Integer tytulId) { this.tytulId = tytulId; }

public TytulKsiazki(Integer tytulId, String tytul, String autorNazwisko, String autorImie, String isbn, String
    wydawnictwo)
{this.tytulId = tytulId;
this.tytul = tytul;
this.autorNazwisko = autorNazwisko;
this.autorImie = autorImie;
this.isbn = isbn;
this.wydawnictwo = wydawnictwo; }

public Integer getTytulId() { return tytulId; }
public void setTytulId(Integer tytulId) { this.tytulId = tytulId; }
public String getTytul() { return tytul; }
public void setTytul(String tytul) { this.tytul = tytul; }
public String getAutorNazwisko() { return autorNazwisko; }
public void setAutorNazwisko(String autorNazwisko) { this.autorNazwisko = autorNazwisko; }
public String getAutorImie() { return autorImie; }
public void setAutorImie(String autorImie) { this.autorImie = autorImie; }
public String getIsbn() { return isbn; }
public void setIsbn(String isbn) { this.isbn = isbn; }
public String getWydawnictwo() { return wydawnictwo; }
public void setWydawnictwo(String wydawnictwo) { this.wydawnictwo = wydawnictwo; }

@XmlTransient
public Collection<Ksiazka> getKsiazkaCollection() { return ksiazkaCollection; }
public void setKsiazkaCollection(Collection<Ksiazka> ksiazkaCollection) {
    this.ksiazkaCollection = ksiazkaCollection; }

@Override
public int hashCode() {
    int hash = 0;
    hash += (tytulId != null ? tytulId.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    if (!(object instanceof TytulKsiazki)) {
        return false; }
    TytulKsiazki other = (TytulKsiazki) object;
    if ((this.tytulId == null && other.tytulId != null) || (this.tytulId != null && !this.tytulId.equals(other.tytulId)))
        return false;
    return true;
}

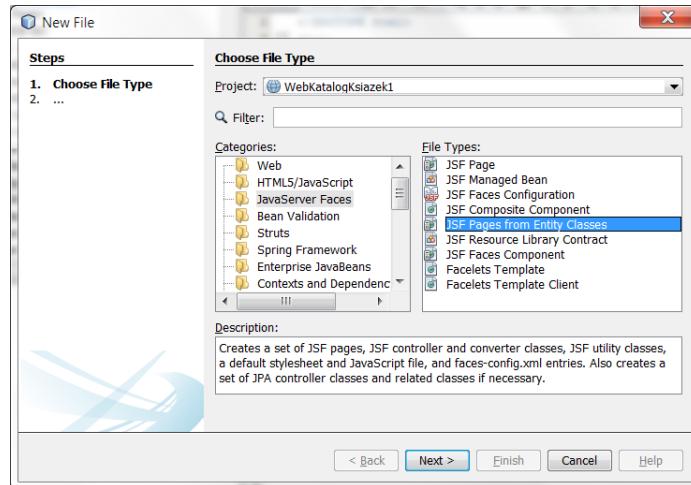
@Override
public String toString() {
    return "entities.TytulKsiazki[ tytulId=" + tytulId + "]";
}
}

```

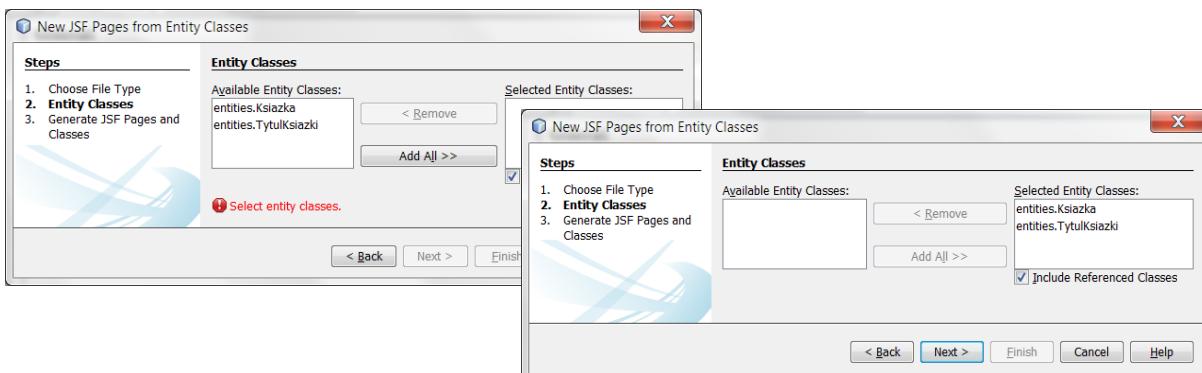
```
package entities;
import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.xml.bind.annotation.XmlRootElement;

@Entity
@Table(name = "KSIAZKA")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Ksiazka.findAll", query = "SELECT k FROM Ksiazka k"),
    @NamedQuery(name = "Ksiazka.findByKsiazkald", query = "SELECT k FROM Ksiazka k WHERE k.ksiazkald = :ksiazkald"),
    @NamedQuery(name = "Ksiazka.findByNumer", query = "SELECT k FROM Ksiazka k WHERE k.numer = :numer")})
public class Ksiazka implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "KSIAZKA_ID")
    private Integer ksiazkald;
    @Basic(optional = false)
    @NotNull
    @Column(name = "NUMER")
    private int numer;
    @JoinColumn(name = "ID_TYTUL", referencedColumnName = "TYTUL_ID")
    @ManyToOne(optional = false)
    private TytulKsiazki idTytul;
    public Ksiazka() { }
    public Ksiazka(Integer ksiazkald) { this.ksiazkald = ksiazkald; }
    public Ksiazka(Integer ksiazkald, int numer) {
        this.ksiazkald = ksiazkald;
        this.numer = numer; }
    public Integer getKsiazkald() { return ksiazkald; }
    public void setKsiazkald(Integer ksiazkald) { this.ksiazkald = ksiazkald; }
    public int getNumer() { return numer; }
    public void setNumer(int numer) { this.numer = numer; }
    public TytulKsiazki getIdTytul() { return idTytul; }
    public void setIdTytul(TytulKsiazki idTytul) { this.idTytul = idTytul; }
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (ksiazkald != null ? ksiazkald.hashCode() : 0);
        return hash; }
    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Ksiazka)) return false;
        Ksiazka other = (Ksiazka) object;
        if ((this.ksiazkald == null && other.ksiazkald != null) || (this.ksiazkald != null && !this.ksiazkald.equals(other.ksiazkald)))
            return false;
        return true; }
    @Override
    public String toString() { return "entities.Ksiazka[ ksiazkald=" + ksiazkald + "]"; }
}
```

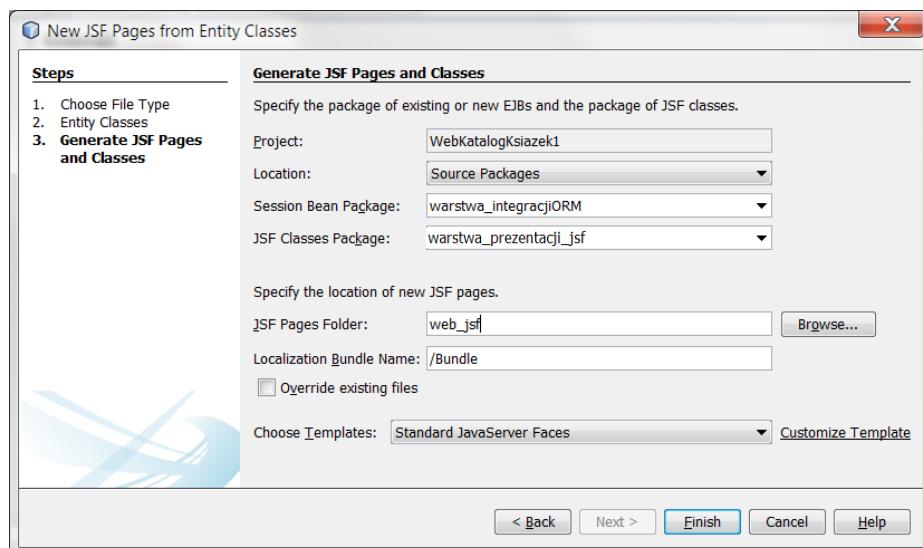
2.7. Generowanie **stron JSF** z encji (oraz komponentów EE) – wybór **New** z listy po kliknięciu prawym klawiszem myszy na nazwę projektu w zakładce **Projects**, następnie wybór pozycji **Other** z kolejnej listy. W formularzu **New File** należy wybrać pozycje **JavaServer Faces/JSF Pages from Entity Classes** i kliknąć na przycisk **Next**.



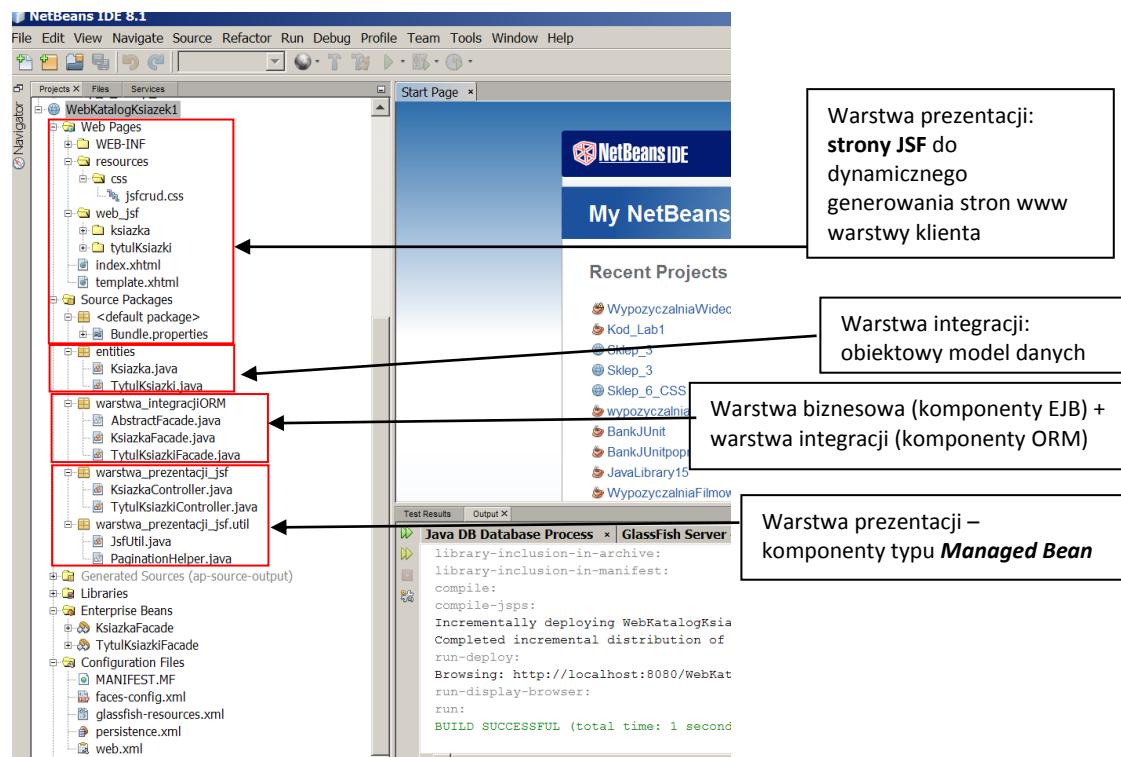
Na kolejnym formularzu należy wybrać z listy **Available Entity Classes** wszystkie klasy za pomocą klawisza **Add All** i następnie kliknąć na klawisz **Next**.



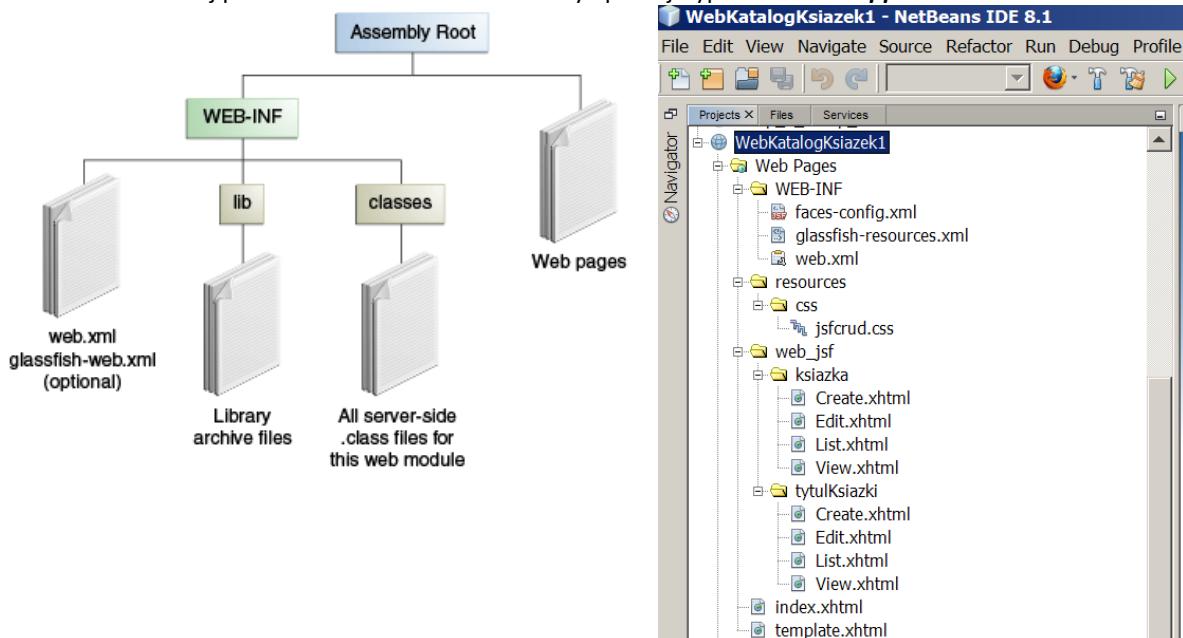
Poniżej przedstawiono widok formularza **New JSF Pages from Entity Classes** po wyborze encji i kliknięciu na przycisk **Next**. Na tym formularzu należy w polu **Session Bean Package** podać nazwę warstwy biznesowej, która jednocześnie pełni rolę warstwy integracji, w polu **JSF Classes Package** podać nazwę warstwy prezentacji lub inaczej warstwy internetowej, a w polu **JSF Pages Folder**, nazwę związaną z warstwą internetową lub inaczej – z warstwą prezentacji. Następnie należy kliknąć na przycisk **Finish**.



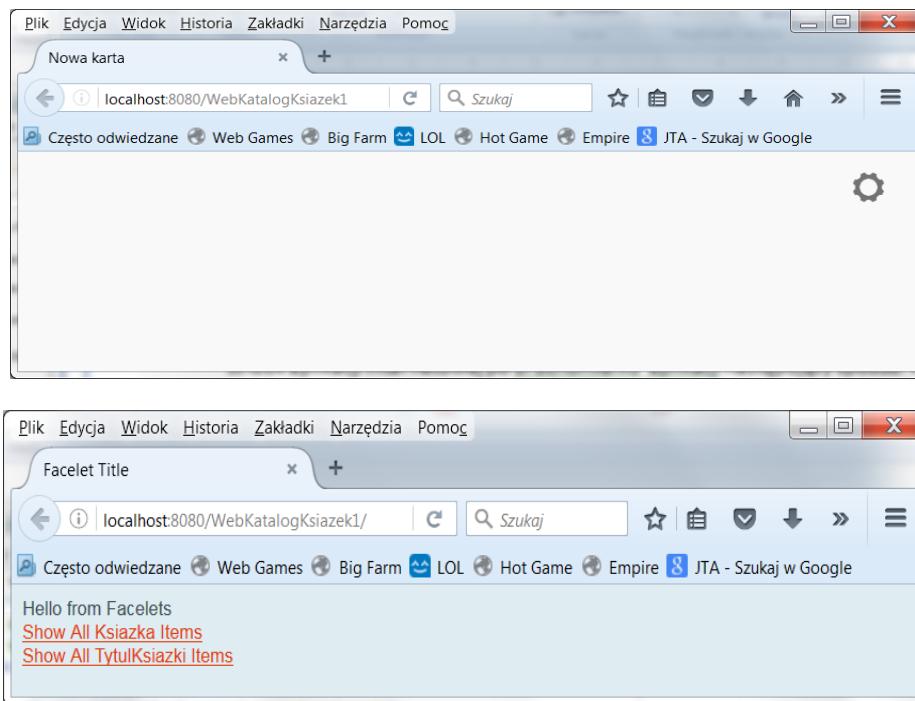
Poniżej pokazano zawartość katalogu wykonanego projektu aplikacji internetowej typu ***Web Application***.



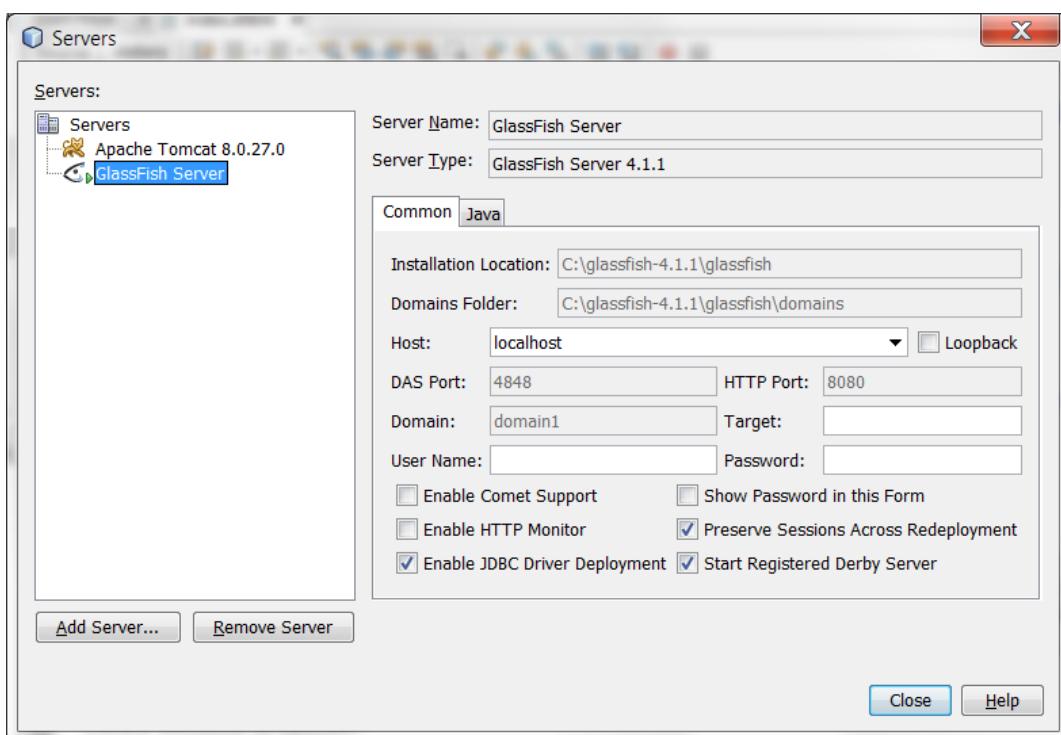
Poniżej przedstawiono schemat struktury aplikacji typu ***Java Web Application***.



2.8. Widok aplikacji internetowej po jej uruchomieniu w następujący sposób: w zakładce **Projects** kliknąć prawym klawiszem myszy na nazwę projektu i wybrać z listy kolejno **Clean and Build/Deploy**. Następnie należy otworzyć okno w przeglądarce **FireFox Mozilla** i wpisać następujący adres strony www: **localhost:8080/WebKatalogKsiazek1**.



Port 8080 jest portem domyślnym serwera typu **GlassFish**. Należy go sprawdzić, przed uruchomieniem aplikacji, w środowisku **NetBeans 8.1** wybierając w **Menu Bar** opcje **Tools/Servers** i na formularzu **Servers**: z pozycji **GlassFish Server** odczytać port podany w polu **HTTP Port** (rysunek poniżej). Wartość tego portu należy wpisać w adresie strony www. Innym sposobem uruchomienia aplikacji w oknie przeglądarki typu **FireFox** jest przypisanie do niej domyślnego dostępu w środowisku **NetBeans 8.1**: należy w oknie **Tools/Options**, w polu **WebBrowser** przypisać wybraną przeglądarkę **FireFox** z listy przeglądarek. Wtedy podczas uruchamiania aplikacji typu **Web Application** wykonać **Clean and Build/Run** (lub **Clean and Build/Deploy/Run** w przypadku problemów z uruchomieniem strony głównej aplikacji).

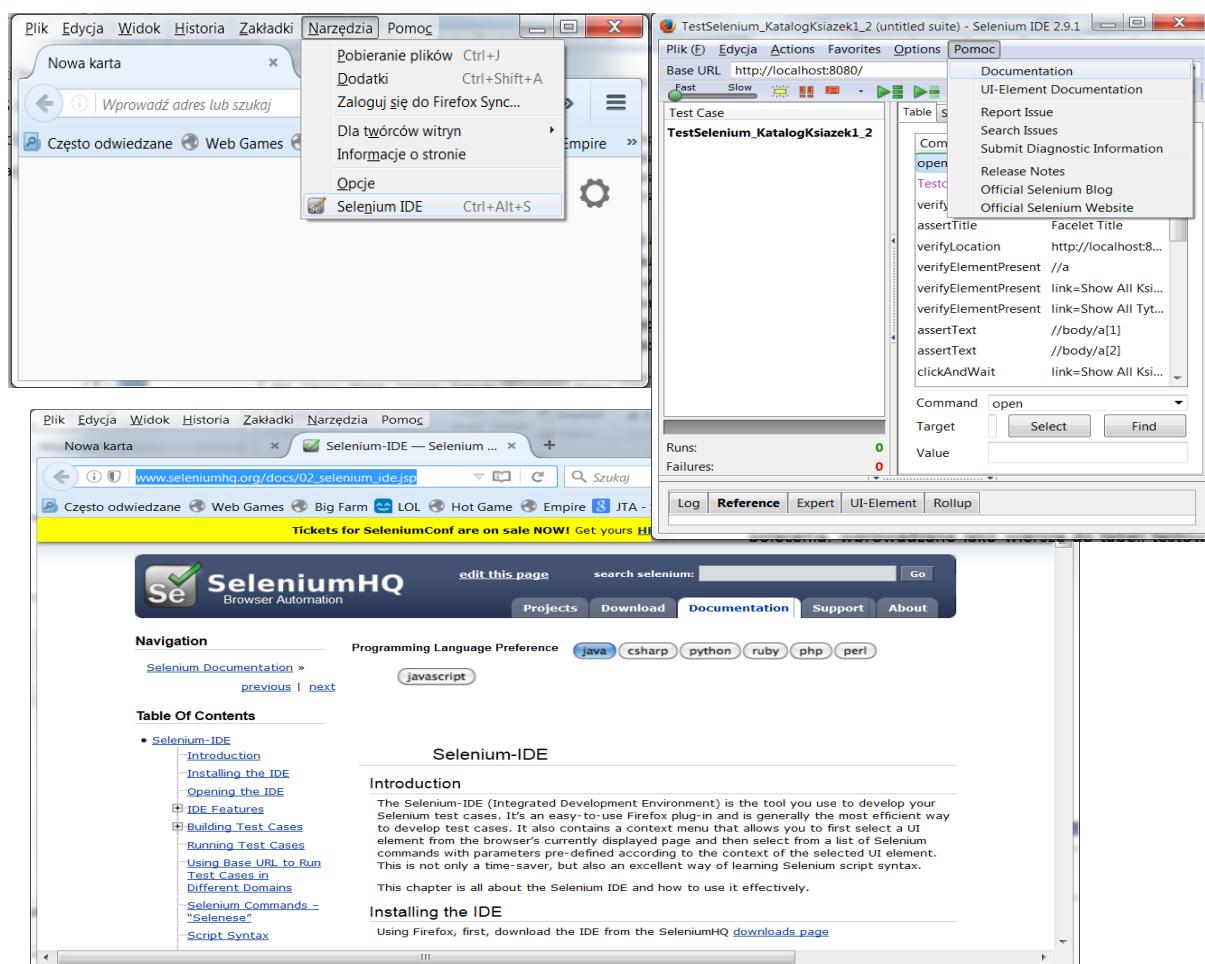


Dodatek 2

Testowanie funkcjonalne aplikacji internetowej za pomocą narzędzia Selenium IDE.

1. W celu zapoznania się ze sposobem instalacji **Selenium IDE**, z programowaniem testów oraz innymi ważnymi informacjami, należy skorzystać z dokumentacji tego środowiska na stronie http://www.seleniumhq.org/docs/02_selenium_ide.jsp.

Podczas budowy testów dokumentacja jest dostępna ze strony narzędzia **Selenium IDE**. Poniżej pokazano uruchomioną przeglądarkę **FireFox**, w której jest już zainstalowane narzędzie **Selenium IDE**. Narzędzie uruchamia się w **Menu Bar**, gdzie należy kliknąć na pozycję **Narzędzia**, a następnie na pozycję **Selenium IDE**. W oknie uruchomionego narzędzia **Selenium IDE** należy kliknąć na pozycję **Pomoc** w **Menu Bar** i następnie na pozycję **Dokumentacja**. Pojawi się strona o adresie podanym powyżej, która w bardzo przystępny sposób wyjaśnia, co można testować oraz w jaki sposób to wykonać – pokazana poniżej.

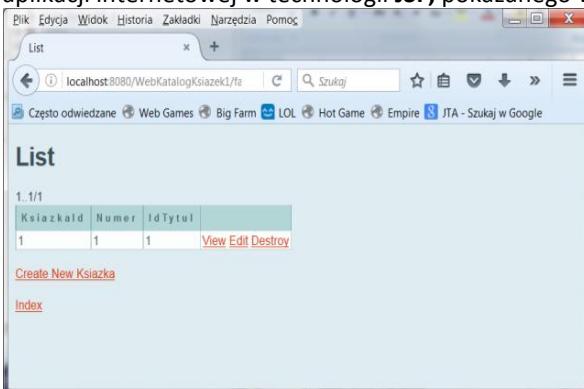


Poniżej podano wykaz linków udostępnionych ze strony prezentującej dokumentację **Selenium IDE**.

- [Selenium-IDE](#)
- [Introduction](#)
- [Installing the IDE](#)
- [Opening the IDE](#)
- [IDE Features](#)
- [Building Test Cases](#)
- [Running Test Cases](#)
- [Using Base URL to Run Test Cases in Different Domains](#)
- [Selenium Commands – “Selenese”](#)
- [Script Syntax](#)
- [Test Suites](#)
- [Commonly Used Selenium Commands](#)
- [Verifying Page Elements](#)
- [Assertion or Verification?](#)

- [Locating Elements](#)
- [Matching Text Patterns](#)
- [The “AndWait” Commands](#)
- [The waitFor Commands in AJAX applications](#)
- [Sequence of Evaluation and Flow Control](#)
- [Store Commands and Selenium Variables](#)
- [JavaScript and Selenese Parameters](#)
- [echo - The Selenese Print Command](#)
- [Alerts, Popups, and Multiple Windows](#)
- [Debugging](#)
- [Writing a Test Suite](#)
- [User Extensions](#)
- [Format](#)
- [Executing Selenium-IDE Tests on Different Browsers](#)
- [Troubleshooting](#)

Aby testować zawartość tabeli z książkami na formularzu strony **List.html**, pokazanego poniżej, potrzebna jest nazwa tabeli jako atrybut ***id*** znacznika **<table>** w języku **xhtml**. Taki atrybut nie został wygenerowany podczas automatycznego tworzenia aplikacji internetowej w technologii **JSF**, pokazanego w **Dodatku 1**.



Poniżej pokazano stronę **JSF** o nazwie **List.xhtml**, która stanowi wzorzec do wygenerowania strony **List.html** w fazie **Response** i wysłanie jej do przeglądarki (p.1 **Dodatku 1**). Po dodaniu atrybutu ***id="książki"*** w znaczniku JSF **<h: dataTable>**, nazwa tabeli może być prezentowana przez ten znacznik uzupełniony podczas generowania strony html.

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:c="http://xmlns.jcp.org/jsf/composite">
    <ui:composition template="/template.xhtml">
        <ui:define name="title">
            <h:outputText value="#{bundle.ListKsiazkaTitle}"></h:outputText>
        </ui:define>
        <ui:define name="body">
            <h:form styleClass="jsfcrud_list_form">
                <h:panelGroup id="messagePanel" layout="block">
                    <h:messages errorStyle="color: red" infoStyle="color: green" layout="table"/>
                </h:panelGroup>
                <h:outputText escape="false" value="#{bundle.ListKsiazkaEmpty}" rendered="#{ksiazkaController.items.rowCount == 0}">
                <h:panelGroup rendered="#{ksiazkaController.items.rowCount > 0}">
                    <h:outputText value="#{ksiazkaController.pagination.pageFirstItem + 1}..#{ksiazkaController.pagination.pageLastItem}>
                    <h:commandLink action="#{ksiazkaController.previous}" value="#{bundle.Previous}" #ksiazkaController.pagination.pageS...>
                    <h:commandLink action="#{ksiazkaController.next}" value="#{bundle.Next}" #ksiazkaController.pagination.pageSize" re...
                    <h: dataTable value="#{ksiazkaController.items}" var="item" border="0" cellpadding="2" id="książki" cellspacing="0" r...
                        <h:column>
                            <f:facet name="header">
                                <h:outputText value="#{bundle.ListKsiazkaTitle_ksiazkaId}" />
                            </f:facet>
                            <h:outputText value="#{item.ksiazkaId}" />
                        </h:column>
                </h:panelGroup>
            </h:form>
        </ui:define>
    </ui:composition>

```

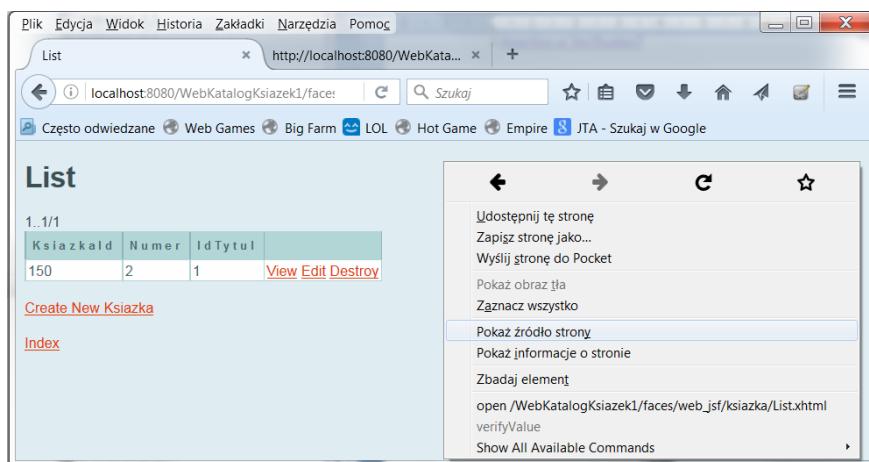
Podobnie nadano ***id="książka"*** w znaczniku **<h:panelGrid>** w pliku **View.xhtml** w tym samym katalogu.

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:c="http://xmlns.jcp.org/jsf/composite">
    <ui:composition template="/template.xhtml">
        <ui:define name="title">
            <h:outputText value="#{bundle.ViewKsiazkaTitle}"></h:outputText>
        </ui:define>
        <ui:define name="body">
            <h:form>
                <h:panelGrid columns="2" id="książka">
                    <h:outputText value="#{bundle.ViewKsiazkaLabel_ksiazkaId}" />
                    <h:outputText value="#{ksiazkaController.selected.ksiazkaId}" title="#{bundle.ViewKsiazkaTitle_ksiazkaId}" />
                    <h:outputText value="#{bundle.ViewKsiazkaLabel_numer}" />
                    <h:outputText value="#{ksiazkaController.selected.numer}" title="#{bundle.ViewKsiazkaTitle_numer}" />
                    <h:outputText value="#{bundle.ViewKsiazkaLabel_idTytul}" />
                    <h:outputText value="#{ksiazkaController.selected.idTytul}" title="#{bundle.ViewKsiazkaTitle_idTytul}" />
                </h:panelGrid>
                <br />
                <h:commandLink action="#{ksiazkaController.destroyAndView}" value="#{bundle.ViewKsiazkaDestroyLink}" />
            </h:form>
        </ui:define>
    </ui:composition>

```

Aby poznać dane znaczników wygenerowanej strony **html**, należy kliknąć prawym klawiszem myszy na daną stronę aplikacji i następnie wybrać pozycję **Pokaż źródło strony**.



Poniżej pokazano tekst strony **List.html** napisanej w języku **xhtml**. Znacznik **<table>** zawiera atrybut **id="j_idt11:ksiazki"**, który służyć będzie do identyfikacji komórek tabeli podczas testowania.

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"><head id="j_idt2">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>List</title><link type="text/css" rel="stylesheet" href="/WebKatalogKsiazek1/...
<h1>List</h1>
<p>
<form id="j_idt11" name="j_idt11" method="post" action="/WebKatalogKsiazek1/faces/web_jsf/ks...
<input type="hidden" name="j_idt11" value="j_idt11" />
<div id="j_idt11:messagePanel"></div><!--<!-->
<table id="j_idt11:ksiazki" border="0" cellpadding="2" cellspacing="0" r...
<thead>
<tr>
<th scope="col">KsiazkaId</th>
<th scope="col">Numer</th>
<th scope="col">IdTytul</th>
<th scope="col">Operacje</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>2</td>
<td>1</td>
<td><a href="#">View</a> <a href="#">Edit</a> <a href="#">Destroy</a></td>
</tr>
</tbody>
</table>
<div id="j_idt11:bottomPanel"></div>

```

Poniżej pokazano tekst strony **View.html** napisanej w języku **xhtml**. Znacznik **<table>** zawiera atrybut **id="j_idt12:ksiazka"**, który służyć będzie do identyfikacji komórek tabeli podczas testowania.

View

KsiazkaId: 228

Numer: 2

IdTytul: 1

[Destroy](#)

[Edit](#)

[Create New Ksiazka](#)

[Show All Ksiazka Items](#)

[Index](#)

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"><head id="j_idt2">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>View</title><link type="text/css" rel="stylesheet" href="/WebKatalogKsiazek1/...
<h1>View</h1>
<p><div id="messagePanel"></div></p>
<form id="j_idt12" name="j_idt12" method="post" action="/WebKatalogKsiazek1/faces/web_jsf/ks...
<input type="hidden" name="j_idt12" value="j_idt12" />
<table id="j_idt12:ksiazka">
<tbody>
<tr>
<td>KsiazkaId:</td>
<td><span title="KsiazkaId">228</span></td>
</tr>
<tr>
<td>Numer:</td>
<td><span title="Numer">2</span></td>
</tr>
<tr>
<td>IdTytul:</td>
<td><span title="IdTytul">1</span></td>
</tr>
</tbody>
</table>
<div id="j_idt12:bottomPanel"></div>

```

2. Testowanie wybranej funkcji aplikacji, której działanie opisano za pomocą scenariuszy przypadków użycia (instrukcja do lab1, p. 4.2, Nazwa PU: Dodaj_ksiazke)

Nazwa PU: Dodaj_ksiazke

Cel: Dodanie nowej książki

Warunki początkowe:

Uruchomienie programu jako aplikacji www lub aplikacji w architekturze typu "klient-serwer,"

Warunki końcowe:

Wprowadzenia danych książki o unikalnym numerze egzemplarza w ramach książek o tym samym tytule

Scenariusz:

1. Należy podać atrybuty tytułu: ISBN jako obowiązkowa, dlatego tworzony jest tytuł wzorcowy, w którym istotny jest tylko ISBN do wyszukiwania rzeczywistego tytułu
2. Należy wywołać **PU Sprawdz_czy_jest_tytul**. Należy sprawdzić, czy tytuł o podanym ISBN już istnieje. Jeśli nie, należy zakończyć PU.
3. Należy utworzyć egzemplarz zawierający numer podany do wyszukiwania egzemplarza i należy przekazać go do **PU Sprawdz_czy_jest_ksiazka**. Jeśli nie istnieje egzemplarz o danym numerze, należy wstawić ten egzemplarz, w przeciwnym wypadku należy zakończyć PU.

Nazwa PU: Sprawdz_czy_jest_ksiazka

Cel: Wyszukiwanie książki o podanym numerze

Warunki początkowe:

Przypadek użycia jest wywoływany z PU Dodaj_ksiazke

Warunki końcowe:

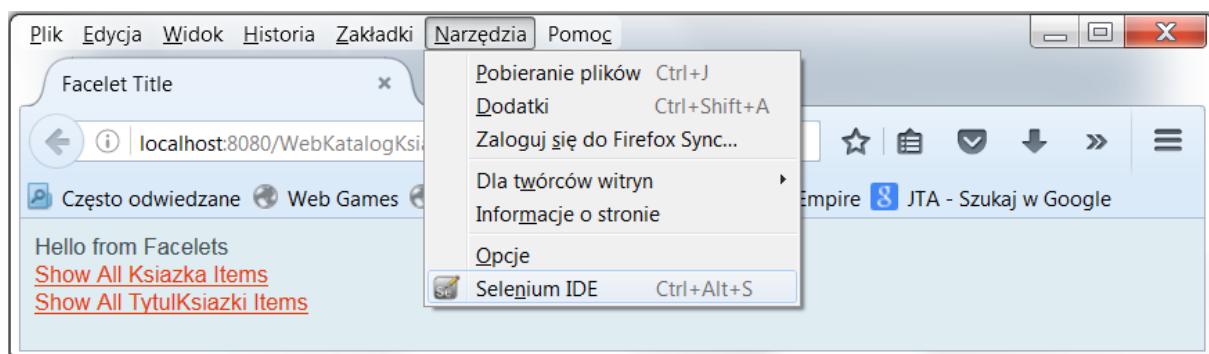
Zwraca wynik, określający, czy podany numer jest unikatowy lub podaje informację, że dany numer już istnieje

Scenariusz:

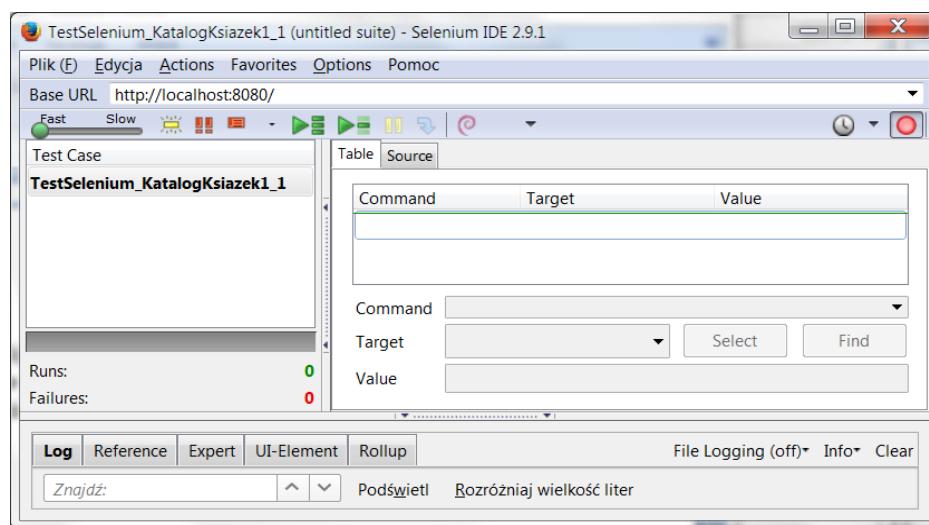
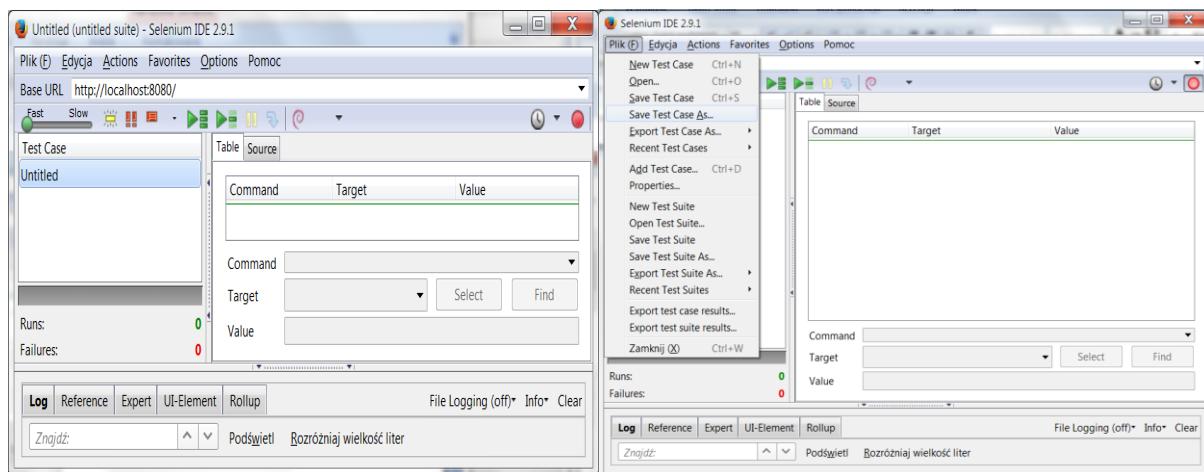
1. Szukanie książki przebiega według atrybutu: numer egzemplarza (obowiązkowo) zgodnie z danymi tytułu podanego do przypadku użycia. Przeszukiwane są egzemplarze należące do konkretnego tytułu
2. Jeśli istnieje egzemplarz o podanym numerze, zwracany jest egzemplarz z zasobów wypożyczalni, w przeciwnym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

3. Wykonanie testu działania wybranej funkcji aplikacji zgodnie ze scenariuszem przypadku użycia

3.1. **Pierwszy etap oparty na nagrywanie testu.** Należy uruchomić wykonaną aplikację w **Dodatku 1** tzn. po uruchomieniu narzędzia **NetBeans 8.1** należy uruchomić aplikację w następujący sposób: w zakładce **Projects** kliknąć prawym klawiszem myszy na nazwę projektu i wybrać z listy kolejno **Clean and Build/Deploy**. Po otwarciu przeglądarki **Firefox** firmy **Mozilla** należy wpisać w polu adresu: **localhost:8080/WebKatalogKsiazek1** (sposób uruchomienia podano w p.2.8 **Dodatku 1**). Następnie należy wybrać pozycję **Narzędzia/Selenium IDE**.

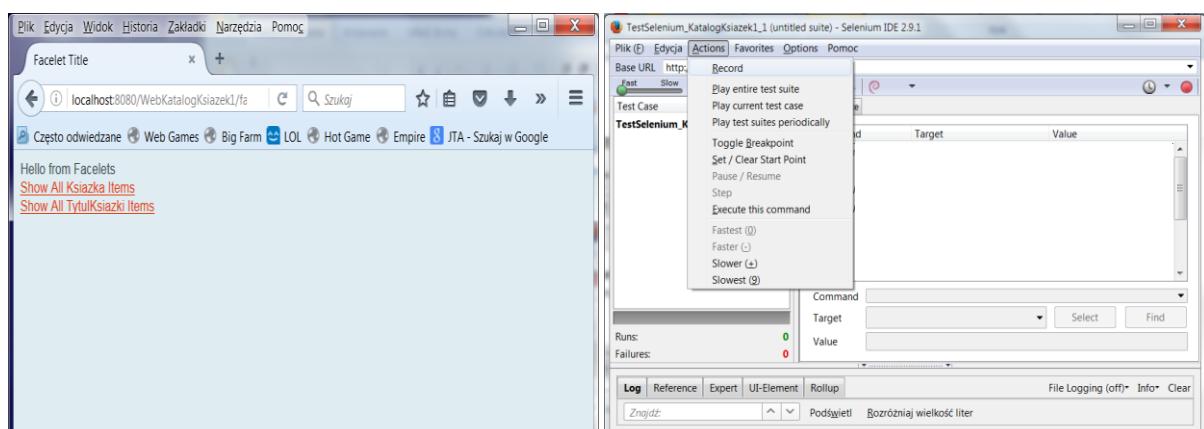


Poniżej pokazano formularz **Selenium IDE 2.9.1**. Należy wybrać **Plik(F)**, następnie pozycję **Save Test Case** lub **Save Test Case As** i następnie wybrać położenie pliku na dysku oraz nazwę pliku zawierającego test funkcjonalny. Po zatwierdzeniu w oknie **Selenium IDE 2.9.1** pokazała się nowa nazwa testu funkcjonalnego – w przykładzie jest to **TestSelenium_KatalogKsiazek1_1**.



Dalej pokazano przebieg nagrywania testu.

Poniżej pokazano uruchamianie testu: należy wybrać w **Menu Bar** pozycję **Action** i kliknąć na pozycję **Record** lub z belki narzędziowej kliknąć na ikonę typu „czerwone kółko”



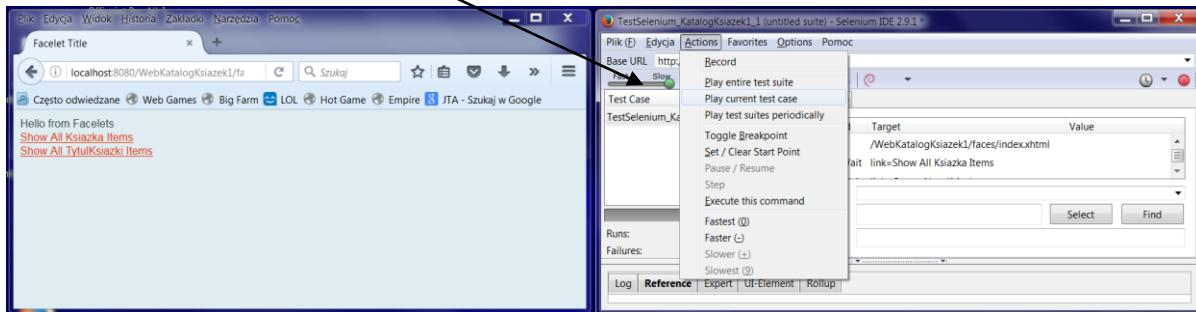
Poniżej na kolejnych rysunkach znajduje się prezentacja nagrania testu przebiegu wprowadzania nowego egzemplarza książki do tabeli **Książka**, wykonana przez użytkownika.

The figure consists of four screenshots of the Selenium IDE interface, illustrating the steps to add a new book record:

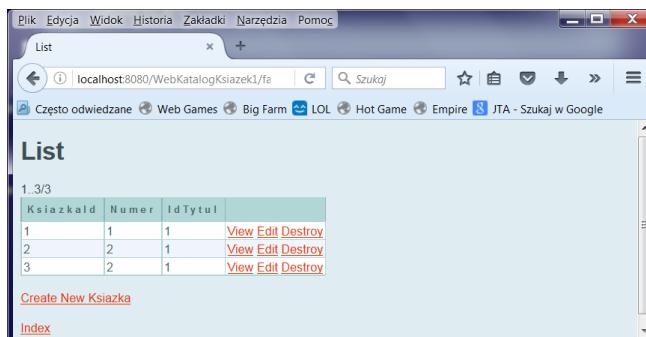
- Screenshot 1:** Shows the Selenium IDE toolbar and a test case named "TestSelenium_KatalogKsiazek1_1". The table panel shows the command "open /WebKatalogKsiazek1/faces/index.xhtml".
- Screenshot 2:** Shows the "Create New Ksiazka" page with fields for KsiazkaId (2), Numer (2), and IdTytul (entities.TytulKsiazki[tytulId=1]). The Selenium IDE test case table shows commands for clicking the "Create New Ksiazka" button and selecting the second item in the dropdown.
- Screenshot 3:** Shows the updated list of books after the new entry has been added. The Selenium IDE test case table now includes commands for saving the new entry and displaying all books again.
- Screenshot 4:** Shows the "Hello from Facelets" page with links to "Show All Ksiazka Items" and "Show All TytulKsiazki Items". The Selenium IDE test case table shows the final commands for these actions.

Nagrywanie testu należy zakończyć klikając na ikonę „czerwone kółko” lub wybierając w **Menu Bar** pozycję **Action/Record**.

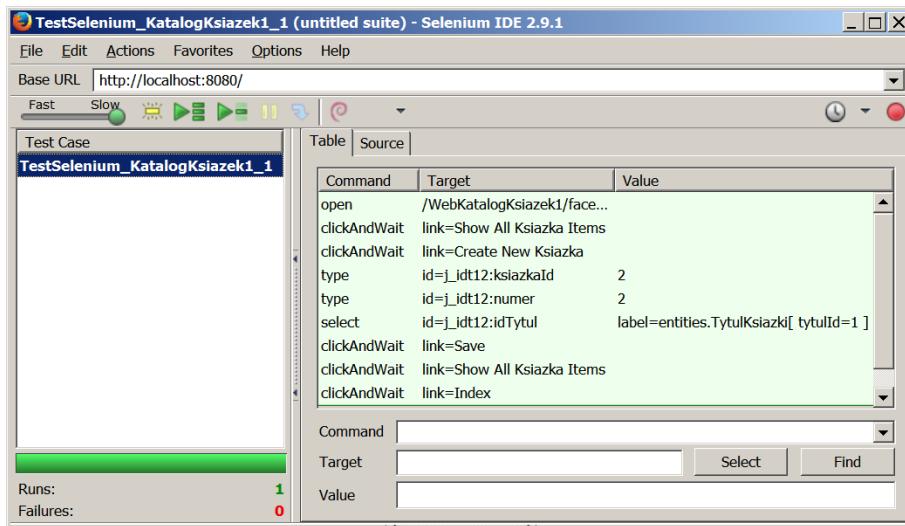
Odtworzenie nagranego testu należy wykonać klikając w **Menu Bar** na pozycję **Action/Play entire test case** lub z belki narzędziowej kliknąć na ikonę typu „zielony trójkąt”. Szybkość odtwarzania testu ustawia się za pomocą suwaka **Fast Slow**



Po odtworzeniu nagranego testu (gdzie ponownie wprowadzano te same dane do tabeli) widać (rysunek poniżej), że zapewnienie spójności danych jest ograniczone do zmiany klucza głównego w tabeli **Książka** przez silnik bazodanowy - w teście wprowadza się klucz główny równy 2, a silnik bazodanowy nadaje kolejnej krotce, zawierającej te same dane, inną wartość klucza głównego dbając jedynie o zachowanie integralności klucza głównego. Zostało to ustalone podczas generowania klas typu **Entity**, gdzie zostały automatycznie nadane następujące adnotacje atrybutowi Id klasom typu **Entity: @Id @GeneratedValue(strategy = GenerationType.IDENTITY)**. Obecnie można zapełnić całą tabelę takimi samymi danymi, które będą się różnić się jedynie wartością klucza głównego. To w przyszłości należy zmienić, gdyż jest to niezgodne ze scenariuszem przypadku użycia dodawania nowej książki (p.2 **Dodatek 2**), który powinien realizować testowany program tzn. każdy egzemplarz książki powinien mieć unikatowy numer, a w tabeli znajdują się dwa egzemplarze należące do tego samego tytułu książki, które mają ten sam numer egzemplarza.



Poniżej przedstawiono kod nagranego testu widocznego jako widok strony ***TestSelenium_KatalogKsiazek1_1.html***.



Poniżej pokazano kod źródłowy nagranego testu strony *TestSelenium_KatalogKsiazek1_1.html* w języku xhtml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="selenium.base" href="http://localhost:8080/" />
    <title>TestSelenium_KatalogKsiazek1_1</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">TestSelenium_KatalogKsiazek1_1</td></tr>
</thead><tbody>
<tr>
    <td>open</td>          <!--kolumna Command-->
    <td>/WebKatalogKsiazek1/faces/index.xhtml</td> <!--kolumna Target-->
    <td></td>           <!--kolumna Value-->
</tr>
<tr>
    <td>clickAndWait</td>
    <td>link=Show All Ksiazka Items</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>link/Create New Ksiazka</td>
    <td></td>
</tr>
<tr>
    <td>type</td>
    <td>id=j_idt12:ksiazkaid</td>
    <td>2</td>
</tr>
<tr>
    <td>type</td>
    <td>id=j_idt12:numer</td>
    <td>2</td>
</tr>
<tr>
    <td>select</td>
    <td>id=j_idt12:idTytul</td>
    <td>label=entities.TytulKsiazki[ tytulId=1 ]</td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>link=Save</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>link>Show All Ksiazka Items</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>link=Index</td>
    <td></td>
</tr>
</tbody></table>
</body></html>
```

3.2. Drugi etap: tworzenie ręczne testu funkcjonalnego do uproszczonego testowania:

- elementów typu UI wybranej strony aplikacji internetowej oraz
- zachowania spójności danych przez aplikację internetową oraz jej wybranych funkcjonalności.

Wykonano nowy test typu **Test Case**, podobnie jak w p.3.1. W przykładzie jest to plik **TestSelenium_KatalogKsiazek1_2.html**. Następnie, ręcznie wpisano polecenia **Selenium IDE** w celu sprawdzenia budowy wybranej głównej strony internetowej aplikacji, pokazane poniżej.

Command	Target	Value
open	/WebKatalogKsiazek1/faces/index.xhtml	
verifyText	id=j_idt2	Facelet Title
assertTitle		
verifyLocation	http://localhost:8080/WebKatalogKsiazek1/faces/index.xhtml	
verifyElementPresent	//a	
verifyElementPresent	link>Show All Ksiazka Items	
verifyElementPresent	link>Show All TytulKsiazki Items	
assertText	//body/a[1]	Show All Ksiazka Items
assertText	//body/a[2]	Show All TytulKsiazki Items
clickAndWait	link>Show All Ksiazka Items	

Command: verifyElementPresent
Target: //a
Value:

verifyElementPresent(locator)
Generated from isElementPresent(locator)
Arguments:
• locator - an element locator
Returns:
true if the element is present, false otherwise
Verifies that the specified element is somewhere on the page.

Formularz zawiera pola **Command**, **Target** i **Value**, które umożliwiają wprowadzanie poleceń oraz argumentów polecenia w postaci wierszy do tabeli testów. Oprócz tego, można wprowadzić komentarz do strony testowej, który służy jedynie do wstawiania informacji o przeprowadzanym teście. Poniżej pokazano sposób dodawanie komentarza do strony testowej: należy prawym klawiszem myszy kliknąć na wybrany wiersz tabeli testowej i wybrać listę pozycję **Insert New Comment**. Linię komentarza należy wprowadzać w polu **Command**.

Command	Target	Value
open	/WebKatalogKsiazek1/f...	

Testowanie zawartości uruchomionej strony

Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Insert New Command
Insert New Comment
Clear All
Toggle Breakpoint B
Set / Clear Start Point S
Execute this command X

TestSelenium_KatalogKsiazek1_2

Command	Target	Value
open	/WebKatalogKsiazek1/f...	
Testowanie zawartości uruchomionej strony		
verifyText	id=j_idt2	Facelet Title
assertTitle		
verifyLocation	http://localhost:8080/W...	
verifyElementPresent	//a	
verifyElementPresent	link>Show All Ksiazka It...	
verifyElementPresent	link>Show All TytulKsiaz...	
assertText	//body/a[1]	Show All Ksiazka Items
assertText	//body/a[2]	Show All TytulKsiazki ...
clickAndWait	link>Show All Ksiazka It...	

Log | Reference | Expert | UI-Element | Rollup | File Logging (off) | Info | Clear

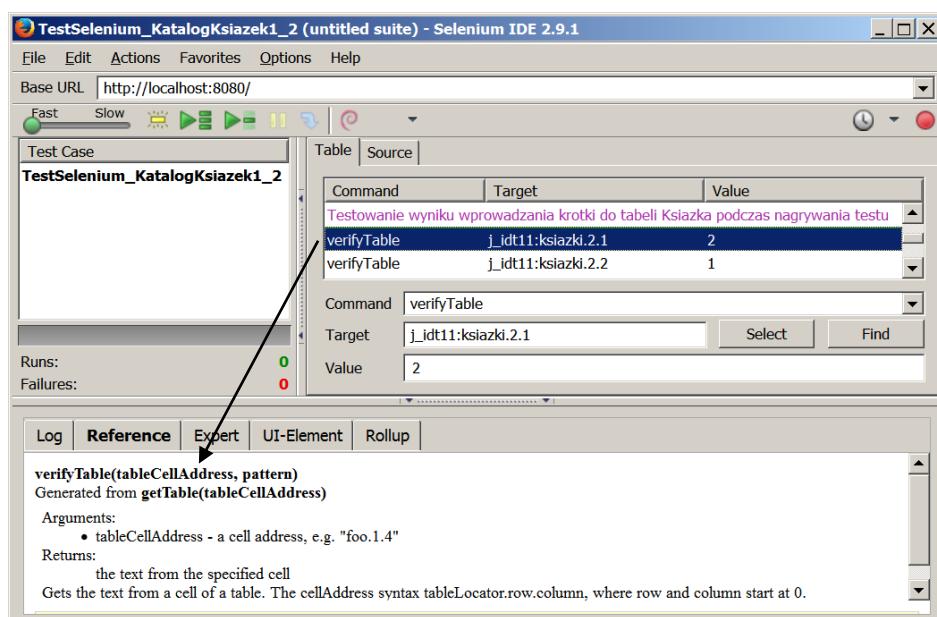
Znajdź: Podświetl Rozróżniaj wielkość liter

W Tabeli 1 przedstawiono testowanie elementów **UI** strony internetowej. Przejście na stronę **index.xhtml**, prezentującą dane książek, w której pokazano i skomentowano poszczególne polecenia **Selenium IDE** wykorzystane w ręcznie tworzonym teście.

Tabela 1

Command	Target	Value	Opis – w zakładce Reference
open	/WebKatalodKsiazek1/faces/index.xhtml		otwiera aplikację internetową i wyświetla stronę index.xhtml
Testowanie			wstawiony komentarz,
verifyText	id=j_idt2	Facelet Title	Sprawdzenie tytułu strony, gdzie atrybut id znacznika head strony index.xhtml jest równy j_idt2
assertTitle	Facelet Title		Inny sposób sprawdzenia poleciem dedykowanym
verifyLocation	http://localhost:8080/WebKatalogKsiazek1/faces/index.xhtml		Weryfikacja adresu strony głównej index.xhtml
verifyElementPresent	//a		Sprawdzenie, czy struktura strony html posiada znacznik a
verifyElementPresent	link>Show All Ksiazka Items		Weryfikacja tekstu wyświetlonego na stronie index.xhtml
verifyElementPresent	link>Show All TytułKsiążki Items		
assertText	//body/a[1]	Show All Ksiazka Items	Sprawdzenie, czy pierwszy ze znaczników a zawiera tekst umieszczany na linku: Show All Ksiazka Items
assertText	//body/a[2]	Show All TytułKsiążki Items	Sprawdzenie, czy drugi ze znaczników a zawiera tekst umieszczany na linku: Show All TytułKsiążki Items

W dolnej części formularza (rysunek poniżej) znajduje się w zakładce **Reference** informacja o znaczeniu i składni każdego z zaznaczonych polecień. Polecenia można wybierać z listy w polu **Command**. Ten rysunek przedstawia kolejny fragment testu sprawdzającego poprawność wstawionych danych podczas nagrywania testu (3-i wiersz tabeli z książkami na stronie **List.xhtml**, wybranie za pomocą linku **Show All TytułKsiążki Items** – **pierwszy wiersz o indeksie 0 jest nagłówkiem tabeli; kolumny tabeli też liczone są od 0**).



Poniżej, na rysunku przedstawiono fragment testu, wprowadzającego trzecią krotkę tabeli **Ksiazka** za pomocą formularza **Create.xhtml**, wybranym po kliknięciu na link **Create New Ksiazka**.

The screenshot shows the Selenium IDE interface with the following details:

- Test Case:** TestSelenium_KatalogKsiazek1_2
- Table:**

Command	Target	Value
clickAndWait	link=Create New Ksiazka	
type	id=j_idt12:ksiazkaId	2
type	id=j_idt12:numer	2
select	id=j_idt12:idTytul	label=entities.TytulKsiazki[tytulId=1]
clickAndWait	link=Save	
- verifyTable:** j_idt11:ksiazki.1.1, Value: 2
- Log:** verifyTable(tableCellAddress, pattern)
Generated from getTable(tableCellAddress)
- Arguments:** tableCellAddress - a cell address, e.g. "foo.1.4"
- Returns:** the text from the specified cell
Gets the text from a cell of a table. The cellAddress syntax tableLocator.row.column, where row and column start at 0.

Kolejny, ostatni fragment tworzonego ręcznie testu, testuje poprawność ostatnio wprowadzonych danych, wyświetlonych w tabelce na stronie **List.xhtml**, wybranej po naciśnięciu linku **Show All Ksiazka Items** (4-y wiersz o indeksie 3 wiersza tabelki). Następnie usuwane są dwa pierwsze wiersze tabeli za bazie danych – proces uruchomiony za pomocą elementu **Destroy** na stronie **List.xhtml**. Następnie, po przejściu na stronę **View** wykonanie testu danych na tym formularzu.

The screenshot shows the Selenium IDE interface with the following details:

- Test Case:** TestSelenium_KatalogKsiazek1_2
- Table:**

Command	Target	Value
clickAndWait	link>Show All Ksiazka Items	
verifyTable	j_idt11:ksiazki.3.1	2
verifyTable	j_idt11:ksiazki.3.2	1
clickAndWait	link=Destroy	
clickAndWait	link=Destroy	
clickAndWait	link=View	
verifyTable	id=j_idt12:ksiazka.1.1	2
verifyTable	id=j_idt12:ksiazka.2.1	1
clickAndWait	link>Show All Ksiazka Items	
- clickAndWait:** link=Destroy
- Log:** clickAndWait(locator)
Generated from click(locator)
- Arguments:** locator - an element locator
Clicks on a link, button, checkbox or radio button. If the click action causes a new page to load (like a link usually does), call waitForPageToLoad.

Kolejna część testu, przedstawiona na zrzutach z ekranu na str.27-28, jest przedstawiona szczegółowo w tabelach 2-4. Opis testu w tabeli 2 służy do sprawdzenia wartości danych książek pobranych jako trzeci wiersz tabeli o indeksie 2 z książkami ze strony [List.html](#) np. podczas nagrywania testu w p.3.1, wywołane za pomocą linku **Show All Ksiazka Items**.

Tabela 2

Command	Target	Value	Opis – w zakładce Reference
Testowanie ...			wstawiony komentarz,
clickAndWait	link=Show All Ksiazka Items		Przejście na stronę List.xhtml , prezentującą dane książek
verifyTable	j_idt11:ksiazki.2.1	2	Sprawdzenie, czy wartość w komórce tabeli o nazwie j_idt11:ksiazki.2.1, wierszu 3 (wliczając wiersz nagłówka) i kolumnie 2 jest równa 2 – numer książki
verifyTable	j_idt11:ksiazki.2.2	1	Sprawdzenie, czy wartość w komórce tabeli o nazwie j_idt11:ksiazki.2.2, wierszu 3 i kolumnie 3 jest równa 1 –id tytułu.

Kolejna część testu służy do wprowadzenia nowej książki za pomocą strony [Create.xhtml](#), wywołanej za pomocą linku **Create New Ksiazka**.

Tabela 3

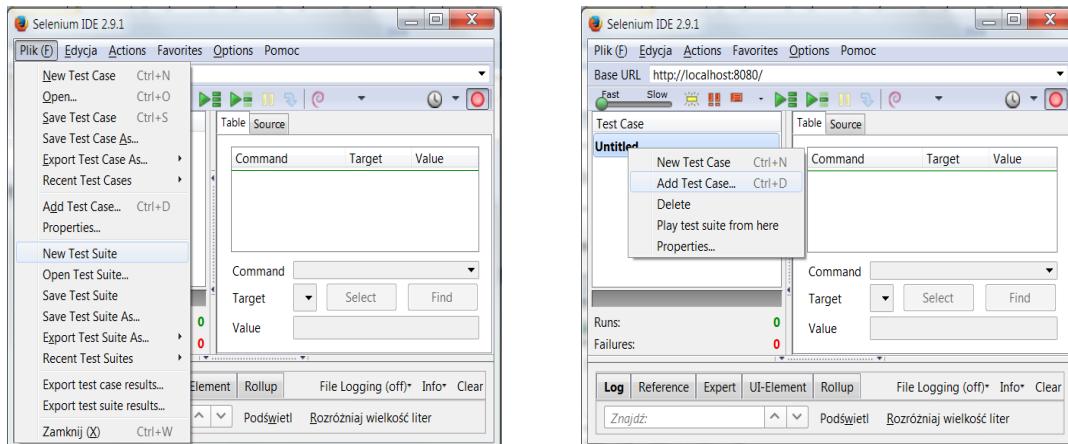
<i>Wprowadzanie nowej krotki do tabeli...</i>			<i>wstawiony komentarz,</i>
clickAndWait	link=Create New Ksiazka		Przejście na stronę Create.xhtml
type	id=j_idt12:ksiazkald	2	
type	id=j_idt12:numer	2	
select	id=j_idt12:idTytul	label=entities.TytulKsiazki[tytuId=1]	
clickAndWait	link=Save		

Kolejna część testu dotyczy testowania zachowania spójności danych przez aplikację internetową.

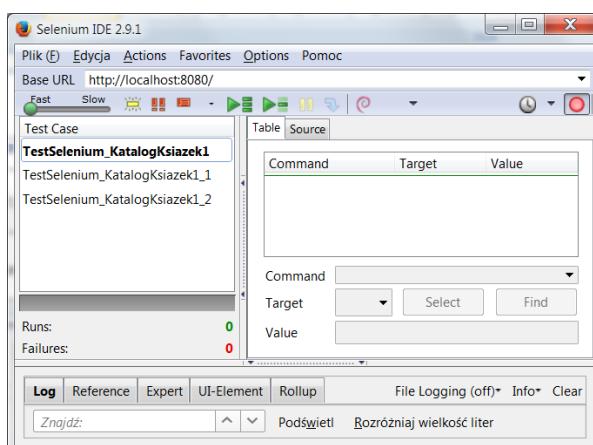
Tabela 4

Testowanie			<i>wstawiony komentarz,</i>
clickAndWait	link=Show All Ksiazka Items		Przejście na stronę List.xhtml , prezentującą dane książek
verifyTable	j_idt11:ksiazki.3.1	2	Sprawdzenie, czy wartość w komórce tabeli o nazwie j_idt11:ksiazki.3.1, wierszu 4 i kolumnie 2 jest równa 2
verifyTable	j_idt11:ksiazki.3.2	1	Sprawdzenie, czy wartość w komórce tabeli o nazwie j_idt11:ksiazki.3.2, wierszu 4 i kolumnie 3 jest równa 1
clickAndWait	link=Destroy		Usunięcie 2 wiersza w tabeli, o indeksie 1
clickAndWait	link=Destroy		Usunięcie kolejnego wiersza w tabeli-pozostał jeden wiersz ostatnio wstawiony
clickAndWait	link=View		Przejście na stronę View.xhtml
verifyTable	id=j_idt12:ksiazka.1.1	2	Testowanie danych ostatniej książki na stronie View.xhtml – drugi wiersz, druga kolumna (numer książki)
verifyTable	id=j_idt12:ksiazka.2.1	1	Testowanie danych ostatniej książki na stronie View.xhtml – trzeci wiersz, druga kolumna (Id tytułu)
clickAndWait	link=Show All Ksiazka Items		Przejście na stronę List.xhtml , prezentującą dane książki

3.3. Przed uruchomieniem obu testów należy wykonać tzw. zestaw testów (*Test Suite*) za pomocą *Plik(F)/New Test Suite*. Aby dodać test do zestawu testów: należy kliknąć na utworzony plik typu ***Test Suite*** i wybrać z listy pozycję ***Add Test Case***. W kolejnych krokach należy dodać utworzone wcześniej pliki typu ***Test Case*** w kolejności: plik z nagrany testem z p.3.1 i następnie plik z testem napisanym ręcznie z p.3.2. Na końcu należy zapisać plik z zestawem testów za pomocą ***Plik(F)/Save Test Suite***. W przykładzie zestawem testów jest plik ***TestSelenium_KatalogKsiazek1***.



Poniżej pokazano utworzony zestaw testów o nazwie ***TestSelenium_KatalogKsiazek1***.



Zestaw testów uruchamia się podobnie jak testy typu ***Test Case***: po kliknięciu na ikonę typu „zielony trójkąt” lub w ***Menu Bar: Actions/Play entire test suite***.

Poniżej pokazano uruchomienie zestawu testów w inny sposób: po kliknięciu prawym klawiszem myszy na pierwszy z testów typu **Test Case** i kliknięciu na pozycję **Play test suite from here**.

Command	Target	Value
clickAndWait	link=Show All Ksiazka Items	
verifyTable	j_idt11:ksiazki.3.1	2
verifyTable	j_idt11:ksiazki.3.2	1
clickAndWait	link=Destroy	
clickAndWait	link=Destroy	
clickAndWait	link=View	
verifyTable	id=j_idt12:ksiazka.1.1	2
verifyTable	id=j_idt12:ksiazka.2.1	1
clickAndWait	link>Show All Ksiazka Items	

Poniżej pokazano kod źródłowy zestawu testów **TestSelenium_KatalogKsiazek1.html**.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta content="text/html; charset=UTF-8" http-equiv="content-type" />
<title>Test Suite</title>
</head>
<body>
<table id="suiteTable" cellpadding="1" cellspacing="1" border="1" class="selenium">
<tbody>
<tr><td>
<b>Test Suite</b>
</td></tr>
<tr><td>
<a href="TestSelenium_KatalogKsiazek1.html">TestSelenium_KatalogKsiazek1</a>
</td></tr>
<tr><td>
<a href="TestSelenium_KatalogKsiazek1_1.html">TestSelenium_KatalogKsiazek1_1</a>
</td></tr>
<tr><td>
<a href="TestSelenium_KatalogKsiazek1_2.html">TestSelenium_KatalogKsiazek1_2</a>
</td></tr>
</tbody>
</table>
</body>
</html>
```