# Laboratorium 3. Serwlety, *Java Bean*, *JSP* i *MySQL*

## Cel zajęć

Realizacja zadań z niniejszego laboratorium pozwoli studentom poznać zasady realizacji wzorca *MVC* na przykładzie implementacji aplikacji współpracującej z bazą danych *MySQL* z wykorzystaniem podstawowego interfejsu *JDBC*.

## Zakres tematyczny

- Przygotowanie prostej aplikacji internetowej z wykorzystaniem wzorca projektowego *MVC*, współpracującej z bazą danych *MySQL* [3, 13, 24].
- Wykorzystanie:
  - serwletu (kontroler we wzorcu *MVC*),
  - stron *JSP* (widoki w *MVC*),
  - komponentu *JavaBean* (model w *MVC*).
- Wykonywanie operacji na danych w serwlecie z wykorzystaniem podstawowego interfejsu *JDBC*.

## Wprowadzenie

Wzorzec projektowy *MVC* (*ang Model View Controller*) zakłada separację kodu tworzącego i operującego na danych od kodu służącego do przedstawiania tych danych. Najprostszą realizację aplikacji internetowej w języku *Java*, spełniającej założenia *MVC* można utworzyć z wykorzystaniem *Java Bean* jako modeli, serwletu jako kontrolera oraz widoków *JSP*. Podstawowe narzędzia implementacji takiej separacji są standardowo dostępne w *API* serwletów (interfejs ***RequestDispatcher***).

Implementacja wzorca *MVC* z zastosowaniem interfejsu ***RequestDispatcher*** składa się z następujących kroków:

1. Zdefiniowanie komponentów reprezentujących dane jako ***JavaBean***.
2. Zastosowanie serwletów do obsługi żądań.
3. Zapisanie danych w komponentach (w serwlecie).
4. Zapisanie (w serwlecie) komponentów w obiekcie żądania, sesji lub kontekście serwletu (metodą ***setAttribute***).
5. Przekazanie żądania z serwletu do strony *JSP* (metoda ***forward*** obiektu ***requestDispatcher***).
6. Pobranie danych (utworzonych w serwlecie) na stronie *JSP* za pomocą znaczników ***jsp:useBean*** i ***jsp:getProperty***.

W *MVC* serwlety odpowiadają za nadsyłane żądania, w ogóle nie generują wyników. Zadanie to realizują strony widoków *JSP*, które z kolei nie tworzą i nie modyfikują komponentów danych a jedynie odwołują się do nich.

## Zadanie 3.1. Przygotowanie bazy danych na serwerze *MySQL*

Korzystając z pliku ***world.sql*** z przykładową bazą danych (do pobrania ze strony: https://dev.mysql.com/doc/index-other.html), utwórz na serwerze *MySQL* bazę danych ***world*** i przejrzyj jej strukturę.

Utwórz nowy projekt aplikacji webowej o nazwie ***pai_lab3***, a następnie do pliku ***pom.xml*** dodaj następującą zależność, która dołączy do projektu sterownik do *MySQL*:

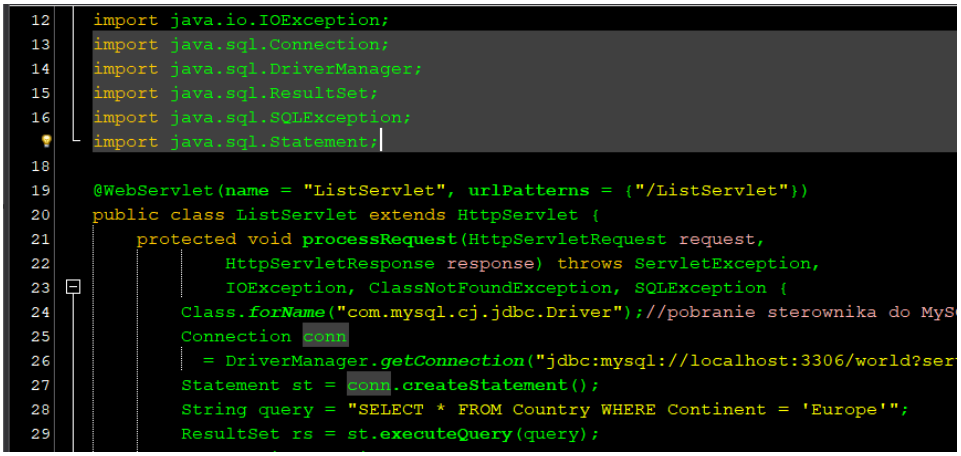
```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
     <groupId>mysql</groupId>
     <artifactId>mysql-connector-java</artifactId>
     <version>8.0.11</version>
</dependency>
```

## Zadanie 3.2. Interfejs *JDBC* i praca z bazą danych w serwlecie

W pakiecie projektu utwórz serwlet ***ListServlet***, który będzie łączył się z bazą danych. W serwlecie pobierz i wyświetl na stronie *HTML* informacje np. o krajach Europy:

```
//pobranie sterownika do MySQL:
Class.forName("com.mysql.cj.jdbc.Driver");
//utworzenie obiektu połączenia do bazy danych MySQL:
Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/world?
serverTimezone=UTC", "root", "");
//utworzenie obiektu do wykonywania zapytań do bd:
Statement st = conn.createStatement();
String query="SELECT * FROM Country WHERE Continent = 'Europe'";
//wykonanie zapytania SQL:
ResultSet rs = st.executeQuery(query);
```

Wprowadź odpowiednią nazwę bazy danych, użytkownika oraz hasło. Metoda pracująca z bazą danych powinna wyrzucać (lub obsługiwać) wyjątki ***ClassNotFoundException*** i ***SQLException***. Jeśli korzystasz z pomocniczej metody serwletu ***processRequest()***, jak na rysunku 3.1, to także metody ***doGet()*** i ***doPost()***, które ją wywołują, powinny obsługiwać te wyjątki. Zwróć też uwagę, że wszystkie klasy do pracy z bazą danych (***Connection, Statement, ResultSet***) są importowane z pakietu ***java.sql***.




**Rys. 3.1. Utworzenie połączenia do bazy danych w serwlecie *ListServlet***

Korzystając z metod klasy ***ResultSet*** sprawdź możliwości (Rys. 3.2) konwersji danych pobieranych z odpowiednich kolumn tabeli ***Country***. Wyświetl informacje o nazwie, kodzie oraz liczbie ludności w krajach Europy (Rys. 3.3):

```
while (rs.next()) {
   //pobierz i wyświetl dane z odpowiedniej kolumny
   out.print(rs.getString("name"));
   //out.println …
}
```

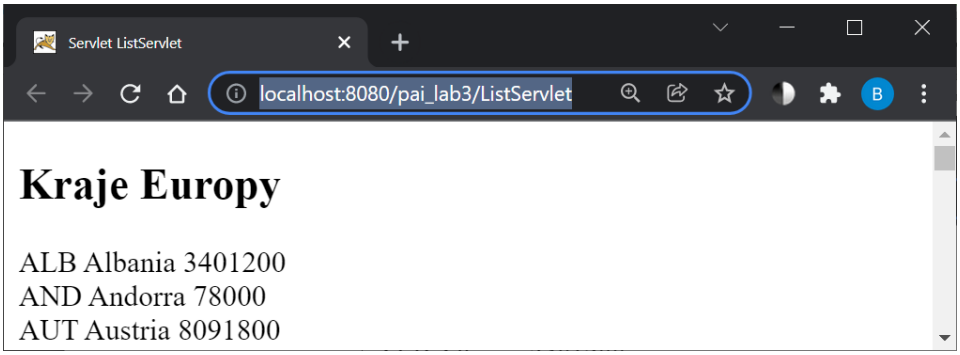
## Zadanie 3.3. Prosty *MVC* – serwlet, model *JavaBean*, widok *JSP*

W poprzednim zadaniu dostęp do danych i generowanie widoku realizowano za pomocą pojedynczej klasy serwletu. Teraz zmodyfikuj poprzedni projekt tak, aby rozdzielić funkcje na kilka klas. Serwlet (klasa kontrolera w *MVC*) nadal obsługuje żądanie i pobiera dane z bazy (jak poprzednio), ale przekazuje je w obiekcie sesji za pomocą klasy modelu (***JavaBean***) do strony widoku *JSP*.

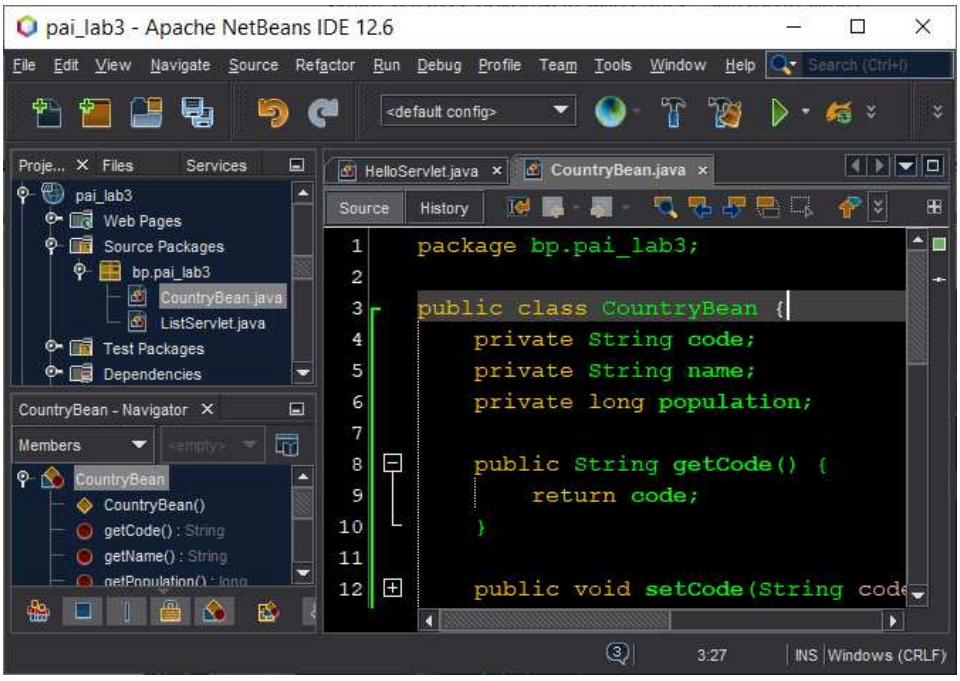
W pakiecie projektu utwórz klasę modelu ***CountryBean*** (spełniającą założenia klasy ***JavaBean***), która pomoże przetwarzać dane pomiędzy serwletem a stronami widoków *JSP*. Klasa powinna implementować interfejs ***Serializable*** oraz zawierać pola prywatne odpowiadające kolumnom tabeli ***Country*** z bazy danych ***world*** (zdefiniuj na razie tylko 3 pola klasy: ***code, name, population***). Korzystając ze wsparcia *IDE*, wygeneruj automatycznie kod metod ***get*** oraz ***set*** w klasie ***CountryBean*** (Rys. 3.4).

| | TINYINT | SMALLINT | INTEGER | BIGINT | REAL | FLOAT | DOUBLE | DECIMAL | NUMERIC | BIT | CHAR | VARCHAR | LONGVARCHAR | BINARY | VARBINARY | LONGVARBINARY | DATE | TIME | TIMESTAMP | CLOB | BLOB | ARRAY | REF | STRUCT | JAVA_OBJECT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| getByte | X | x | x | x | x | x | x | x | x | x | x | x | x | | | | | | | | | | | | |
| getShort | x | X | x | x | x | x | x | x | x | x | x | x | x | | | | | | | | | | | | |
| getInt | x | x | X | x | x | x | x | x | x | x | x | x | x | | | | | | | | | | | | |
| getLong | x | x | x | X | x | x | x | x | x | x | x | x | x | | | | | | | | | | | | |
| getFloat | x | x | x | x | X | x | x | x | x | x | x | x | x | | | | | | | | | | | | |
| getDouble | x | x | x | x | x | X | X | x | x | x | x | x | x | | | | | | | | | | | | |
| getBigDecimal | x | x | x | x | x | x | x | X | X | x | x | x | x | | | | | | | | | | | | |
| getBoolean | x | x | x | x | x | x | x | x | x | X | x | x | x | | | | | | | | | | | | |
| getString | x | x | x | x | x | x | x | x | x | x | X | X | x | x | x | x | x | x | x | | | | | | |
| getBytes | | | | | | | | | | | | | | X | X | x | | | | | | | | | |
| getDate | | | | | | | | | | | x | x | x | | | | X | | x | | | | | | |
| getTime | | | | | | | | | | | x | x | x | | | | | X | x | | | | | | |
| getTimestamp | | | | | | | | | | | x | x | x | | | | x | x | X | | | | | | |
| getAsciiStream | | | | | | | | | | | x | x | X | x | x | x | | | | | | | | | |
| getUnicodeStream | | | | | | | | | | | x | x | X | x | x | x | | | | | | | | | |
| getBinaryStream | | | | | | | | | | | | | | x | x | X | | | | | | | | | |
| getClob | | | | | | | | | | | | | | | | | | | | X | | | | | |
| getBlob | | | | | | | | | | | | | | | | | | | | | X | | | | |
| getArray | | | | | | | | | | | | | | | | | | | | | | X | | | |
| getRef | | | | | | | | | | | | | | | | | | | | | | | X | | |
| getCharacterStream | | | | | | | | | | | x | x | X | x | x | x | | | | | | | | | |
| getObject | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | X | X |

**Rys. 3.2. Metody do konwersji typów *SQL* na typy języka *Java* (*ResultSet.getXXX*) [12]**
http://info.ee.pw.edu.pl/Java/1.4.2/docs/guide/jdbc/getstart/mapping.html




**Rys. 3.3. Wynik działania serwletu *ListServlet***




**Rys. 3.4. Definicja klasy modelu *CountryBean***

Korzystając z klasy ***CountryBean*** oraz mechanizmu sesji, zmodyfikuj serwlet ***ListServlet*** tak, aby przekazywał dane o krajach (w obiekcie sesji) do strony widoku ***countryList.jsp***. W sesji, pod kluczem "***list"*** zapisz listę ***ArrayList*** zawierającą obiekty typu ***CountryBean***. W serwlecie pozostaw **tylko** instrukcje konieczne do pracy z bazą danych i obiektem sesji. Samym wyświetlaniem listy krajów z obiektu sesji zajmie się już strona widoku *JSP*.

W serwlecie **nie twórz** już i nie korzystaj z obiektu ***out***, nie potrzebny jest też cały blok ***try-catch***, który w zadaniu 3.2 wykorzystywano do tworzenia treści odpowiedzi):

```
HttpSession session=request.getSession(true);
CountryBean country;
ArrayList<CountryBean> list=new ArrayList<CountryBean>();
  while (rs.next()) {
     country = new CountryBean();
     //pobierz dane z odpowiedniej kolumny
     //przypisz je do właściwości obiektu CountryBean
     country.setName(rs.getString("name"));
     // …
     list.add(country);
  }
session.setAttribute("list", list);
```

Następnie w serwlecie dodaj przekierowanie do strony *JSP*:

```
response.sendRedirect("countryList.jsp");
```

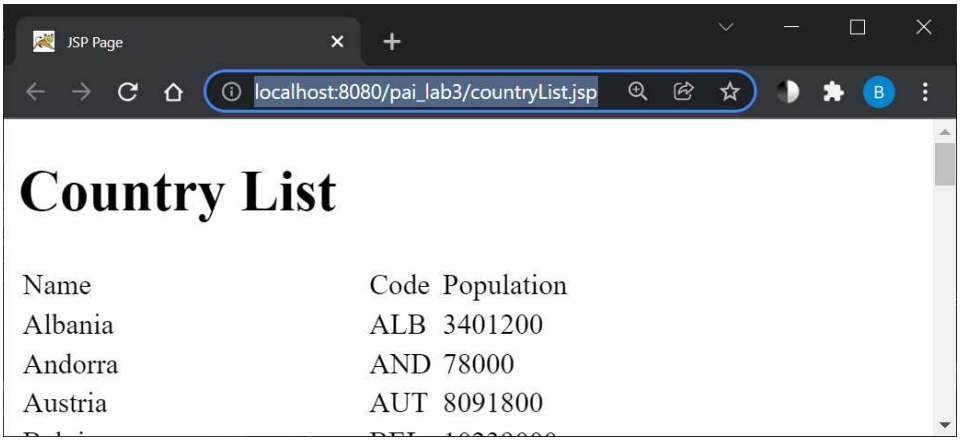
Do projektu dodaj nową stronę widoku ***countryList.jsp*** oraz wyświetl tam dane pobrane z obiektu sesji (Rys. 3.5):

```
<% ArrayList<CountryBean> list =
 (ArrayList<CountryBean>)session.getAttribute("list");
%>
```

Do wyświetlenia danych na stronie *JSP* skorzystaj z pętli:

```
for(CountryBean country:list){}
```

Uruchom ponownie ***ListServlet*** (Rys. 3.5) i zwróć uwagę na adres *URL*, który jest widoczny w przeglądarce po jego uruchomieniu (porównaj z adresem z rysunku 3.3).

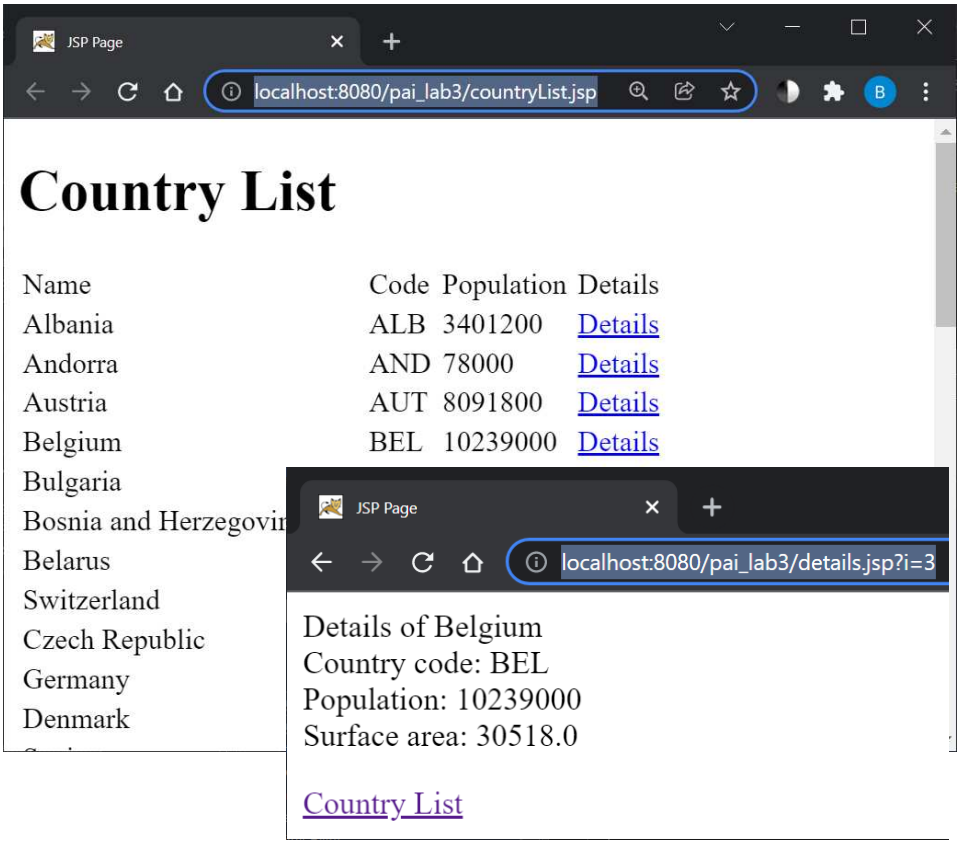



**Rys. 3.5. Widok wygenerowany przez stronę *countryList.jsp* po uruchomieniu serletu *ListServlet***

W zmodyfikowanej aplikacji, serwlet zajmuje się logiką operacji na danych a strona *JSP* wyświetla przygotowane przez serwlet dane, korzystając z modelu ***CountryBean***.

## Zadanie 3.4. Identyfikator obiektu w adresie *URL*

Zmodyfikuj stronę *JSP* tak, aby obok każdego kraju wyświetlane było hiperłącze prowadzące do szczegółowych informacji o wybranym państwie (Rys. 3.6). W hiperłączu wskaż plik, do którego ma nastąpić przekierowanie, łącznie z przekazaniem ***indeksu*** wybranego państwa z listy (***list.indexOf(country)***) jako parametru do nowej strony ***details.jsp***. Na stronie ***details.jsp*** wyświetl pełniejsze informacje o wybranym kraju (***list.get(indeks)***). Do strony ***details.jsp*** dodaj także link prowadzący z powrotem do listy krajów (Rys. 3.6).




**Rys. 3.6. Lista krajów europejskich z tabeli *Country* wyświetlona przez stronę *countryList.jsp***