

Marcin Czernek

Nr albumu 39924

Semestr 3 informatyka niestacjonarne

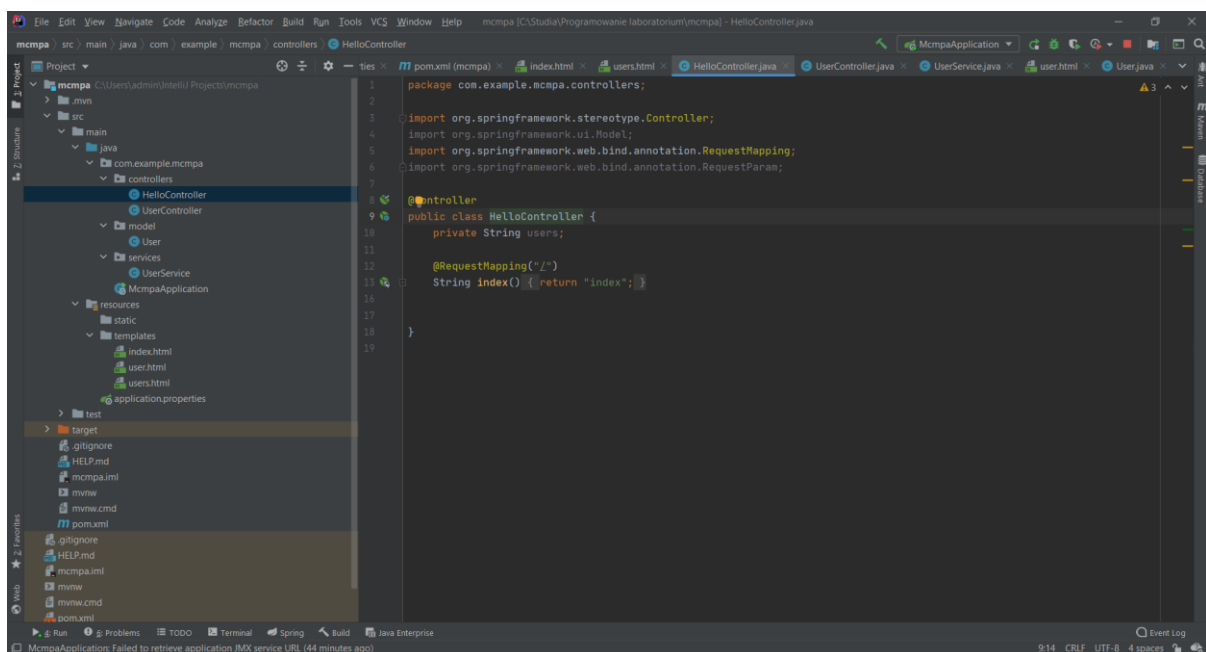
Sprawozdanie z Laboratorium 4 – Java Spring MPA

Jest to aplikacja MPA czyli multiple page application. Taka aplikacja działa na rozwiązaniu generowania treści strony HTML przez serwer i wysyłaniu do klienta. Aplikacja wyświetla w przeglądarce użytkowników ma także dobudowane metody dodawania nowych (add) użytkowników i usuwania ich (delete). Składa się z jednej głównej klasy w modelu "User", serwisu "UserService", dwóch kontrolerów "HelloController" i "UserController" a także widoki index.html, user.html i users.html

Pierwszą najważniejszą rzeczą po wygenerowaniu projektu spring initializer jest dodanie biblioteki springa w pom.xml

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
</dependencies>
```

Następnie tworze kontroler z funkcjami, które są endpointami. Pierwszy kontroler to "HelloController". Tu ważne jest anotacja @Controller która informuje Springa że ta klasa jest kontrolerem. Ten kontroler składa się z funkcji, które będą wywoływane, gdy przyjdzie zapytanie pod któryś z adresów. Funkcja mapuje na adres endpointa "/" Hello będzie zwracał tylko ten indeks.



```
package com.example.mcpa.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

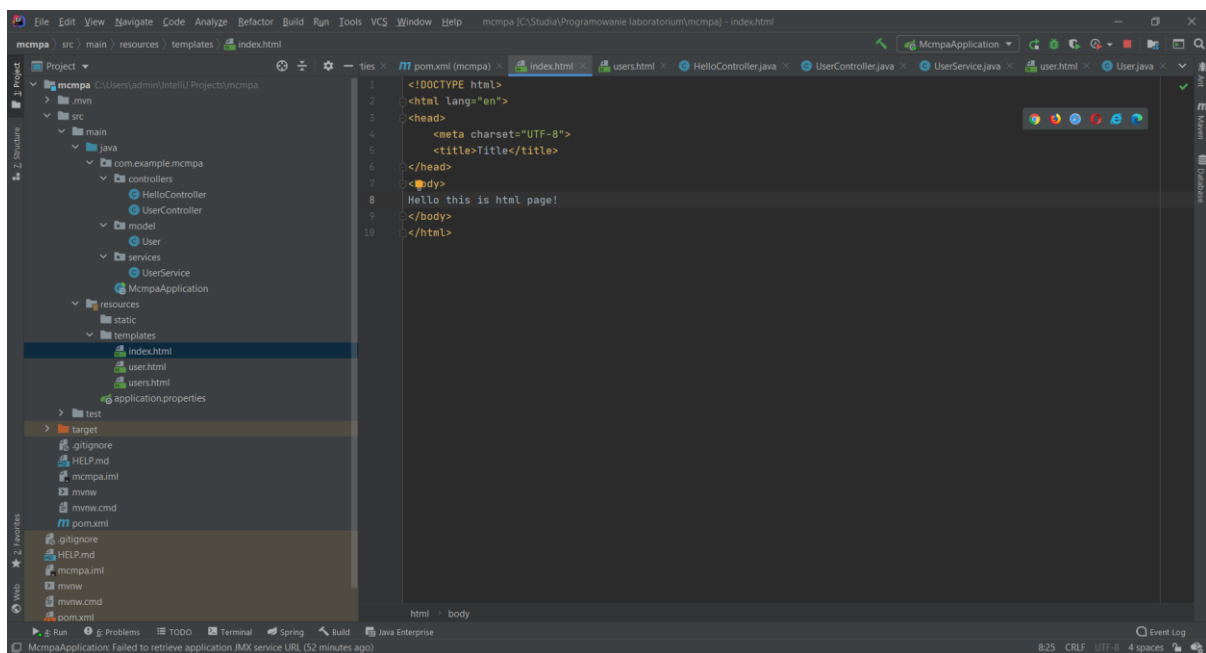
@Controller
public class HelloController {
    private String users;

    @RequestMapping("/")
    String index() { return "index"; }
}
```

Dodanie biblioteki thymeleaf da mi możliwość dodawanie plików html . Po każdym dodaniu biblioteki pamiętam o przeładowaniu Mavena.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

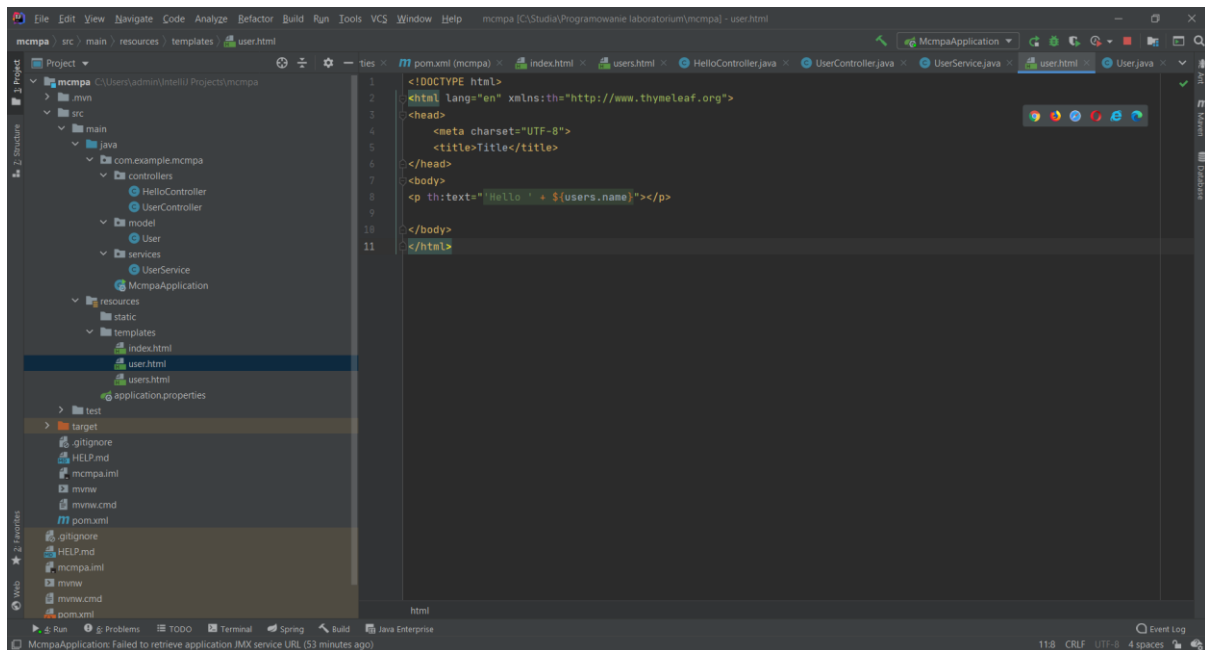
Tworzę więc w swoim projekcie plik html o nazwie index.html strona ta będzie zwracana w HelloController



Po wprowadzeniu do przeglądarki linka localhost:8080 otrzymuję żądaną stronę

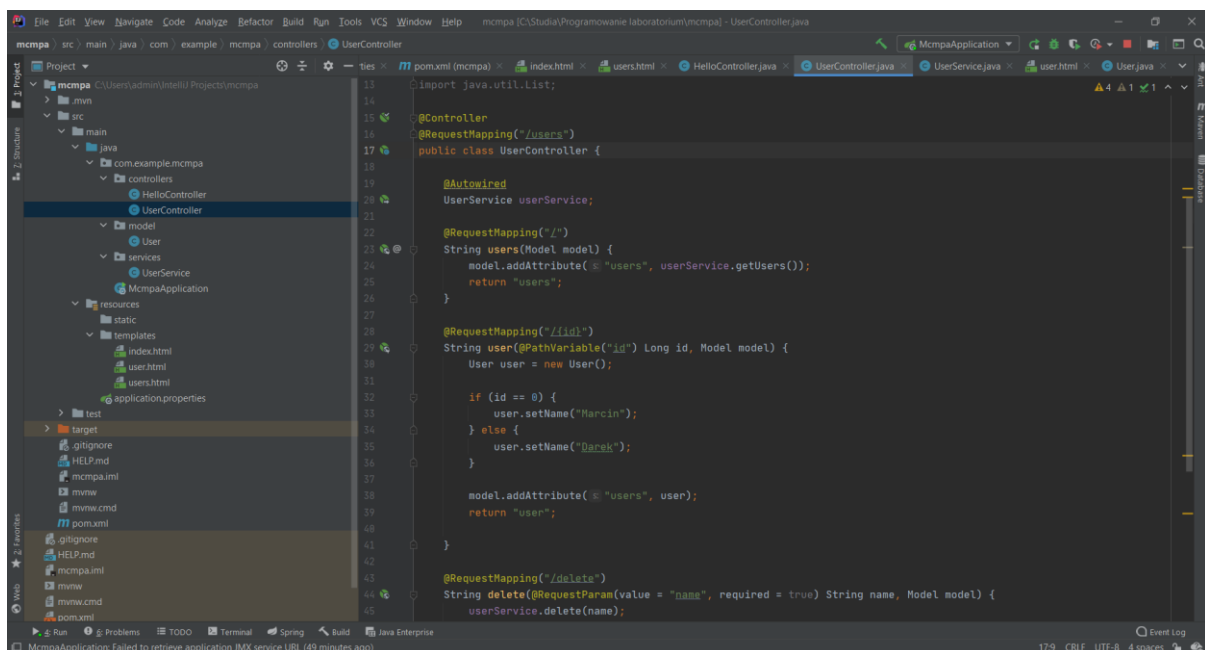


W pliku users.html dodam informacje, że ma być parsowana przez bibliotekę thymeleaf którą dodałem do bibliotek. Gdy przyjdzie zapytanie pod user to będzie wyświetlała się ta lista użytkowników.



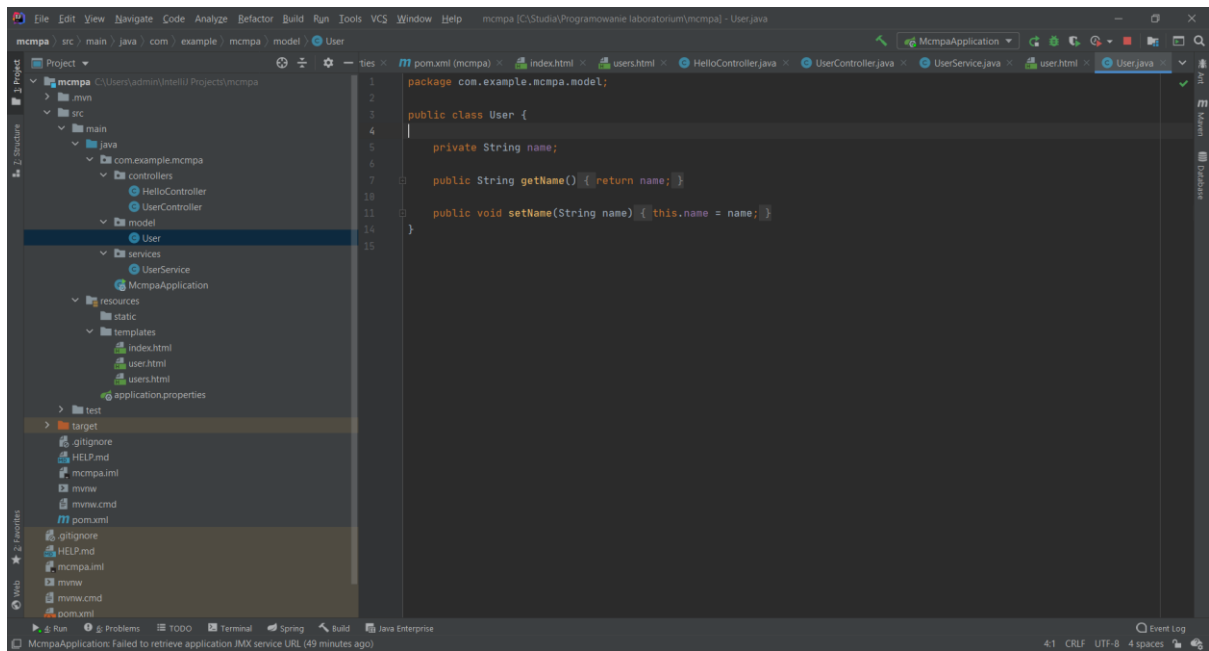
```
1 <!DOCTYPE html>
2 <html lang="en" xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <meta charset="UTF-8">
5 <title>Title</title>
6 </head>
7 <body>
8 <p th:text="Hello ' + ${users.name}"></p>
9
10 </body>
11 </html>
```

Utworzony UserController będzie miał funkcje dodawania, zwracania i kasowania większej ilości obiektów.



```
13 import java.util.List;
14
15 @Controller
16 @RequestMapping("/users")
17 public class UserController {
18
19     @Autowired
20     UserService userService;
21
22     @RequestMapping("/")
23     String users(Model model) {
24         model.addAttribute("users", userService.getUsers());
25         return "users";
26     }
27
28     @RequestMapping("/{id}")
29     String user(@PathVariable("id") Long id, Model model) {
30         User user = new User();
31
32         if (id == 0) {
33             user.setName("Marcin");
34         } else {
35             user.setName("Darek");
36         }
37
38         model.addAttribute("users", user);
39         return "user";
40     }
41
42     @RequestMapping("/delete")
43     String delete(@RequestParam(value = "name", required = true) String name, Model model) {
44         userService.delete(name);
45     }
46 }
```

Klasa User zawiera proste pola name, getName, setName



Strona user.html będzie wyświetlała nazwę użytkownika z UserControllerera

