CPAN-252 WEB APPLICATION DEVELOPMENT FOR CPAN DIPLOMA. PROF. ANTON KOVUNOV

BUILDING FIRST SIMPLE APP USING SPRING WEB AND THYMELEAF

SPRING WEB APPLICATION

- We will start with a small addition to the Tekken Reborn application by adding a simple home controller
- It will be annotated with @Controller annotation, annotation similar to @Component but with additional “sauce”

```java
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {

    @GetMapping("/")
    public String index() {
        return "home";
    }
}
```

-

CONTROLLER ANNOTATION

- Essentially to make Spring class a Spring Bean, you need to annotate it with @Component annotation, @Controller is just a meta annotation that includes @Component in itself and provides more semantic clarity on the role of the class.
- So you can use @Component, but there are more descriptive options like @Service, @Repository, @Controller that serve same purpose

CONTROLLER ANNOTATION

```
 */
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Component
public @interface Controller {

    /**
     * The value may indicate a suggestion for a logical component name,
     * to be turned into a Spring bean in case of an autodetected component.
     * @return the suggested component name, if any (or empty String otherwise)
     */
    @AliasFor(annotation = Component.class)
    String value() default "";

}
```

HOME CONTROLLER

- So we defined @GetMapping() index method that returns String “home”.
- Let’s decompose it:
- So when we run our Application and receive get request on the root path, it will search for a view called ‘home.html’ and will display it on the page. Let’s create the view using Thymeleaf

WHAT IS THYMELEAF

- Thymeleaf is a template engine that can provide integrated view experience with html core and enriched features like using custom paths, calling Java class methods and providing validations
- It’s similar to JSP which was a standard in Java EE apps, but it’s overall more versatile and just more to my liking :)
- So let’s define our first template

TEKKEN TEMPLATE

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Home</title>
</head>
<body>
    <h1>Welcome to the Tekken Reborn application</h1>
    <img th:src="@{/images/TekkenReborn.jpeg}">
</body>
</html>
```

CONTROLLER TEST AND SPRING ACTIONS IN SIMPLE TERMS

- When we start our Spring Boot application and open localhost:8080, GET request is executed towards / path.
- Then Spring checks for controller with / GET Mapping available
- It finds our HomeController and sees that we return “home” string
- It understands that it needs to find that template in `templates` folder
- It finds home page and displays it

LET'S WRITE A TEST

```java
package com.cpan252.tekkenreborn.controller;

import static org.hamcrest.Matchers.containsString;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.view;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.test.web.servlet.MockMvc;

@WebMvcTest
public class HomeControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Test
    public void testHomePage() throws Exception {
        mockMvc.perform(get(urlTemplate: "/"))
                .andExpect(content().string(containsString("Home")))
                .andExpect(status().isOk())
                .andExpect(view().name(expectedViewName: "home"));
    }

}
```

QUESTIONS?

On the lab we going to add some logic to our app and dive deeper into templating