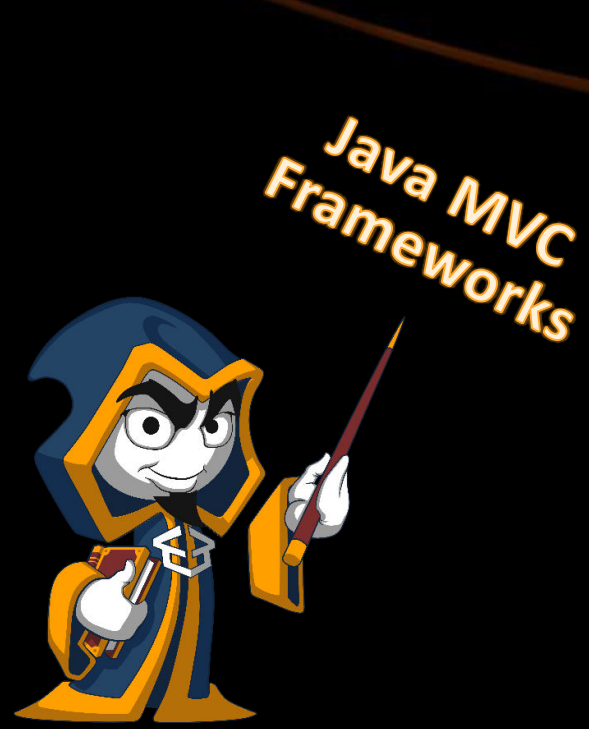
Java MVC Frameworks

Spring Essentials.
Thymeleaf & Controllers.

SoftUni Foundation



SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Contents

Questions

SoftUni Foundation

sli.do

#java-web

SoftUni Foundation

Thymeleaf

The Templating Engine

What is Thymeleaf?

- Thymeleaf is a view engine used in Spring
- It allows us to:
  - Use variables in our views
  - Execute operations on our variables
  - Iterate over collections
  - Make our views dynamical

How to use Thymeleaf?

SoftUni Foundation

- Use Spring Initializr to import Thymeleaf, or use this dependency in your **pom.xml**:

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

- Define the Thymeleaf library in your html file:

```
<html xmlns="http://www.w3.org/1999/xhtml"
                      xmlns:th="http://www.thymeleaf.org">
```

Change Thymeleaf Version

SoftUni Foundation

- You also need to change the Thymeleaf version in your `pom.xml`:

```
<properties>
  <thymeleaf.version>
    3.0.9.RELEASE
  </thymeleaf.version>
  <thymeleaf-layout-dialect.version>
    2.2.0
  </thymeleaf-layout-dialect.version>
</properties>
```

Thymeleaf Tags and Attributes

SoftUni Foundation

- All Thymeleaf tags and attributes begin with **th:** by default
- Example of Thymeleaf attribute:

```
<p th:text="Example"></p>
```

- Example of Thymeleaf tag(element processor):

```
<th:block>
  ...
</th:block>
```

- **th:block** is an attribute container that disappears in the HTML

Thymeleaf Standard Expressions

SoftUni Foundation

- Variable Expressions

```
${...}
```

- Selection Expressions

```
*{...}
```

- Link (URL) Expressions

```
@{...}
```

- Fragment Expressions

```
~{...}
```

Thymeleaf Variable Expressions

SoftUni Foundation

- Variable Expressions are executed on the context variables

```
${...}
```

- Examples:

```
${session.user.name}
```

```
${title}
```

```
${game.id}
```

Thymeleaf Selection Expressions

SoftUni Foundation

- Selection Expressions are executed on the previously selected object

```
*{...}
```

- Example:

```
<div th:object="${book}">
  ...
  <span th:text="*{title}">...</span>
  //equivalent to ${book.title}
  ...
</div>
```

Thymeleaf Link Expressions

SoftUni Foundation

- Link Expressions are used to build URLs

```
@{...}
```

- Example:

```
<a th:href="@{/register}">Register</a>
```

- You can also pass query string parameters:

```
<a th:href="@{/details(id=${game.id})}">Details</a>
```

- Create dynamic URLs

```
<a th:href="@{/games/{id}/edit(id=${game.id})}">Edit</a>
```

Variables in Thymeleaf

SoftUni Foundation

- Variables in Thymeleaf are defined using the following attribute:

```
th:with="var = 'pesho'"
```

- Where 'var' is the name of our variable and 'pesho' it's the value
- Variables exists only in the scope of the tag

```
<div th:with="name = ${user.name}">
  <p th:text="${name}"> // valid
</div>

<p th:text="${name}"> // invalid
```

Thymeleaf Fragment Expressions

SoftUni Foundation

- Fragment Expressions are easy way to move around templates

```
~{...}
```

- Example:

```
<div th:with="frag=~{footer :: #main}">
  …
  <p th:insert="${frag}">
  …
</div>
```

Thymeleaf Fragments

SoftUni Foundation

- Including entire file

```
th:insert="~{fragments/header}"
```

- Including only a fragment

```
th:insert="~{fragments/header :: dark}"
```

- Including fragment with parameter

```
th:insert="~{fragments/header (theme='dark')}"
```

- You can also use **`th:replace`** in order to replace the entire element

Forms in Thymeleaf

SoftUni Foundation

- In Thymeleaf you can create almost normal HTML forms:

```
<form th:action="@{/user}" th:method="post">
    <input type="number" name="id"/>
    <input type="text" name="name"/>
    <input type="submit"/>
</form>
```

- You can have a controller that will accept an object of given type:

```
@PostMapping("/user")
public ModelAndView register(@ModelAttribute User user)
{ ... }
```

Forms in Thymeleaf (2)

SoftUni Foundation

- You can pass objects to forms in order to use validations:

```
<form th:action="@{/user}" th:method="post"
                                th:object=${user}>
    <input type="number" th:field="*{id}"/>
    <input type="text" th:field="*{name}"/>
    <input type="submit"/>
</form>
```

- The `th:field` attribute creates different attributes based on the input type

Conditional Statements in Thymeleaf

SoftUni Foundation

- If statements can be created in the following way:

```
<div th:if="${…}">
  <p>The statement is true</p>
</div>
```

- You can create inverted if statements using **th:unless**:

```
<div th:unless="${…}">
  <p>The statement is false</p>
</div>
```

Conditional Statements in Thymeleaf (2)

SoftUni Foundation

- You can also create switch in which the default case is "*":

```
<div th:switch="${writer.role}">
  <p th:case="'ADMIN'">The user is admin</p>
  <p th:case="'MOD'">The user is moderator</p>
  <p th:case="*">The user has no privileges</p>
</div>
```

- Or use ternary operator:

```
<p th:text="${row.even} ? 'even' : 'odd'">
  ...
</p>
```

Loops in Thymeleaf

SoftUni Foundation

- Creating a for loop:

```
<div th:each="element :
                ${#numbers.sequence(start, end, step)}">
    <p th:text="${element}"></p>
</div>
```

- Example:

```
<div th:each="element : ${#numbers.sequence(1, 5)}">
    <p th:text="${element}"></p>
</div>
// 1 2 3 4 5
```

Loops in Thymeleaf (2)

SoftUni Foundation

- Creating a for-each loop:

```
<div th:each="book : ${books}" th:object="${book}">
  <p th:text="*{author}"></p>
</div>
```

- Getting the iterator:

```
<div th:each="u, iter : ${users}" th:object="${u}">
  <p th:text="|ID: *{id}, Username: *{name}|"></p>
  <p th:text="|${iter.index} of ${iter.size}|"></p>
</div>
```

SoftUni Foundation

Spring Controllers

Annotations, IoC Container

Spring Controllers

SoftUni Foundation

- Defined with the `@Controller` annotation:

```
@Controller
public class HomeController {
…
}
```

- Controllers can contain multiple actions on different routes.

Controller Actions

SoftUni Foundation

- Annotated with with `@RequestMapping(…)`

```
@RequestMapping("/home")
public String home() {
    return "home-view";
}
```

- Or

```
@RequestMapping("/home")
public ModelAndView home(ModelAndView mav) {
    mav.setViewName("home-view");
    return mav;
}
```

Request Mapping

SoftUni Foundation

- Problem when using `@RequestMapping` is that it accepts all types of request methods (get, post, put, delete, head, patch…)

- Execute only on GET requests :

```
@RequestMapping(value="/home", method=RequestMethod.GET)
public String home() {
    return "home-view";
}
```

Get Mapping

SoftUni Foundation

- Easier way to create route for a GET request:

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

- This is alias for **RequestMapping** with method GET

Post Mapping

SoftUni Foundation

- Similar to the **GetMapping** there is also an alias for **RequestMapping** with method POST:

```
@PostMapping("/register")
public String register() {
    …
}
```

- Similar annotations exist for all other types of request methods

Controller Routing

SoftUni Foundation

- You can set all actions in a controller to start with a given routing:

```
@Controller
@RequestMapping("/admin")
public class AdminController {
  @GetMapping("/")
  public String adminPanel() {
      …
  }
}
```

- Calling the **adminPanel()** will be done on route "**/admin/**"

Passing Attributes to View

SoftUni Foundation

- Passing a string to the view:

```
@GetMapping("/")
public String welcome(Model model) {
  model.addAttribute("name", "Pesho");

  return "welcome";
}
```

- The `Model` object will be automatically passed to the view as context variables and the attributes can be accessed from Thymeleaf using the Expression syntax - `${name}`

Working with the Session

SoftUni Foundation

- The session will be injected from the IoC container when called:

```
@GetMapping("/")
public String home(HttpSession httpSession) {
   …
   httpSession.setAttribute("id", 2);
   …
}
```

- Later the session attributes can be accessed from Thymeleaf using the expression syntax and the **#session** object:

```
${session.id}
```

Setting a Cookie

SoftUni Foundation

- In order to set a cookie, we need to access the **HttpResponse**:

```
@GetMapping("/darktheme")
public String darkTheme(HttpServletResponse httpResponse)
{
   …
   httpResponse.addCookie("theme", "dark");
   …
}
```

Getting a Cookie

SoftUni Foundation

- Accessing a cookie, you've set previously:

```
public String index(@CookieValue("theme") String theme) {
   …
}
```

- Setting a default value

```
public String index(@CookieValue(value = "theme",
defaultValue = "bootstrap.min.css") String theme ) {
   …
}
```

Http Request

SoftUni Foundation

- Accessing the `HttpRequest`:

```
public String index(HttpServletRequest httpRequest) {
   …
}
```

- Accessing all cookies:

```
public String index(HttpServletRequest httpRequest) {
   Cookie[] cookies = httpRequest.getCookies();
   …
}
```

Request Parameters

SoftUni Foundation

- Getting a parameter from the query string:

```
@GetMapping("/details")
public String details(@RequestParam("id") Long id) {
   …
}
```

- `@RequestParam` can also be used to get POST parameters

```
@PostMapping("/register")
public String register(@RequestParam("name") String name)
{ … }
```

Request Parameters with Default Value

SoftUni Foundation

- Getting a parameter from the query string:

```
@GetMapping("/comment")
public String comment(@RequestParam(name="author",
defaultValue = "Annonymous") String author)
{ … }
```

- Making parameter optional:

```
@GetMapping("/search")
public String search(@RequestParam(name="sort", required
= false) String sort)
{ … }
```

Form Objects

SoftUni Foundation

- Spring will automatically try to fill objects with a form data

```
@PostMapping("/register")
public String register(@ModelAttribute UserDTO userDto) {
   …
}
```

- The input field names must be the same as the object field names

Redirecting

- Redirecting after POST request:

```
@PostMapping("/register")
public String register(@ModelAttribute UserDTO userDto) {
  …
  return "redirect:/login";
}
```

Redirecting with Parameters

SoftUni Foundation

- Redirecting with query string parameters

```
@PostMapping("/register")
public String register(@ModelAttribute UserDTO userDto,
RedirectAttributes redirectAttributes) {

  redirectAttributes.addAttribute("errorId", 3);
  return "redirect:/login";
}
```

Redirecting with Attributes

SoftUni Foundation

- Keeping objects after redirect

```
@PostMapping("/register")
public String register(@ModelAttribute UserDTO userDto,
RedirectAttributes redirectAttributes) {
  …
  redirectAttributes.addFlashAttribute("userDto",
userDto);

  return "redirect:/register";
}
```

Summary

- Thymeleaf is a powerful view engine
  - You can work with variables and helper objects
  - You can create loops and conditional statements
  - You can easily create forms
- The Spring controllers have built-in IoC container
  - You can create routings on actions and controllers
  - You have access to the HttpRequest, HttpResponse, HttpSession, Cookies and others
  - You can redirect between actions

Java MVC Frameworks – Spring Essentials

SoftUni Foundation

Questions?

https://softuni.bg/courses/java-mvc-frameworks-spring

License

SoftUni Foundation

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

Software University

SoftUni Foundation

FORUM