Семинар №4
Spring MVC. Использование шаблонизатора Thymeleaf.

1. **Инструментарий:**

Урок
Презентация

---

2. **Цели семинара №4:**
   ● Понять принцип работы Spring MVC и Thymeleaf.
   ● Закрепить навыки работы с аннотациями @Controller и @RestController

   По итогам семинара №4 слушатель должен **знать**:
   ● Что такое Spring MVC и Thymeleaf.
   ● Как работают аннотации @Controller и @RestController.
   ● Как использовать Thymeleaf для генерации динамических HTML-страниц

   По итогам семинара №4 слушатель должен **уметь**:
   ● Работать с Spring MVC и Thymeleaf.
   ● Создавать веб-страницы с помощью @Controller и Thymeleaf.
   ● Правильно применять аннотации @Controller и @RestController.

---

3. План Содержание:

Этап урока	Тайминг, минуты	Формат
Введение, обзор темы	20	Модерирует преподаватель
Задание 1	40	Студенты выполняют, преподаватель помогает в решении проблем
Задание 2	40	Студенты выполняют, преподаватель помогает в решении проблем
Вопросы и обсуждение	20	Модерирует преподаватель
Длительность:	**120**	

4. Блок 1.

Тайминг:
Объяснение правил – 10 минут
Работа над заданием – 30 минут

Задание:

1. Создайте новый проект на базе Spring MVC.
2. Ваше приложение должно обрабатывать запрос GET /about.
3. При переходе по URL /about должна открываться статическая страница "О нас".
4. Страница "О нас" должна содержать следующую информацию: название вашей вымышленной компании, краткое описание деятельности и список основных участников с их ролями.
5. Всю эту информацию вы можете придумать самостоятельно.
6. Обратите внимание, что страница должна быть статической, то есть всю информацию вы заранее заносите в HTML-файл, а не получаете из базы данных или другого источника данных.
7. После создания приложения убедитесь, что при переходе по URL /about отображается страница "О нас" с корректной информацией.

Пример решения:

1. В файле pom.xml вам нужно добавить зависимости для Spring MVC:

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

2. Создайте класс `AboutController`:

```
    @Controller
    public class AboutController {

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

3. В каталоге `src/main/resources/static` создайте HTML-файл `about.html`. Этот файл будет вашей статической страницей "О нас". Он может выглядеть примерно так:

```
<!DOCTYPE html>
    <html>
    <head>
        <meta charset="UTF-8">
        <title>О нас</title>
    </head>
    <body>
        <h1>Компания XYZ</h1>
        <p>Мы занимаемся разработкой высококачественного
программного обеспечения.</p>
        <h2>Наша команда:</h2>
        <ul>
            <li>Иван Иванов - главный разработчик</li>
            <li>Петр Петров - тестировщик</li>
            <li>Сергей Сергеев - менеджер проекта</li>
        </ul>
    </body>
    </html>
```

4. Запустите приложение и откройте в браузере http://localhost:8080/about. Вы должны увидеть страницу "О нас".

Пожалуйста, обратите внимание, что адрес http://localhost:8080/about может отличаться в зависимости от настроек вашего приложения.

Часто встречающиеся ошибки:
Неправильное обработка HTTP-запросов или ответов.

5. Блок 2.

Тайминг:
Объяснение правил – 10 минут
Работа над заданием – 30 минут

Задание:

1. Создайте новый проект на базе Spring MVC.
2. Ваше приложение должно слушать на URL /random.
3. При переходе по URL /random должна открываться страница, на которой отображается случайное число в диапазоне от 1 до 100.
4. Число должно генерироваться на стороне сервера, а не в браузере пользователя.
5. При каждом обновлении страницы число должно меняться.
6. После создания приложения убедитесь, что при переходе по URL /random и последующем обновлении страницы отображается новое случайное число.

Пример решения:
1. Создайте новый Spring MVC проект. Это можно сделать, например, через Spring Initializr.
2. В `pom.xml` добавьте зависимости для Spring MVC:

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

3. Создайте контроллер для обработки запросов к URL `/random`:

```
    @Controller
    public class RandomController {

        @GetMapping("/random")
        public String getRandomNumber(Model model) {
            Random random = new Random();
            model.addAttribute("number",
random.nextInt(100) + 1);
            return "random";
        }
    }
```

Этот контроллер генерирует случайное число от 1 до 100 и добавляет его в модель под именем "number".

4. В каталоге src/main/resources/templates создайте файл random.html:

```
  <!DOCTYPE html>
    <html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Random Number</title>
    </head>
    <body>
        <h1>Случайное число: <span
th:text="${number}"></span></h1>
    </body>
    </html>
```

Этот файл отображает случайное число, полученное из модели.

5. Запустите приложение и откройте в браузере http://localhost:8080/random. При каждом обновлении страницы вы должны видеть новое случайное число.

Часто встречающиеся ошибки:

Неправильное использование аннотаций @Controller и @RestController.

6. **Домашнее задание**

Условие:

1. Создание базового веб-приложения Spring MVC

Начните с создания простого веб-приложения с использованием Spring MVC. Это может быть простой сайт, который выводит "Привет, мир!" на главной странице. Используйте аннотацию @Controller и @RequestMapping для маршрутизации запросов на эту страницу.

2. Добавление Thymeleaf в проект

Добавьте Thymeleaf в свое веб-приложение Spring MVC. Создайте простую страницу с некоторыми переменными, которые заполняются с помощью модели Spring MVC и отображаются на странице с использованием Thymeleaf.

3. Создание формы ввода и обработка данных формы

Создайте страницу с формой ввода, используя Thymeleaf для рендеринга формы. Затем создайте контроллер Spring MVC, который обрабатывает отправку формы и выводит полученные данные. Это может быть форма для регистрации или любая другая форма, которая собирает информацию от пользователя.