Model View Controller

with Spring Boot and Thymeleaf

Contents

- Model View Controller
  - the MVC pattern
  - Web application with Spring Boot and Thymeleaf
- More syntax from Thymeleaf
- More about the Spring framework
- Making exercises

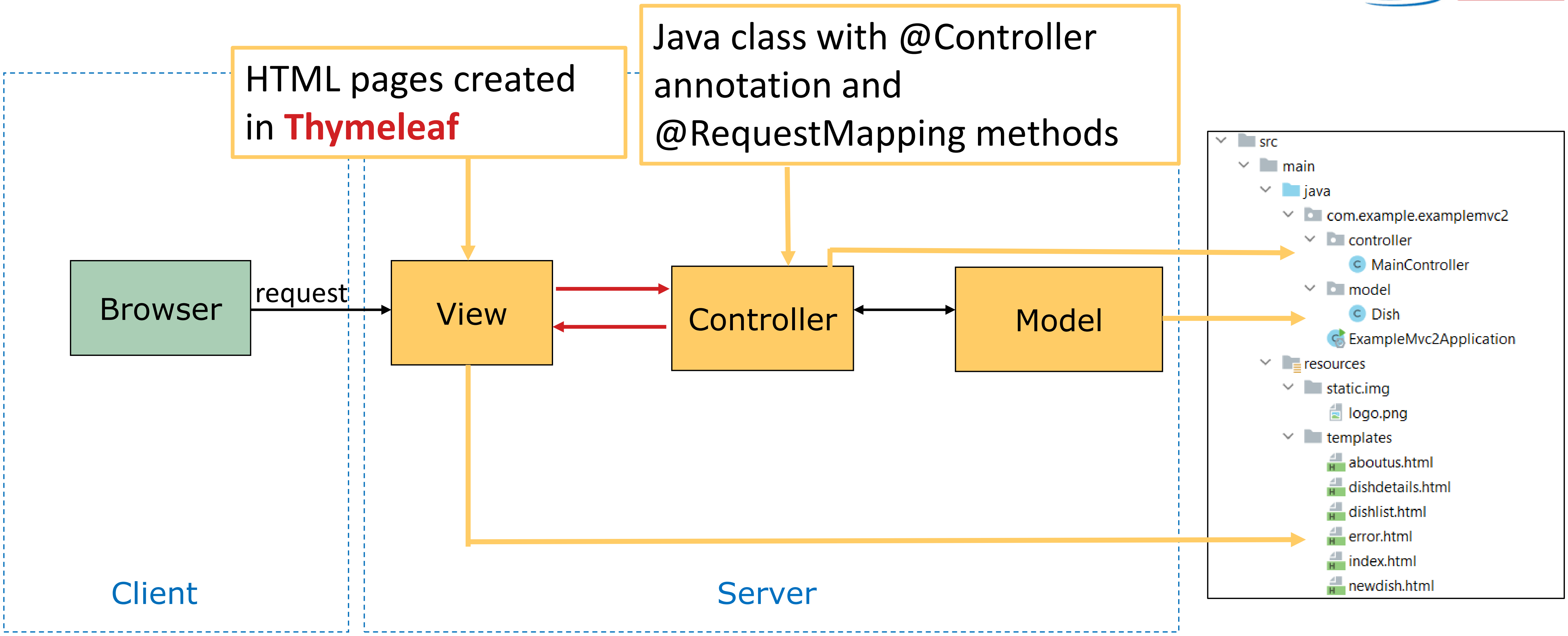
REHEARSAL: The MVC pattern

MVC stands for Model-View-Controller. This is a pattern that ensures that programmes are built in 3 layers for the sake of readability, maintainability and extensibility.

The 3 layers within the MVC pattern:

- In the back-end, there is the Model layer: this layer is filled with self-made classes that will be used to place data (attributes) and execute functionalities (methods).
- The Controller layer = the intermediate or middle tier: it contains the classes that control communication between the view and the model.
- The 'front-end' or the 'view' layer regulates the interface for the user. In a web application, this layer consists of web pages in which html, css, images, etc. are placed.

Spring Boot Web Applications with Thymeleaf



HTML pages created in **Thymeleaf**

Java class with @Controller annotation and @RequestMapping methods

```
src
  main
    java
      com.example.examplemvc2
        controller
          MainController
        model
          Dish
        ExampleMvc2Application
    resources
      static.img
        logo.png
      templates
        aboutus.html
        dishdetails.html
        dishlist.html
        error.html
        index.html
        newdish.html
```

What is Spring - Spring Boot - Benefits?

- Youtube video about spring / spring boot

What is Spring?

- Spring is one of the most widely used JEE frameworks for building applications for the java platform
- It aims to simplify the JEE development and helps developers be more productive at work
- Unlike other frameworks, spring focuses on several areas of an application and provides a wide range of features
- One of the major features of the spring framework is the dependency injection. It helps make things simpler by allowing us to develop loosely coupled applications

What is Spring Boot?

- While the spring framework focuses on providing flexibility to you, spring boot aims to shorten the code length and provide you with the easiest way to develop a web application. With annotation configuration and default codes, spring boot shortens the time involved in developing an application.
- It helps create a stand-alone application with less or almost zero-configuration
- Autoconfiguration is a special feature in spring boot. It automatically configures a class based on that requirements.

Benefits of Spring Boot

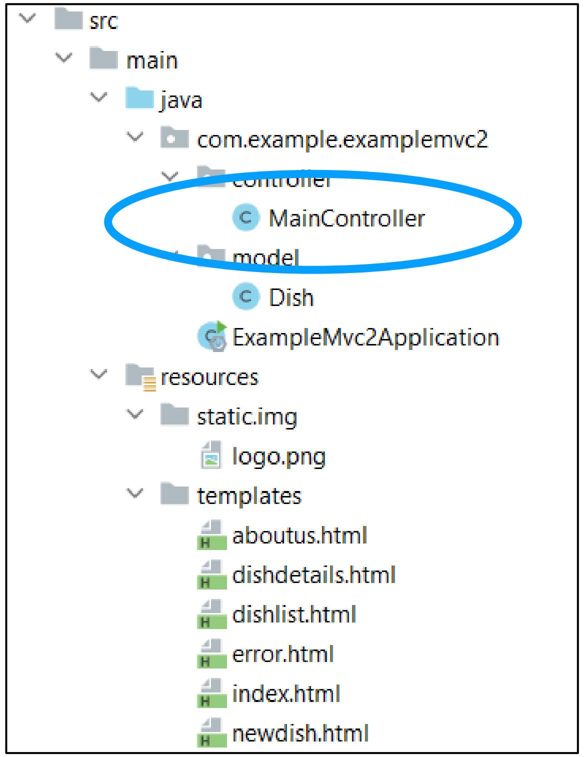
1. Dependency resolution
2. Minimum configuration
3. Embedded server for testing
4. Bean auto scan
5. Health metrics

The Spring and Spring Boot Framework

- To make the framework work, we are going to work with annotations in the code.
- Annotations start with @ and always refer to the code immediately following.
- Annotations are usually very short (and therefore seem unimportant) but they do have a major impact on the operation of your project.
- So don't forget them and put them in the right place...

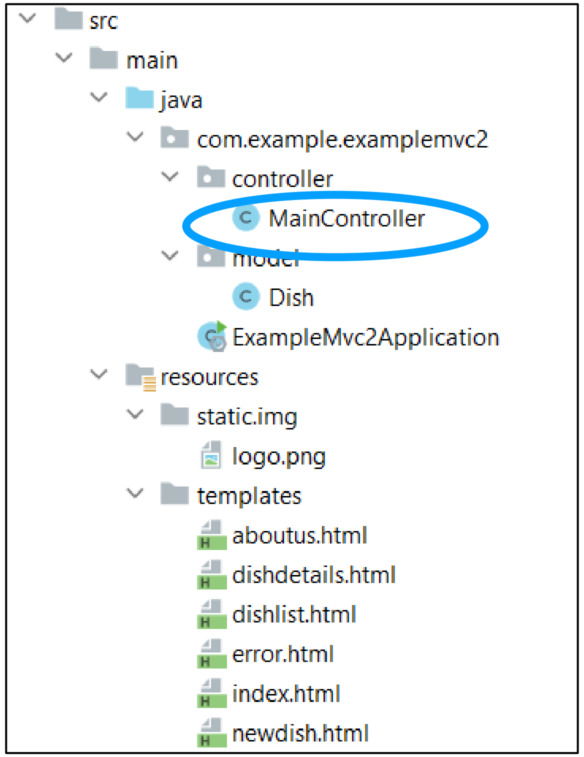
What is a Controller in Spring Boot?

- *The Spring Web MVC framework is a rich "model view controller" web framework. Spring MVC lets you create special @Controller (or @RestController) classes to handle incoming HTTP requests.*
- *Methods in your controller are mapped to HTTP by using @RequestMapping annotations*
- => no extra code is needed to make a class into a a controller from a class and to check the content of the HTTP requests. This is done by the annotations ...
- => BUT every http-request MUST now pass through the Controller. It is not possible in this framework to go from one html page directly to another html page...



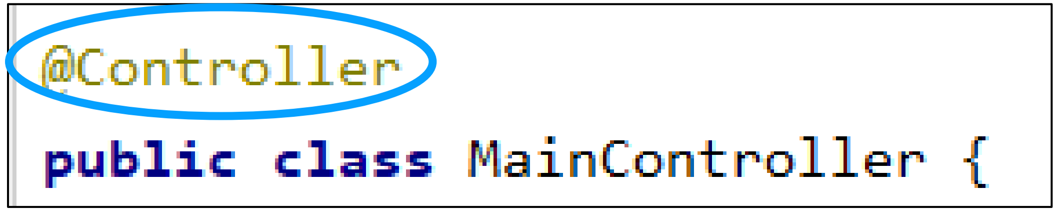
```
src
  main
    java
      com.example.examplemvc2
        controller
          MainController
        model
          Dish
        ExampleMvc2Application
    resources
      static.img
        logo.png
      templates
        aboutus.html
        dishdetails.html
        dishlist.html
        error.html
        index.html
        newdish.html
```

What is a Controller in Spring Boot?



```
src
  main
    java
      com.example.examplemvc2
        controller
          MainController
        model
          Dish
        ExampleMvc2Application
    resources
      static.img
        logo.png
      templates
        aboutus.html
        dishdetails.html
        dishlist.html
        error.html
        index.html
        newdish.html
```

- A controller is a class (with a capital letter!) in the folder/package *controller*
- The code must contain the annotation **@Controller** (just before the class):

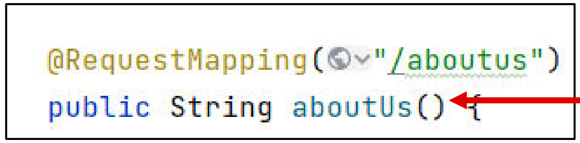


```
@Controller
public class MainController {
```

  - *annotation that makes it clear that this is not just a class, but a class that will work as a controller from the MVC pattern*
  - => when starting the application automatically (by the spring-boot framework) one object of the class MainController is created with which the html-request (post- and get-) can/will be received...

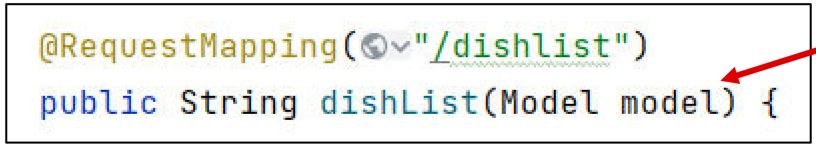
What is/was a Controller?

- every request coming from an HTML page must be "caught" by a method in a controller class that is preceded by the annotation @RequestMapping ("...")
- Such a method serves to
  1. redirect the user to the next HTML page



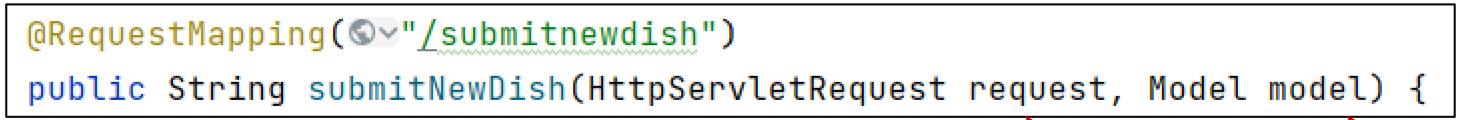
```
@RequestMapping("/aboutus")
public String aboutUs() {
```

  2. data / objects (with data in them) to that page



```
@RequestMapping("/dishlist")
public String dishList(Model model) {
```

  3. data/objects to that page and fetch data from the previous page



```
@RequestMapping("/submitnewdish")
public String submitNewDish(HttpServletRequest request, Model model) {
```

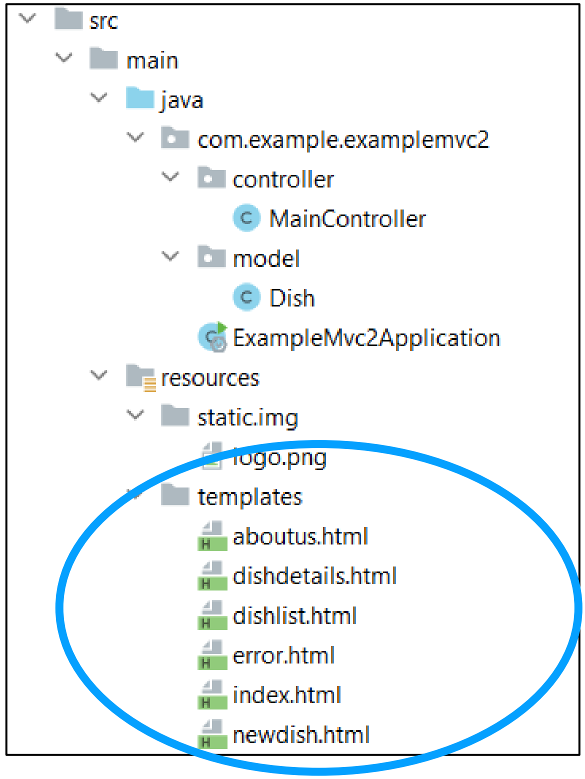
What is Thymeleaf?

- Thymeleaf is a server-side Java template engine
- Thymeleaf provides a number of additional attributes (**th:**) for use in the HTML tags of static HTML pages.
  Together with data from the model, these tags then dynamically add or remove content from the HTML page
- The extra attributes in the html tags are interpreted by the compiler - in the background - and converted into real HTML pages that can be displayed in the web browser.

What is Thymeleaf?

- The Thymeleaf HTML pages are located on the server in the "resources/templates" folder and they must contain the following code (at the top):

```
<html lang="en" xmlns:th="http://www.thymeleaf.org">
```



```
src
  main
    java
      com.example.examplemvc2
        controller
          MainController
        model
          Dish
        ExampleMvc2Application
    resources
      static.img
        logo.png
      templates
        aboutus.html
        dishdetails.html
        dishlist.html
        error.html
        index.html
        newdish.html
```

Thymeleaf-syntax - @{...}

- With **@{..}** you can refer to a URL
- Required to be used in th:action (instead of action), th:href (instead of href), th:src (instead of src)
  - e.g. `th:action="@{/samplewithdata}".`
- Benefit: You can dynamically add content to the URL
  - e.g. `th:href="@{/samplemetadata(id=${customer.getId()})}"`
  - This results, for example, in a URL `href="examplemetrics?id=2".`
  - The content of the parameter "id" can be retrieved (as we did last year) via request.getParameter see
- Documentation on
  - https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html#link-urls
- NB Images that you wish to display in your web applications should be placed in the "static" folder under src/resources. See lesson example.

Thymeleaf Syntax: Standard Expression Syntax

- Within the "Variable Expressions": ${...} you can use
  - Literals: 'text', null, true, false, ...
  - Arithmetic operations: +, -, *, /, %
  - Boolean operations: and, or, !, not
  - Comparison and equality: >, <, >=, <=, ==, !=
  - Conditional operators: (if) ? (then) : (else)
- Documentation and examples can be found here:
  - https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html#standard-expression-syntax

Thymeleaf syntax: Expression Utility Objects

- You can use interesting methods to format dates, numbers, Strings etc... by using Thymeleaf's "Expression Utility Objects". You do this with the prefix **#**
  - eg. `<span th:text= "${#strings.toUpperCase(str)}" />` will convert the contents of the variable str to upper case, but of course you can still use the java equivalent:
    `<span th:text= "${str.toUpperCase()}" />`
  - The #numbers and its methods are interesting on the other hand, e.g. to build an iteration:

    ```
    < th:block th:each="i: ${#numbers.sequence(2015, 2020, 1)}">
    ```
- More information and examples can be found at:
  - https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html#appendix-b-expression-utility-objects
  - https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html#expression-utility-objects

More syntax on Spring Boot and Thymeleaf

- General
  - See "**Thymeleaf Spring Cheatsheet**" on Canvas
  - https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html#introducing-thymeleaf

Exercises

- Open the “example-mvc-2”-project to refresh your memory
- There are NO start folders for creating the exercises. You must create your own application folder in IntelliJ. On Canvas you will find the document “How to create a project in IntelliJ”
- Create the package *model* and put the requested Java classes in it
  - Ensure exact same naming conventions
  - Please note:
    - class names always begin with capital letters
    - packages, attributes and methods begin with lower case
  - Make sure you have the right imports (in case of doubt => look at the example project)
- Create the package *controller* and place the requested controller class in it:
  - Don't forget to put @Controller at the top of the class!!!
  - Make sure you have the right imports (also here)
- When in doubt => look at the example project *example-mvc-2*