Java Programming 3

Thymeleaf - Bootstrap

KdG

Agenda this week

Last week - project Review
Thymeleaf
Bootstrap

KdG

Agenda this week

Last week - project Review
Thymeleaf
Bootstrap

KdG

Last week - Project Review

- Last week review:
  - Application Context - Logging - Intro SpringMVC
- Project Remarks:
  - Process the inputfields of the form?
    - You could create a special container class for that: “ViewModel”

```
public class AuthorViewModel {
   private String name;
   private String birthday;
   private String nationality;
   …
```

More tips and tricks on how to process the form in your Controller next week, when we look at Controllers in detail!

```
@PostMapping("/add")
public String processAddAuthorForm(AuthorViewModel authorViewModel) {
   log.debug("Processing addauthor form...");
   LocalDate birthday = LocalDate.parse(authorViewModel.getBirthday());
   authorService.addAuthor(authorViewModel.getName(), birthday,
authorViewModel.getNationality());
```

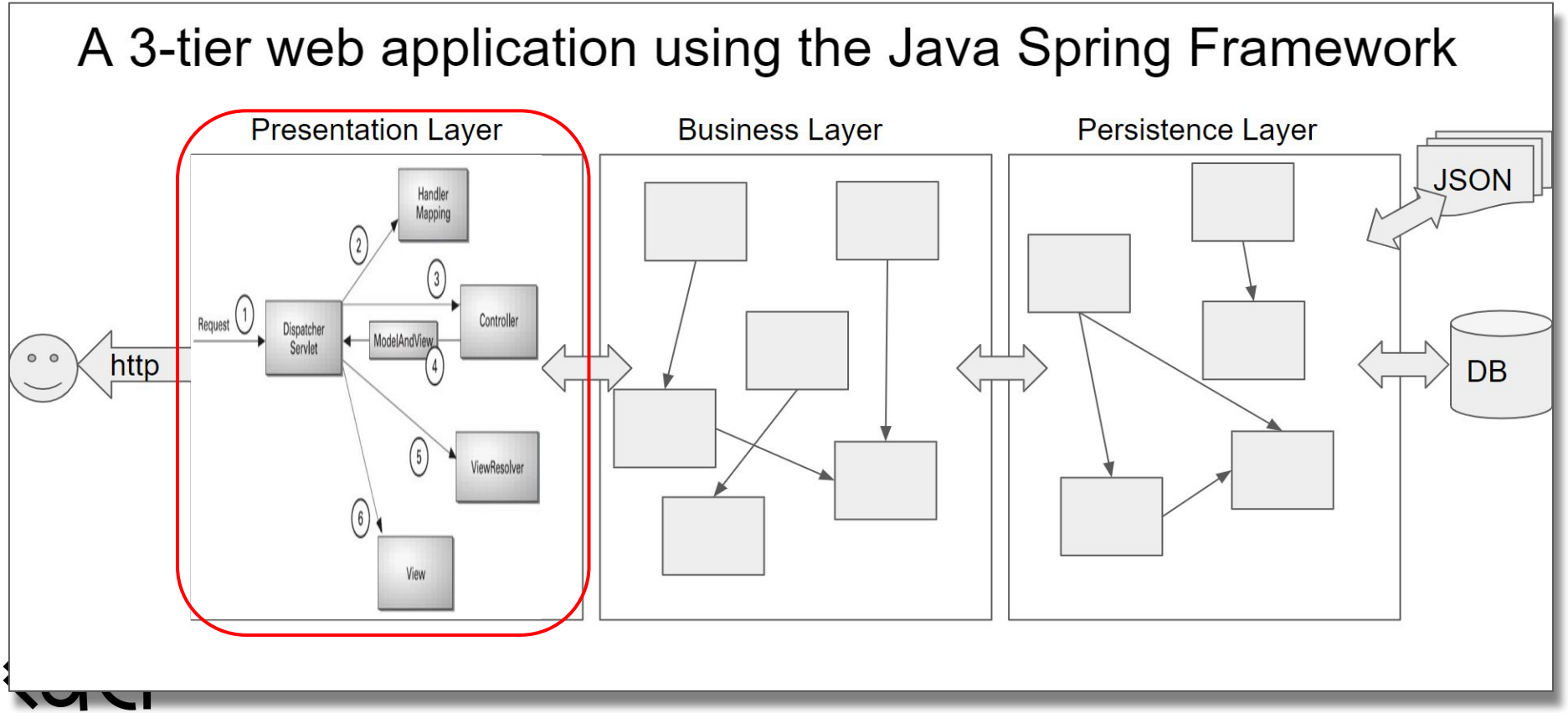
KdG

Agenda this week

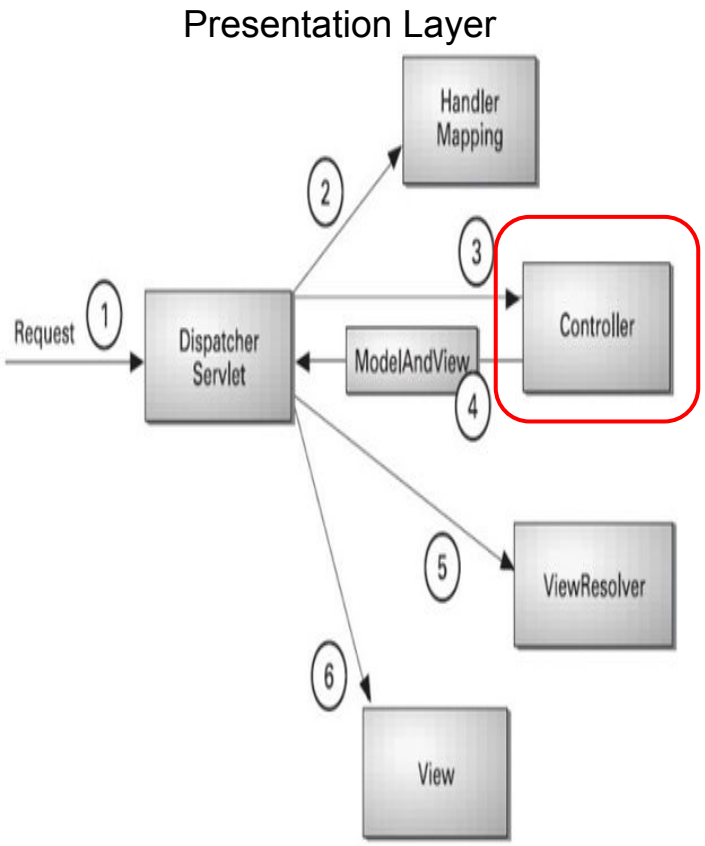
Last week - project Review
Thymeleaf
Bootstrap

KdG

Recap: the Presentation Layer



Presentation Layer: Controllers and Views

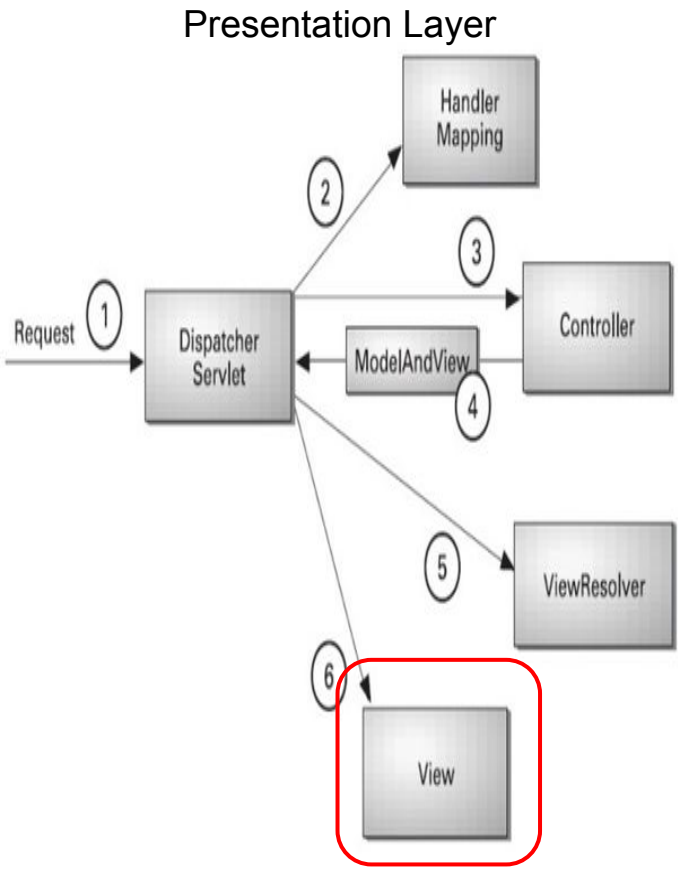


Controllers:

- We will write Controllers using Java (with @Controller annotation).
- We typically create a controller for each 'entity' in the domain, although it depends on the application...
- Each controller can then have methods voor getting (list of) entity/ies and for adding entities and/or updating and/or deleting.
- Controller talks to the service layer
- The controller passes data and the logical name of the view to the DispatcherServlet, who then runs the (Thymeleaf) view with this data

KdG

Presentation Layer: Controllers and Views

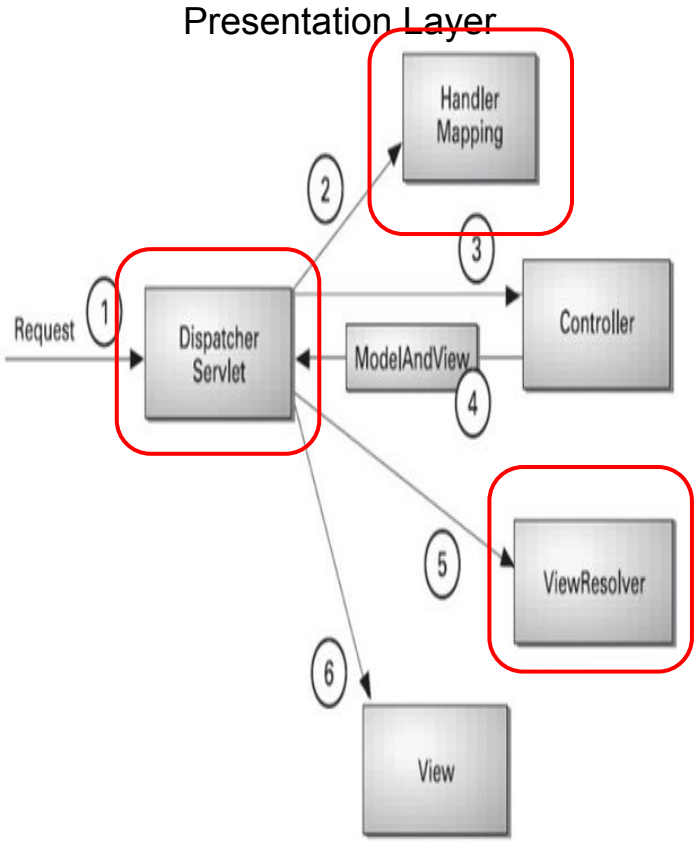


Views:

- We will write Views using Thymeleaf templates
- We create a Thymeleaf view for each page of our application
- The Controller decides what view is necessary and passes on data to the view (via the DispatchServlet)
- Thymeleaf template generates the HTML using the data it received from the Controller

KdG

Presentation Layer: Controllers and Views




```
DispatherServlet, HandlerMapping,
ViewResolver
```

- These object are automatically created by the Spring framework and added to the Spring container
- They are configured for Thymeleaf because Spring Boot finds the Thymeleaf dependency in the classpath. (This process is called “autoconfiguration”)

KdG

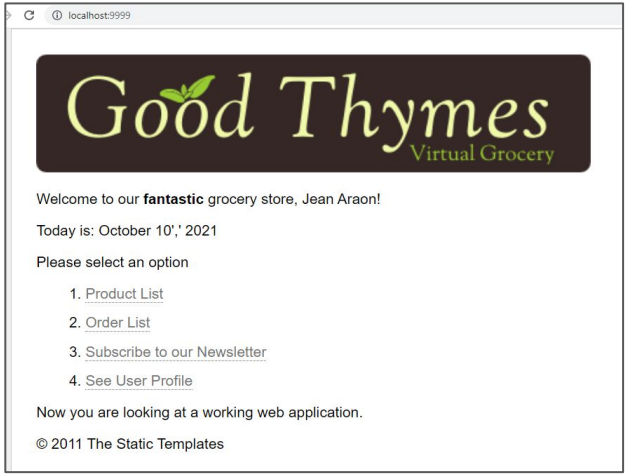
Thymeleaf

- a server-side Java template engine
- main goal is to bring *natural templates* — HTML that can be correctly displayed in browsers and also work as static prototypes
- With modules for Spring Framework

KdG

Thymeleaf Spring example project

- Clone the [Good Thymes Virtual Grocery example project]
  - This project is used as an example in the thymeleaf documentation:
    - [Using Thymeleaf]
    - Thymeleaf and Spring: https://www.thymeleaf.org/doc/tutorials/3.1/thymeleafspring.html
  - Run it!



KdG

Five different *Expression* types

- Tutorial: Getting started in 5 minutes
  - @{...} → URL expressions
  - #{...} → Message expressions
  - ${...} → Variable expressions
  - ~{...} → Fragment expressions
  - *{...} → Selection variable expressions

KdG

Exercise 1.1

- Run the Good Thymes application - Test it!
- On what port does the web application run?
- Inspect the Java code
  - Draw a UML diagram for the Domain Model
  - Inspect the different layers of the application
    - How do they differ from our implementations?
    - What would you change?

KdG

Exercise 1.2: inspect home.html

- Look up and explain each of the following th:attributes in detail!
  - In <link>:
    - `th:href="@{/css/gtvg.css}"`
  - In <img>:
    - `th:src="@{/images/...}”`
    - `th:alt-title=”#{logo}”`
  - In <p>:
    - `th:utext="#{home.welcome(${session.user.name})}`
  - In :
    - `th:with="df=#{date.format}"`
    - `th:text="${#calendars.format(session.today,df)}`
  - In <a>:
    - `th:href="@{/product/list}”`
  - In :
    - `th:insert="~{footer::copy}"`

KdG

Exercise 1.3: inspect product/list.html

- Look up and explain each of the following th:attributes in detail!
  - In <tbody>:
    - `th:remove=”all-but-first”`
  - In <tr>:
    - `th:each="prod : ${prods}"`
    - `th:class="${prodStat.odd}? 'odd'"`
  - In <td>:
    - `th:text="${prod.inStock}? #{true} : #{false}"`
  - In :
    - `th:text="${#lists.size(prod.comments)}`
  - In <a>:
    - `th:unless="${#lists.isEmpty(prod.comments)}"`

KdG

@{...}: URL Expressions

- Will be replaced by path to the correct resource
- Can be in static or in templates folder
- Can have parameters: will be added to the URL as URL parameters
- Examples:

```
<link rel="stylesheet" type="text/css" media="all" href="../../css/gtvg.css"
th:href="@{/css/gtvg.css}" />
```

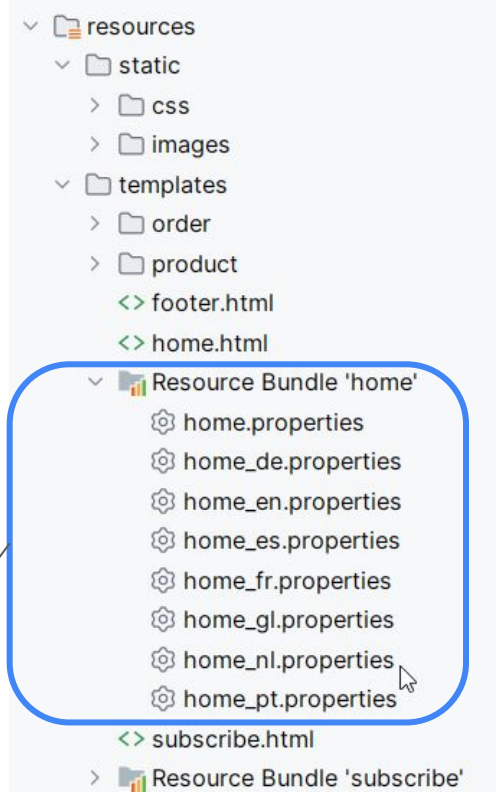
```
<img src="../../images/gtvglogo.png" alt="GTVG logo" title="GTVG logo"
    th:src="@{/images/gtvglogo.png}" th:alt-title="#{logo}" />
```

```
<a th:href="@{/order/details(id=${orderId},type=${orderType})}">
```

#{...} → Message expressions

- They refer to a value in a resource bundle
  - Resource Bundle = set of .properties files, one for each language, containing strings
  - IntelliJ has a plugin that simplifies working with resource bundles
  - Default location: resources/messages.properties (messages_nl.properties, …)

```
home.welcome=Welkom in onze <b>kruidenier</b>, {0}!
logo=Logo Good Thymes Virtual Grocery
date.format=dd MMMM'','' yyyy
```



KdG

#{...} → Message expressions

- Message expressions can have parameters
- Examples:

```
<img th:src="@{/images/gtvglogo.png}" th:alt-title="#{logo}" />
```

```
<p th:utext="#{home.welcome(${session.user.name})}">Welcome!</p>
```

Unescaped text

KdG

${...} → variable expressions

- In variable expressions you can use the attributes that were added to the Model object in the Controller class.
- You can also make use of the *Expression Objects* using the #
- Examples:

```
@Controller
public class ProductListController {

   @RequestMapping("/product/list")
   public String product(Model model){
       model.addAttribute("prods",prods.findAll());
       return "product/list";
   }
```

```
<tr th:each="prod : ${prods}" th:class="${prodStat.odd}? 'odd'">
```

```
<span th:text="${#lists.size(prod.comments)}">2</span>
```

~{...} → Fragment expressions

- You can refer to reusable HTML snippets called “fragments”
- The fragments reside in another file and have the th:fragment attribute
- Typically used for headers and footers.
- They can also have parameters

In footer.html

```
<div th:fragment="copy">
 &copy; 2011 The Good Thymes Virtual Grocery
</div>
```

```
<div th:insert="~{footer::copy}">&copy; 2011 The Static Templates</div>
```

KdG

*{...} → Selected variable expressions

- You first select a variable with th:object="${...}"
- You can now reference it using *{...}
- Example:

```
<div th:object="${order.customer}">
 <p><b>Name:</b> <span th:text="*{name}">Frederic Tomato</span></p>
 <p><b>Since:</b> <span th:text="*{#calendars.format(customerSince,'dd MMM
yyyy')}">13 jan 2011</span></p>
</div>
```

For java 8 LocalDate use #temporals

KdG

And there's more

- Variables (`th:with`)
- Arithmetic operations
- Comparators
- Conditional HTML (`th:if, th:unless`)
- Iterations (`th:each`)
  - Status of iterated object (Stat suffix)
- Call methods on objects
  - Use with extreme care: do not put logic in your templates!

See

- Introduction to using thymeleaf in Spring (start at section 5)
- https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html

KdG

Not only for HTML…

- CSS Templates
- JavaScript Templates
- Text Templates
- XML Templates
- Raw Templates

KdG

Exercise 2: Use Thymeleaf

- Download the SMSWithThymeleaf start application and inspect
- Add the following to the allstudents.html template:
  - Add a CSS file to the static folder and add a link to it in the template
  - Show the number of students in the page title: “A list of .. students”
  - Show a table of all the students
    - Add table header: Name - Birthday
    - Show name and birthday of all the students
    - Show birthday in a nice format → use Expression Objects
    - Show odd lines in the table in darker color → use Stat suffix
    - Add footer with copyright information → use a Fragment
    - Add a welcome to the user to the top left of the page → use a Fragment with a parameter
  - Add an image to the page → use @{..} to the image in the static folder
  - Show the user at the top right (“Welcome …”)
  - Provide a Resource Bundle: the page should be available in 2 languages
    - Messages: the page title, the welcome message, the table header and the footer

KdG

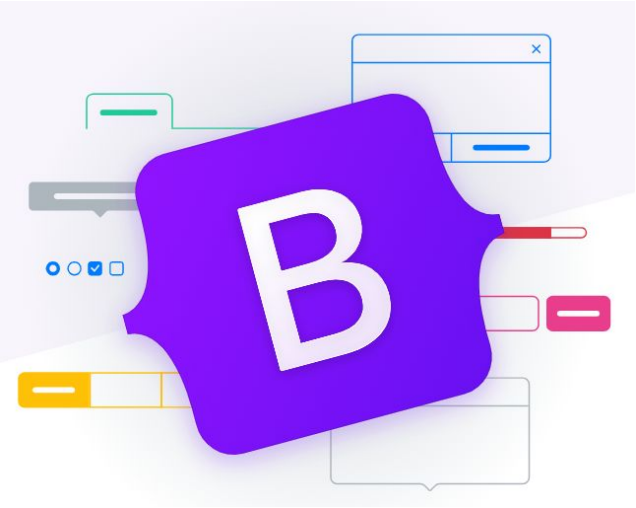
Agenda this week

Last week - project Review
Thymeleaf
Bootstrap

KdG

Bootstrap

- Bootstrap is a CSS framework to jumpstart web development providing
  - A responsive grid system
  - Prebuilt components with behaviour built-in using javascript
    - Navbar
    - Form
    - Carousel
    - …

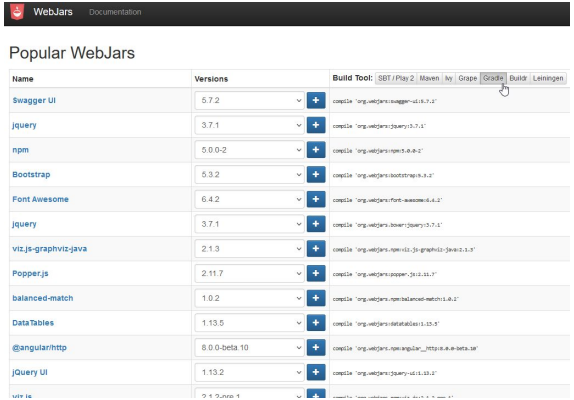


KdG

Bootstrap: add dependency using WebJars

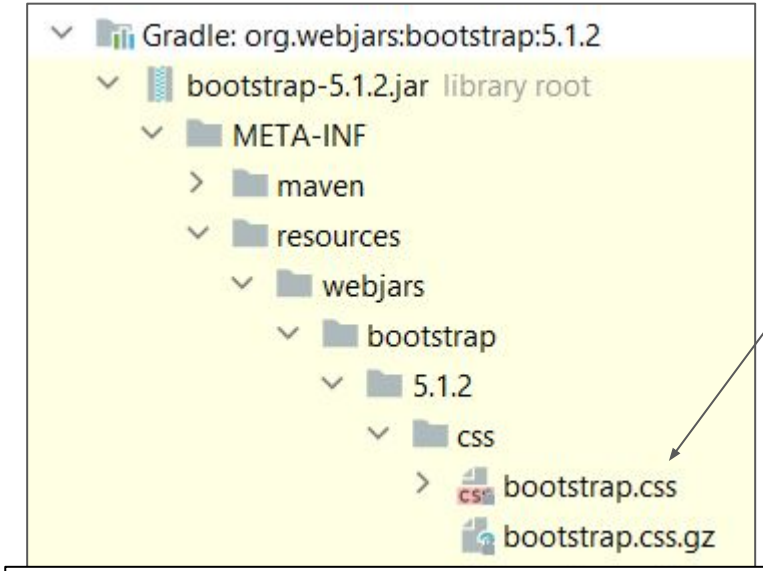
- We can use Gradle to download Client side web libraries like Bootstrap for us!
- WebJars wraps them into a jar (Java Archive) that can be added as a dependency to the build.gradle file for your server

```
implementation("org.webjars:bootstrap:5.3.2")
```



KdG

WebJars : check the External Libraries in IntelliJ

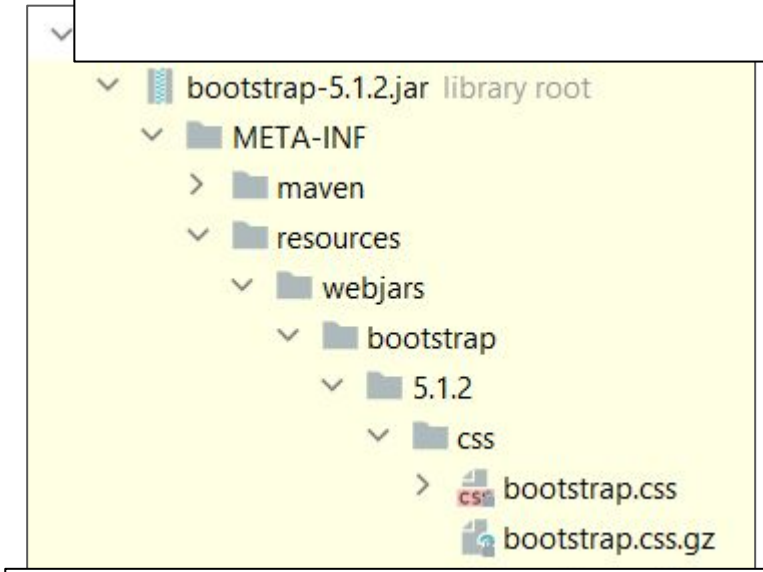


The Bootstrap jar is included in the classpath, so you can access the bootstrap CSS and JavaScript files from within Thymeleaf…
Adapt for your Bootstrap version :(

```
<link rel="stylesheet" th:href="@{/webjars/bootstrap/5.1.2/css/bootstrap.min.css}"/>
<script defer th:src="@{/webjars/bootstrap/5.1.2/js/bootstrap.bundle.min.js}">
</script>
```

Webjars-locator: version agnostic references!

```
implementation("org.webjars:bootstrap:5.3.2 ")
implementation("org.webjars:webjars-locator-core:0.48")
```



If we add the webjars-locator dependency to our project, Spring Boot will use this library to deduce the correct version of the webjar: we don’t need to add the version number in our references in HTML!

```
<link rel="stylesheet" th:href="@{/webjars/bootstrap/css/bootstrap.min.css}"/>
<script defer th:src="@{/webjars/bootstrap/js/bootstrap.bundle.min.js}">
</script>
```

Bootstrap: does it work?

```
...
<h1>Hello World</h1>
<div class="container">
   <div class="alert alert-warning alert-dismissible fade show"
role="alert">
       <strong>Holy guacamole!</strong> You should check in on some of those
fields below.
       <button type="button" class="btn-close" d
aria-label="Close"></button>
   </div>
</div>
...
```

Let’s add a Bootstrap alert to our index.html and test if it works. Don’t forget to add the <link> and <script> tags to the Bootstrap files! Clicking the X should close the Alert!

Holy guacamole! You should check in on some of those fields below.

Bootstrap: why the script tag?

```
<link rel="stylesheet" th:href="@{/webjars/bootstrap/css/bootstrap.min.css}"/>
<script defer th:src="@{/webjars/bootstrap/js/bootstrap.bundle.min.js}">
</script>
```

Bootstrap is more than just CSS:it also includes some JavaScript code, for example to close the Alert when clicking on the X!

Holy guacamole! You should check in on some of those fields below.

Bootstrap: what’s this .min.css and .bundle.min.js?

```
<link rel="stylesheet" th:href="@{/webjars/bootstrap/css/bootstrapmin.css}"/>
<script defer th:src="@{/webjars/bootstrap/js/bootstrapbundle.min.js}">
</script>
```

.min stands for “minified”: it is CSS but all unnecessary characters are removed to make it as small as possible.

The **bundle** means that it bundles more than one js file: it also includes the popper.js library which Bootstrap uses for popups...

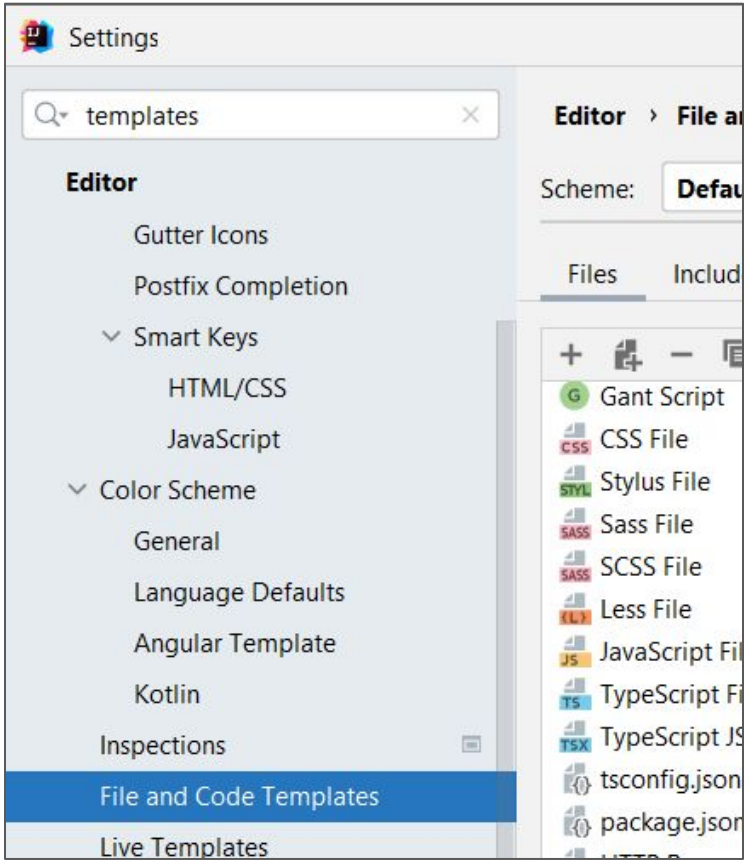
Holy guacamole! You should check in on some of those fields below.

Exercise 3: Bootstrap with webjars

- Use the SMSWithThymeleaf project
- Add dependencies to the Bootstrap webjar and webjars-locator
- Include the <link> and <script> tags: does it work?
- What happens if you remove the script tag?
- Tip: add a new template to IntelliJ that create a Thymeleaf starter page which includes the correct <link> and <script> tag

KdG

Tip: add a Code Template to IntelliJ!



KdG

Tip: add a Code Template to IntelliJ!

Files | Includes | Code | Other

- Gant Script
- CSS File
- Stylus File
- Sass File
- SCSS File
- Less File
- JavaScript File
- TypeScript File
- TypeScript JSX File
- tsconfig.json
- package.json
- HTTP Request
- HTTP Request Scratch
- HTTP Public Environment File
- HTTP Private Environment File

Name: Thymeleaf with Bootstrap Extension: html

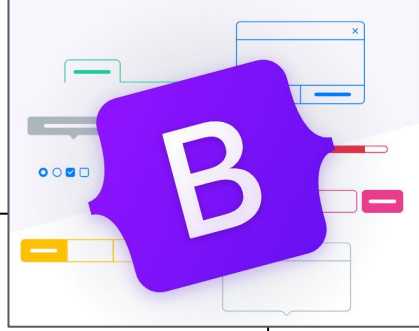
File name: Template to generate file name and path (optional)

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>#[[$Title$]]#</title>
    <link rel="stylesheet" th:href="@{/webjars/bootstrap/css/bootstra
    <script defer th:src="@{/webjars/bootstrap/js/bootstrap.bundle.mi
</head>
<body>
#[[$END$]]#
</body>
```

☑ Reformat according to style ☑ Enable Live Templates

KdG

Bootstrap example

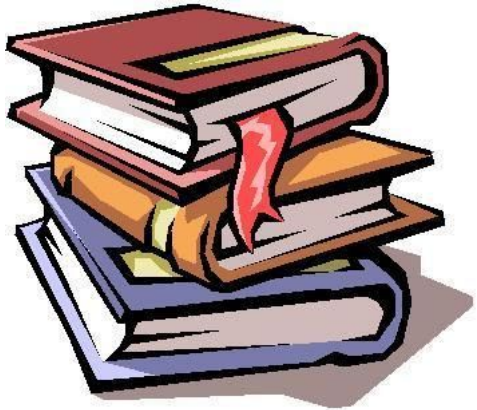



```
<div class="container">
<form class="row g-3 needs-validation" novalidate>
  <div class="col-md-4">
    <label for="validationCustom01"  class="form-label">First
name</label>
    <input type="text"  class="form-control" id="validationCustom01"
value="Mark" required>
    <div class="valid-feedback">
      Looks good!
    </div>
  </div>
  <div class="col-md-4">
    <label for="validationCustom02"  class="form-label">Last
name</label>
    <input type="text"  class="form-control" id="validationCustom02"
value="Otto" required>
    <div class="valid-feedback">
      Looks good!
    </div>
```

Bootstrap uses CSS classes. By adding them certain CSS will be applied to these elements.
→ **not always very ‘semantic’ HTML!**

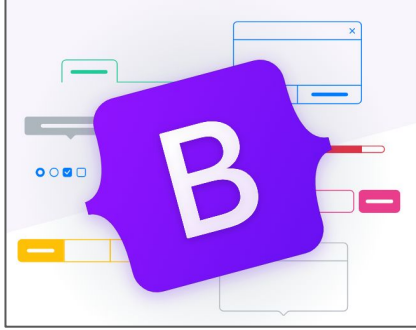
Bootstrap tutorial

- The Bootstrap documentation is good with live examples embedded.
- It covers a lot of topics!
  - Layout
    - Containers
    - Grid System
    - Fixed and Fluid Layouts
    - Responsive Layouts
  - Content
    - Typography
    - Tables
  - Bootstrap icons are in a separate package
  - …
  - There is also a section with more examples.
- There is also a good tutorial at https://www.tutorialrepublic.com/twitter-bootstrap-tutorial



KdG

The Bootstrap Grid System: basic idea



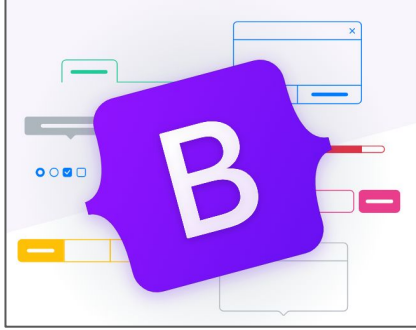
takes 3/12 of row	takes 4/12 of row	takes 5/12 of row

```
<div class="container">
 <div class="row">
   <div class="col-3">
     takes 3/12 of row
   </div>
   <div class="col-4">
     takes 4/12 of row
   </div>
   <div class="col-5">
     takes 5/12 of row
   </div>
 </div>
</div>
```

- All content is in a .container
- Containers have horizontal .row children
- The immediate children of .row elements are vertical .col (column) elements
- A row is divided into 12 equal columns. You can define the number of columns a child occupies by adding a number to the class: for example .col-3 use 3 out of the 12 columns.
- .col without a number divides remaining columns evenly

KdG

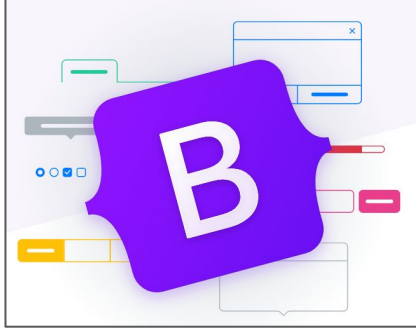
Responsive Grid Breakpoints



	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

KdG

The Bootstrap Grid System: combining .col



col-sm-3 col-md
col-sm-9 col-md-8
col-sm-12 col-md

col-sm-3 col-md	col-sm-9 col-md-8
col-sm-12 col-md	

col-sm-3 col-md	col-sm-9 col-md-8	col-sm-12 col-md

- By using different versions of the .col classes on the same element you can add responsiveness to the website
- xs: no classes: all divs take 1 row
- sm: 3-9-12 ratio: over 2 rows
- md: 2-8-2 ratio

```
<div class="row">
   <div class="col-sm-3 col-md">col-sm-3 col-md</div>
   <div class="col-sm-9 col-md-8">col-sm-9 col-md-8</div>
   <div class="col-sm-12 col-md">col-sm-12 col-md</div>
</div>
```

https://codepen.io/jdrijke/pen/OGvYGO

Contest!

- In the following exercises you will use bootstrap to create a professional looking web application in no time!
  - Use the grid system to make it responsive
  - Use forms with client side validation
  - Update the look of your table
  - Add nice looking icons
  - Add a responsive navigation bar on top of your pages
  - …
- who creates the nicest web application?

KdG

Exercise 4: Bootstrap gridsystem & forms

- Read the section in the tutorial
- https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-grid-system.php
- https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-forms.php
- Add a addstudent page to the SMSWithThymeleaf application
  - Create a nice looking form to be able to add a student
  - Be sure to put the same footer and welcome user message on the page (use Fragments!)
  - Add some extra fields to the student:
    - Adres
    - Phone
    - E-mail
  - Test on different screen sizes!

KdG

Exercise 5: Bootstrap tables

- Read the section in the tutorial
- https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-tables.php
- Update the layout of your table in the allstudents page. Use bootstrap to do the coloring of the odd rows…

KdG

Exercise 6: Bootstrap icons

- Read the section in the tutorial
- https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-icons.php
- Lookup the different Bootstrap icons and try adding some to your pages:
  - Add a special icon to students born after 2000?
  - Add an icon to your footer information?
  - …
- Bootstrap fonts are in a separate npm package
  - Any npm package can be added to gradle as a webjar. In build.gradle.kts:

```
implementation("org.webjars.npm:bootstrap-icons:1.11.1")
```

  - In your webpage load the CSS from gradle

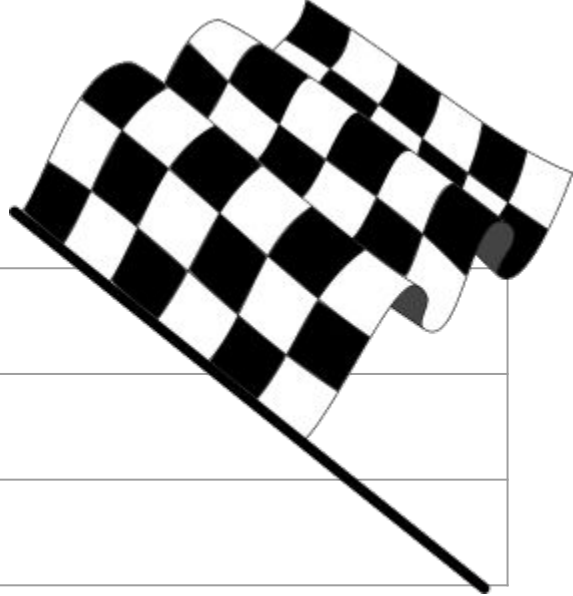
```
<link rel="stylesheet" th:href="@{/webjars/bootstrap-icons/font/bootstrap-icons.css} ">
```

Exercise 6: Bootstrap icons

- Note: instead of hosting web dependencies on your server using webjars, you can also load bootstrap and bootstrap icons from a CDN on the internet:

```
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-icons.css"
>
```

Agenda this week



Last week - project Review
Thymeleaf
Bootstrap

KdG

Project

- Use Thymeleaf and Bootstrap to work out the Views of your web application:
  - Include Bootstrap using webjars
  - In your pages:
    - Make them responsive: they look nice on any screen size
    - Update the layout of your tables
    - Add a bootstrap Navbar (use fragments!)
    - Add a footer (use fragments!)
    - If I click on an item, I get a new page containing all details of this item.
    - Use bootstrap forms to update the look of your add... pages.
    - Implement client-side validation (required fields)
  - Add support for a second language to your website: all text comes from properties files.

KdG