

Dasar-Dasar Pemrograman 2

Lab 10

Spring Boot



FAKULTAS
ILMU
KOMPUTER

Dokumen ini berisi pengenalan Spring Boot dan tutorial pembuatan Spring Boot *project* dengan **Eclipse** atau **VS Code**. Silakan sesuaikan dengan IDE/Editor yang kamu pakai.

Jika kamu menggunakan **IntelliJ**, cukup buka <https://start.spring.io>, *generate project* Spring Boot yang diinginkan (disarankan menggunakan Gradle), ekstrak templatnya, dan buka dengan IntelliJ.

Membuat Spring Boot Project Menggunakan Eclipse

1. Menginstall Maven:

Kamu bisa menggunakan Maven atau Gradle (silakan pilih salah satu ketika membuat *project* Spring nanti) untuk mengatur proyek Spring Boot. Contoh untuk Eclipse kali ini menggunakan Maven.

- Buka <https://maven.apache.org/download.cgi> lalu download Binary zip archive (File : *apache-maven-3.6.3-bin.zip*)

Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

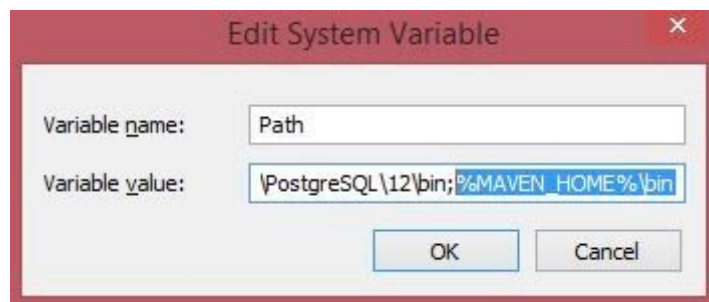
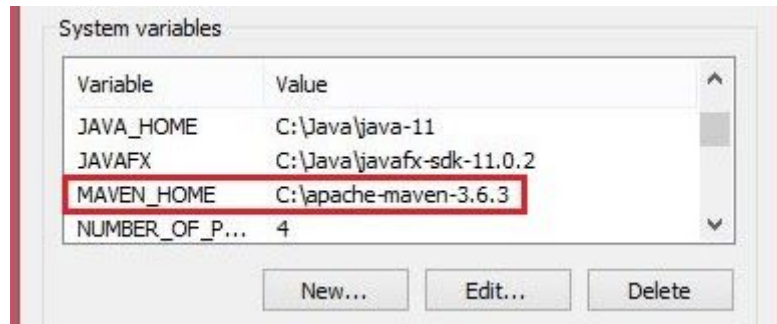
In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public **KEYS** used by the Apache Maven developers.

	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.6.3-bin.tar.gz	apache-maven-3.6.3-bin.tar.gz.sha512	apache-maven-3.6.3-bin.tar.gz.asc
Binary zip archive	apache-maven-3.6.3-bin.zip	apache-maven-3.6.3-bin.zip.sha512	apache-maven-3.6.3-bin.zip.asc
Source tar.gz archive	apache-maven-3.6.3-src.tar.gz	apache-maven-3.6.3-src.tar.gz.sha512	apache-maven-3.6.3-src.tar.gz.asc
Source zip archive	apache-maven-3.6.3-src.zip	apache-maven-3.6.3-src.zip.sha512	apache-maven-3.6.3-src.zip.asc

- Extract di C:\apache-maven-3.6.3 (Taruh file zip di C:\ lalu “Extract here”)

This PC > Local Disk (C:) > apache-maven-3.6.3				
Name	Date modified	Type	Size	
bin	11/7/2019 12:32 PM	File folder		
boot	11/7/2019 12:32 PM	File folder		
conf	11/7/2019 12:32 PM	File folder		
lib	11/7/2019 12:32 PM	File folder		
LICENSE	11/7/2019 12:32 PM	File	18 KB	
NOTICE	11/7/2019 12:32 PM	File	6 KB	
README.txt	11/7/2019 12:32 PM	Text Document	3 KB	

- Tambahkan pada *System Variable* di *System Environment*:
 - MAVEN_HOME = C:\apache-maven-3.6.3
 - Tambahkan pada PATH: %MAVEN_HOME%\bin



Pastikan maven sudah dapat digunakan dengan cara berikut:

```
C:\Users\Fairuzi>dir %MAVEN_HOME%
Volume in drive C has no label.
Volume Serial Number is C80A-364D

Directory of C:\apache-maven-3.6.3

11/07/2019  12:32 PM    <DIR>          .
11/07/2019  12:32 PM    <DIR>          ..
11/07/2019  12:32 PM    <DIR>          bin
11/07/2019  12:32 PM    <DIR>          boot
11/07/2019  12:32 PM    <DIR>          conf
11/07/2019  12:32 PM    <DIR>          lib
11/07/2019  12:32 PM                17,504 LICENSE
11/07/2019  12:32 PM                5,141 NOTICE
11/07/2019  12:32 PM                2,612 README.txt
               3 File(s)          25,257 bytes
               6 Dir(s)  88,960,172,032 bytes free

C:\Users\Fairuzi>mvn -v
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: C:\apache-maven-3.6.3\bin\..
Java version: 13.0.2, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-13.0.2
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Jika menggunakan Unix-like OS (contoh: MacOS/Linux), tambahkan \$MAVEN_HOME sesuai folder mavenmu dan ubah \$PATH pada ~/.bash_profile, ~/.bashrc, atau ~/.zshrc (targantung shell yang kamu gunakan) seperti berikut:

```
nano ~/.bash_profile
GNU nano 2.0.6 File: /Users/pt.gojek/.bash_profile

[[ -s "$HOME/.profile" ]] && source "$HOME/.profile" # Load the default .profile

[[ -s "$HOME/.rvm/scripts/rvm" ]] && source "$HOME/.rvm/scripts/rvm" # Load RVM$

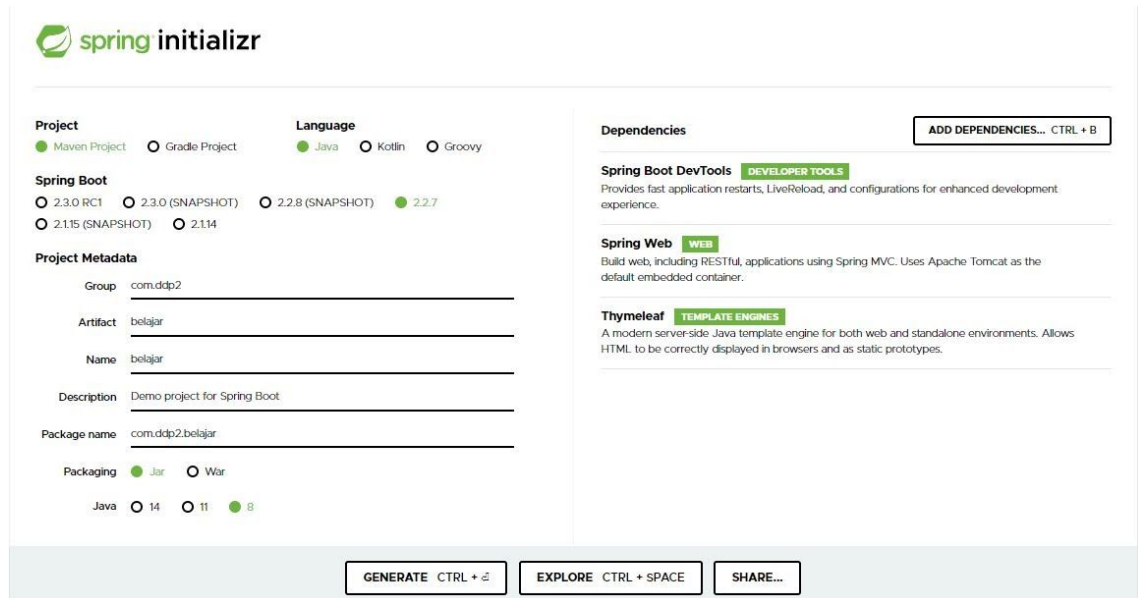
export MAVEN_HOME="$HOME/Downloads/apache-maven-3.6.3"
export PATH="$PATH:$MAVEN_HOME/bin"
```

lalu source file tersebut dan pastikan mvn sudah dapat digunakan

```
pt.gojek@PTs-MacBook-Air: ~
pt.gojek@PTs-MacBook-Air ~ nano ~/.bash_profile
pt.gojek@PTs-MacBook-Air ~ source ~/.bash_profile
pt.gojek@PTs-MacBook-Air ~ ls $MAVEN_HOME
LICENSE NOTICE README.txt bin boot conf lib
pt.gojek@PTs-MacBook-Air ~ mvn -v
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: /Users/pt.gojek/Downloads/apache-maven-3.6.3
Java version: 14.0.1, vendor: Oracle Corporation, runtime: /Library/Java/JavaVir
tualMachines/jdk-14.0.1.jdk/Contents/Home
Default locale: en_ID, platform encoding: UTF-8
OS name: "mac os x", version: "10.15.4", arch: "x86_64", family: "mac"
```

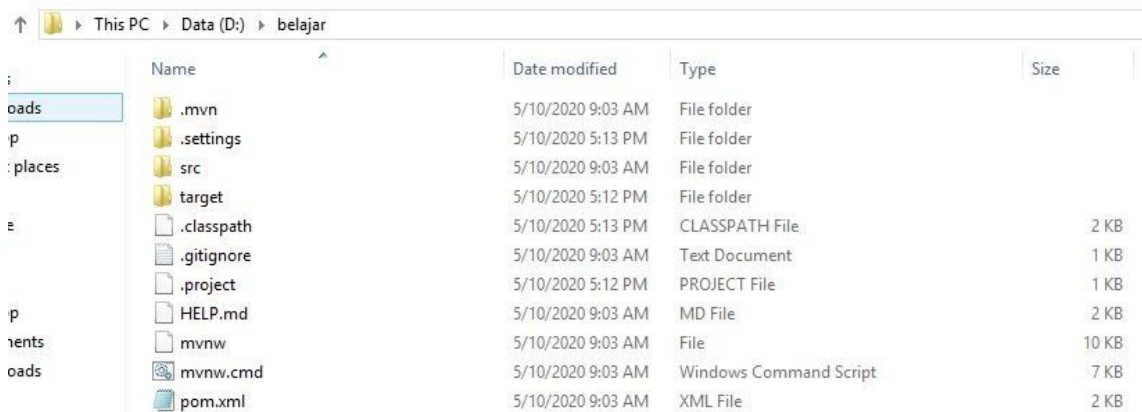
2. Membuat Project Spring:

- Buka web <https://start.spring.io/>
- Spring Boot: pilih 2.2.7 atau 2.2.6 (Pilih yang tidak ada tulisan SNAPSHOT)
- Project Metadata:
 - Group: com.ddp2
 - Artifact: belajar
- Dependencies:
 - Spring Boot Devtools
 - Spring Web
 - Thymeleaf
- Java: sesuaikan dengan yang kalian pakai
- Klik tombol "GENERATE"



The image shows the Spring Initializr web interface. It has a header with the 'spring initializr' logo. Below the logo, there are sections for 'Project', 'Language', 'Spring Boot', 'Project Metadata', and 'Dependencies'. The 'Project' section has radio buttons for 'Maven Project' (selected) and 'Gradle Project'. The 'Language' section has radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'. The 'Spring Boot' section has radio buttons for versions: '2.3.0 RC1', '2.3.0 (SNAPSHOT)', '2.2.8 (SNAPSHOT)', '2.2.7' (selected), '2.1.15 (SNAPSHOT)', and '2.1.14'. The 'Project Metadata' section has input fields for 'Group' (com.ddp2), 'Artifact' (belajar), 'Name' (belajar), 'Description' (Demo project for Spring Boot), and 'Package name' (com.ddp2.belajar). There are also radio buttons for 'Packaging' ('Jar' selected, 'War' unselected) and 'Java' ('14' unselected, '11' unselected, '8' selected). The 'Dependencies' section has a button 'ADD DEPENDENCIES... CTRL + B' and lists 'Spring Boot DevTools' (DEVELOPER TOOLS), 'Spring Web' (WEB), and 'Thymeleaf' (TEMPLATE ENGINES). At the bottom, there are buttons 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE', and 'SHARE...'.

- Setelah itu akan otomatis mendownload file: belajar.zip
- Pindahkan file belajar.zip ke folder D:\ dan extract file tersebut



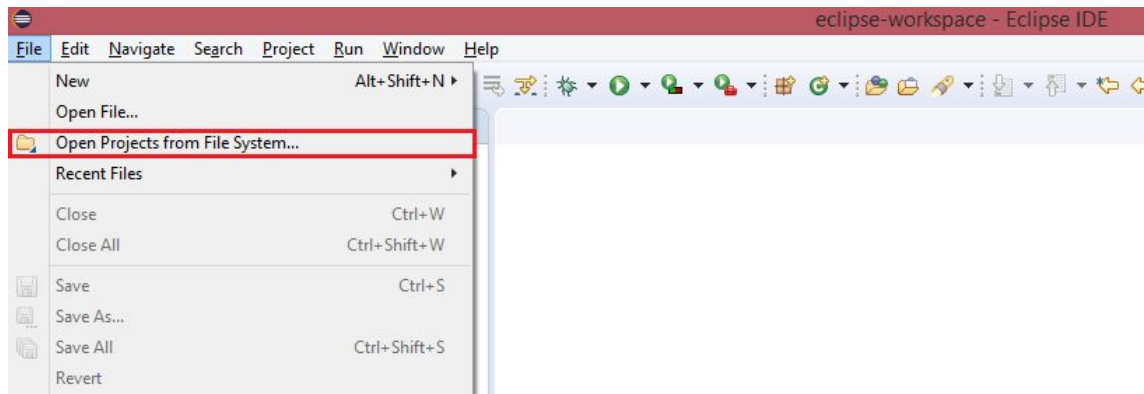
The image shows a Windows File Explorer window with the address bar set to 'This PC > Data (D:) > belajar'. The left sidebar shows the 'belajar' folder selected. The main pane displays a list of files and folders with columns for Name, Date modified, Type, and Size.

Name	Date modified	Type	Size
.mvn	5/10/2020 9:03 AM	File folder	
.settings	5/10/2020 5:13 PM	File folder	
src	5/10/2020 9:03 AM	File folder	
target	5/10/2020 5:12 PM	File folder	
.classpath	5/10/2020 5:13 PM	CLASSPATH File	2 KB
.gitignore	5/10/2020 9:03 AM	Text Document	1 KB
.project	5/10/2020 5:12 PM	PROJECT File	1 KB
HELP.md	5/10/2020 9:03 AM	MD File	2 KB
mvnw	5/10/2020 9:03 AM	File	10 KB
mvnw.cmd	5/10/2020 9:03 AM	Windows Command Script	7 KB
pom.xml	5/10/2020 9:03 AM	XML File	2 KB

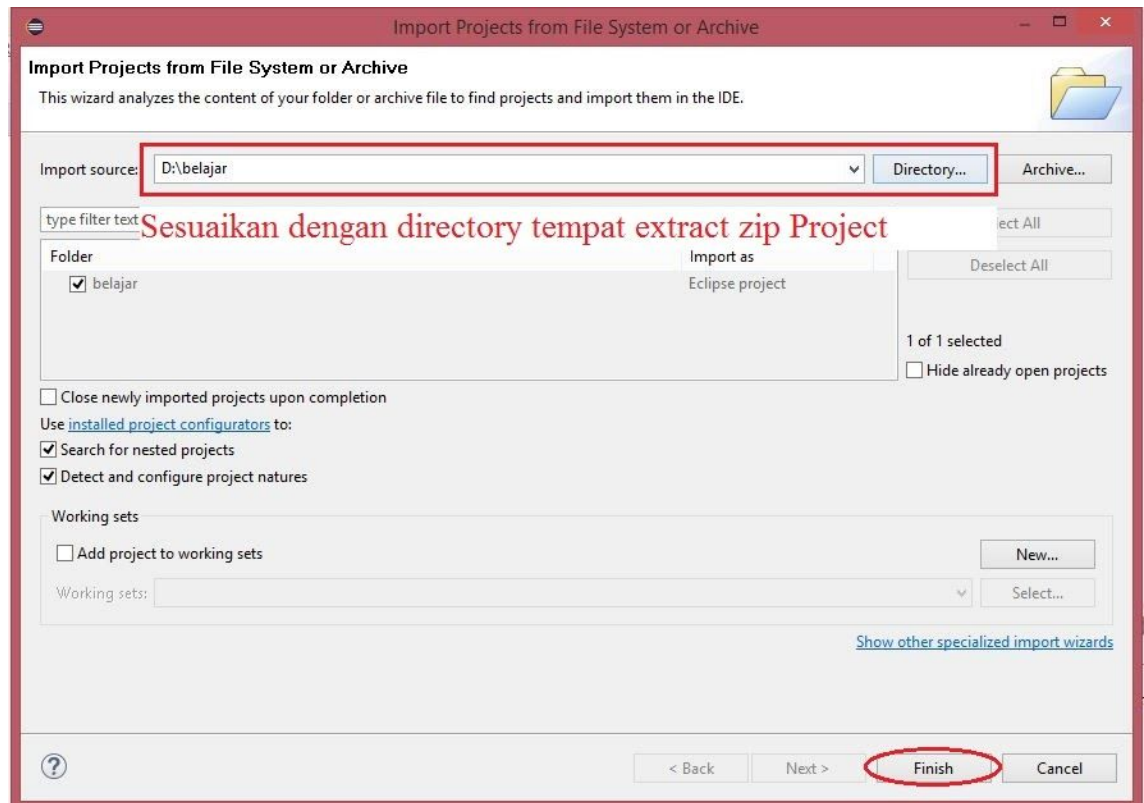
Note: Group, Artifact, dan directory tempat extract file adalah untuk keperluan tutorial ini. Silakan mengubah nama dan directory sesuai kebutuhan.

3. Membuka Spring Project Melalui Eclipse:

- Jalankan Eclipse
- Buka Menu: File -> Open Project from File System



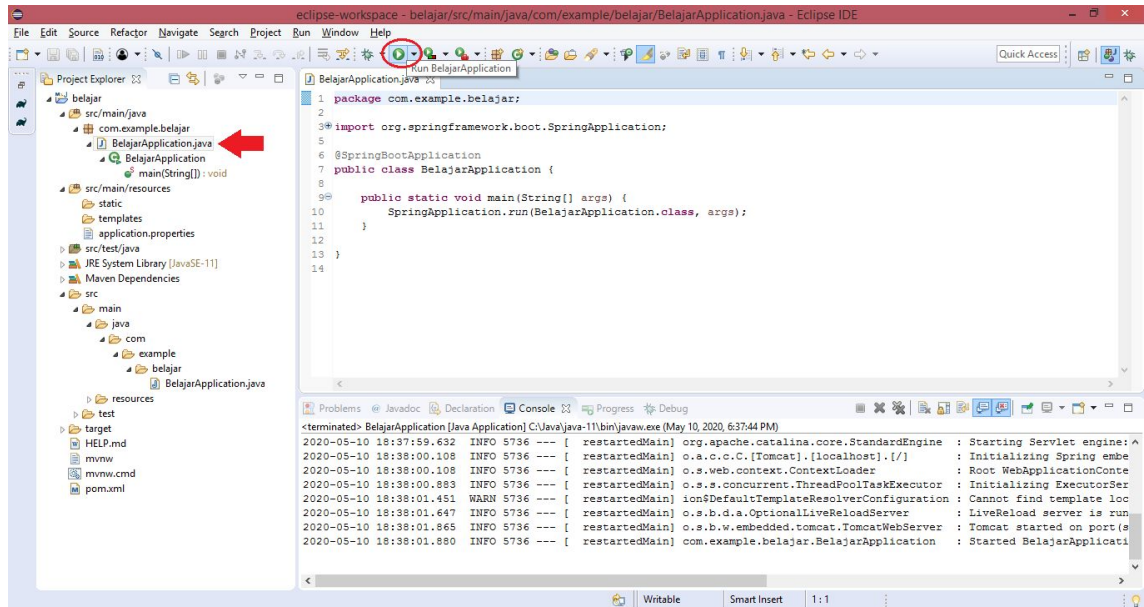
- Buka directory D:\belajar (Disesuaikan dengan folder tempat extract file zip pada langkah sebelumnya)
- Tekan "Finish"



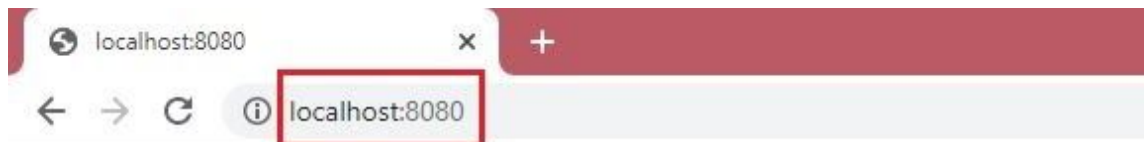
- Eclipse akan mengkonfigurasi Project dengan Maven (Mungkin akan memerlukan waktu yang cukup lama saat pertama kali menjalankan Project)

4. Menjalankan Spring Boot Application:

- Pada directory src/main/java/com/example/belajar, buka file BelajarApplication.java (Application adalah program yang digunakan untuk menjalankan sebuah Spring Application yang ditandai dengan anotasi @SpringBootApplication)
- Klik Tombol Run



- Setelah Project Spring sudah berhasil dijalankan, tampilan web dapat dilihat pada link <http://localhost:8080/> (Seharusnya masih menampilkan Whitelabel Error Page karena belum membuat file Controller yang akan dibahas pada bagian berikutnya).



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sun May 10 19:34:31 SGT 2020

There was an unexpected error (type=Not Found, status=404).

No message available

- Aplikasi Spring sudah dapat digunakan

Membuat Spring Boot Project Menggunakan VS Code

1. Install Maven/Gradle seperti cara di atas (opsional).

Kamu bisa menggunakan Maven atau Gradle (silakan pilih salah satu ketika membuat *project* Spring nanti) untuk mengatur proyek Spring Boot.

Maven/Gradle **tidak wajib** diinstal karena *extension* pada VS Code sudah otomatis menggunakan *wrapper* (versi portabel yang tidak perlu diinstal) yang disediakan pada *project* yang di-generate.

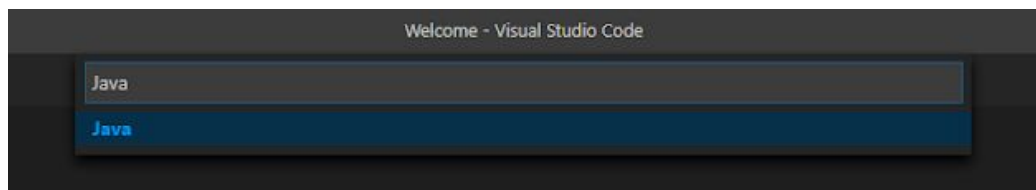
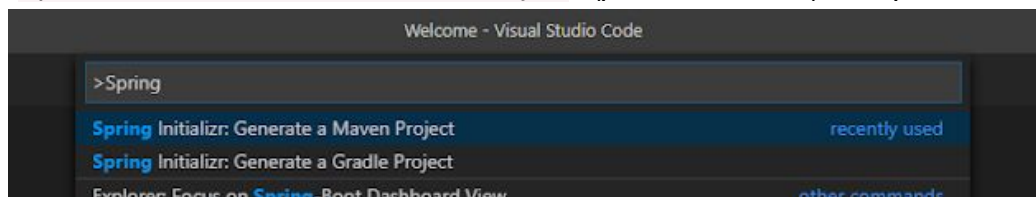
Namun, **jika kamu mau**, kamu bisa menginstal Maven/Gradle ke komputer kamu.

2. Install Extensions berikut:

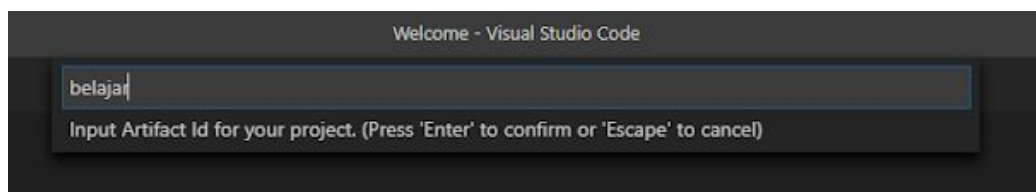
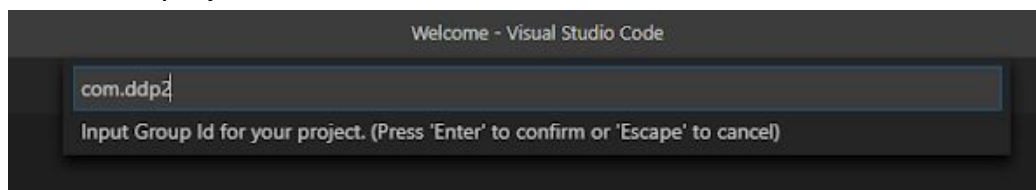
- Maven for Java/Gradle Extension Pack (pilih salah satu)
- Spring Initializr Java Support
- Spring Boot Tools
- Spring Boot Dashboard

3. Membuat Project Spring

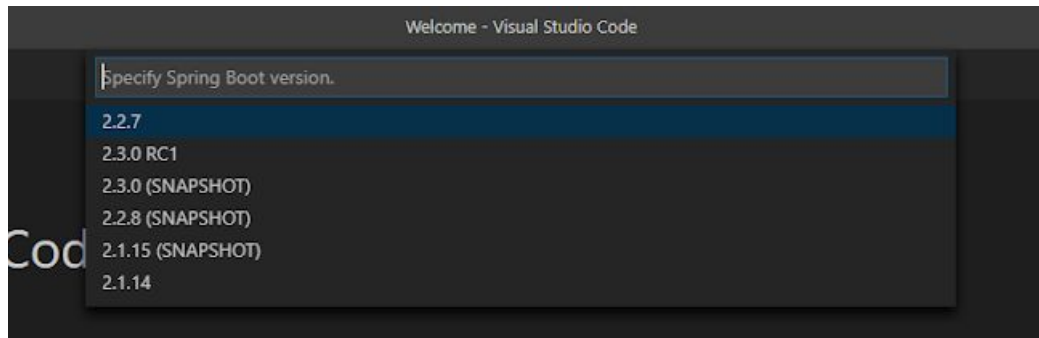
- Buka Command Palette (Ctrl+Shift+P atau Cmd+Shift+P pada Mac), masukkan command “Spring Initializr: Generate a Maven Project” atau “Spring Initializr: Generate a Gradle Project” (pilih salah satu) dan pilih Java.



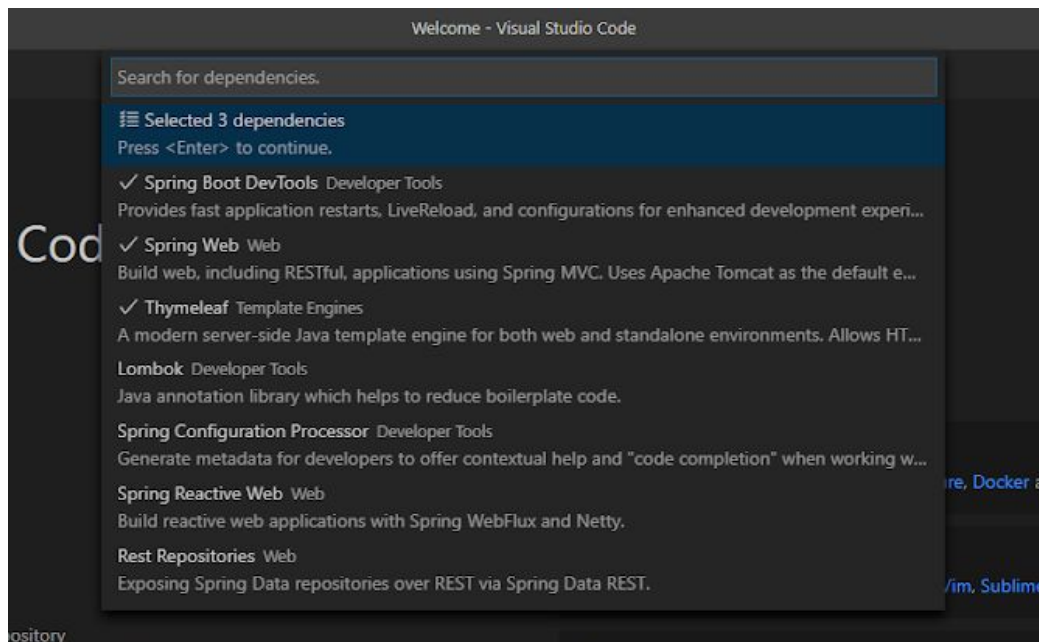
- Masukkan project metadata.



- Pilih Spring Boot 2.2.7



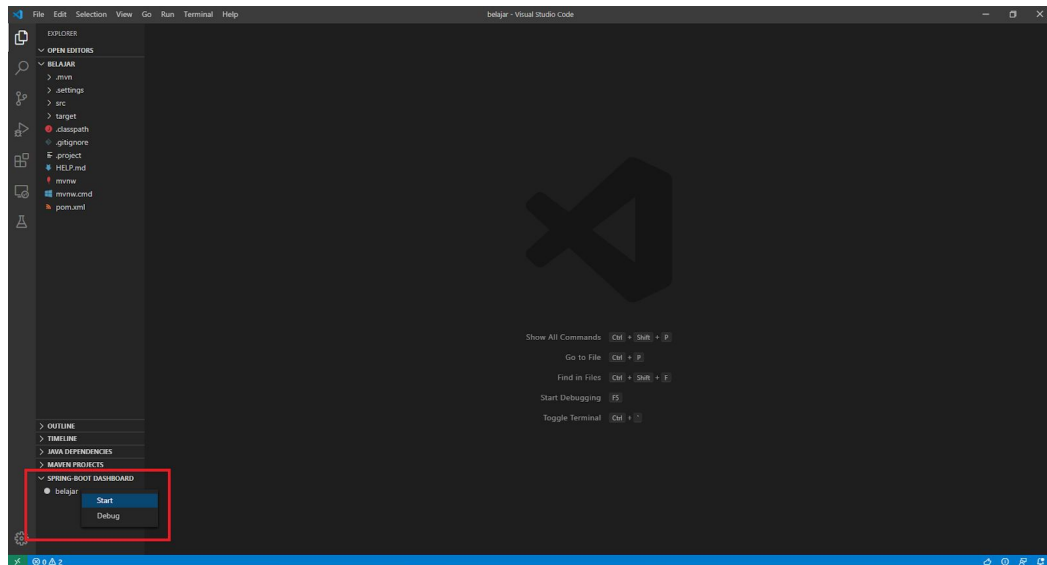
- Pilih dependencies yang dibutuhkan (Spring Boot DevTools, Spring Web, dan Thymeleaf)



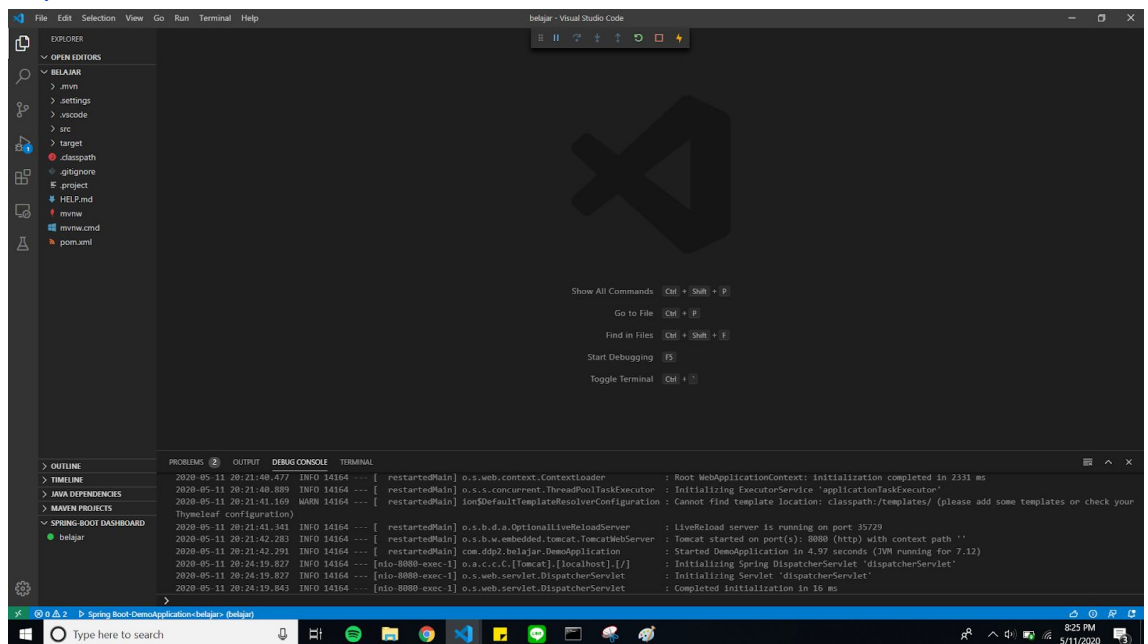
- Tekan “Enter” dan pilih folder tempat untuk menyimpan project tersebut

4. Menjalankan Spring Boot Application

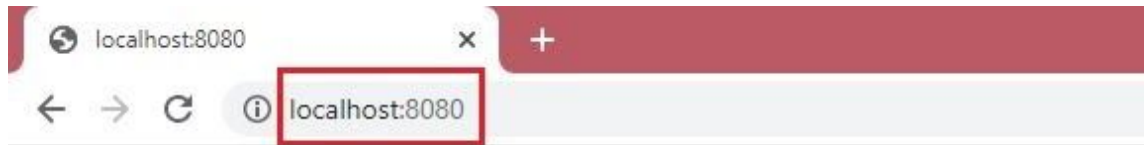
- Buka folder “belajar” yang telah di-*generate* tadi dengan VS Code.
- Pada Explorer, terdapat bagian Spring-Boot Dashboard, klik kanan “belajar” dan pilih Start.



- Jika Project Spring berhasil dijalankan, maka dapat diakses pada <http://localhost:8080/>.



- Setelah Project Spring sudah berhasil dijalankan, tampilan web dapat dilihat pada link <http://localhost:8080/> (Seharusnya masih menampilkan Whitelabel Error Page karena belum membuat file Controller yang akan dibahas pada bagian berikutnya).



Whitelabel Error Page

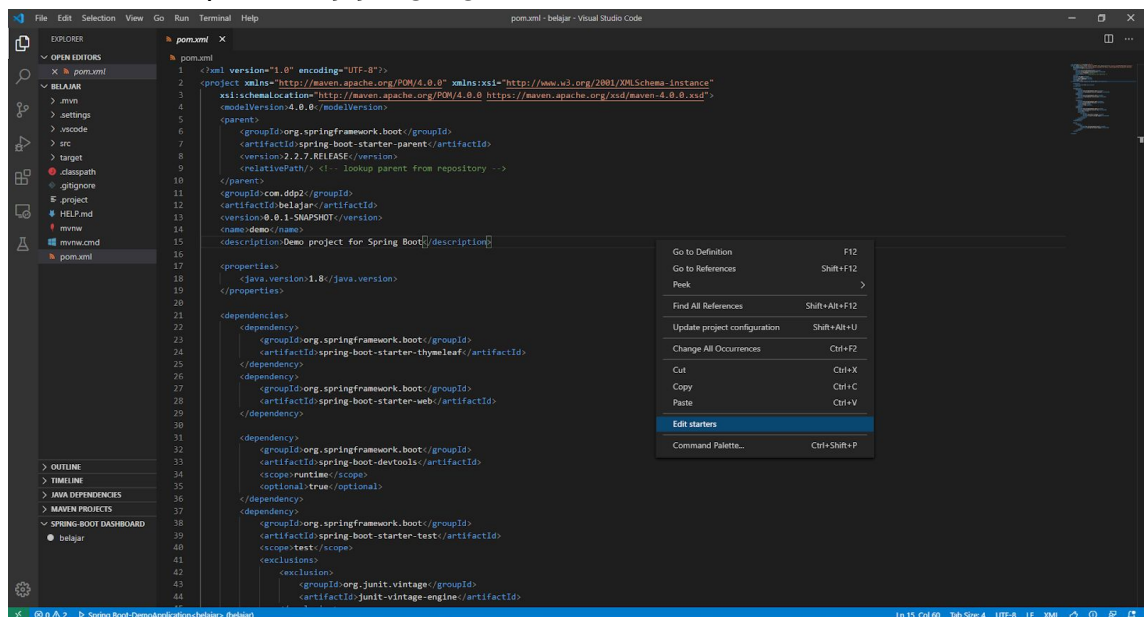
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sun May 10 19:34:31 SGT 2020

There was an unexpected error (type=Not Found, status=404).

No message available

- Aplikasi Spring sudah dapat digunakan
- 5. Edit Dependencies Project yang sudah dibuat (jika perlu)**
- Buka file pom.xml (Maven) atau build.gradle (Gradle)
 - Maven: Klik kanan, pilih “Edit starters”
 - Pilih dependency yang ingin ditambahkan



- Gradle: masukkan *dependency* yang ingin ditambahkan ke bagian **dependencies**.

Referensi:

<https://code.visualstudio.com/docs/java/java-spring-boot>

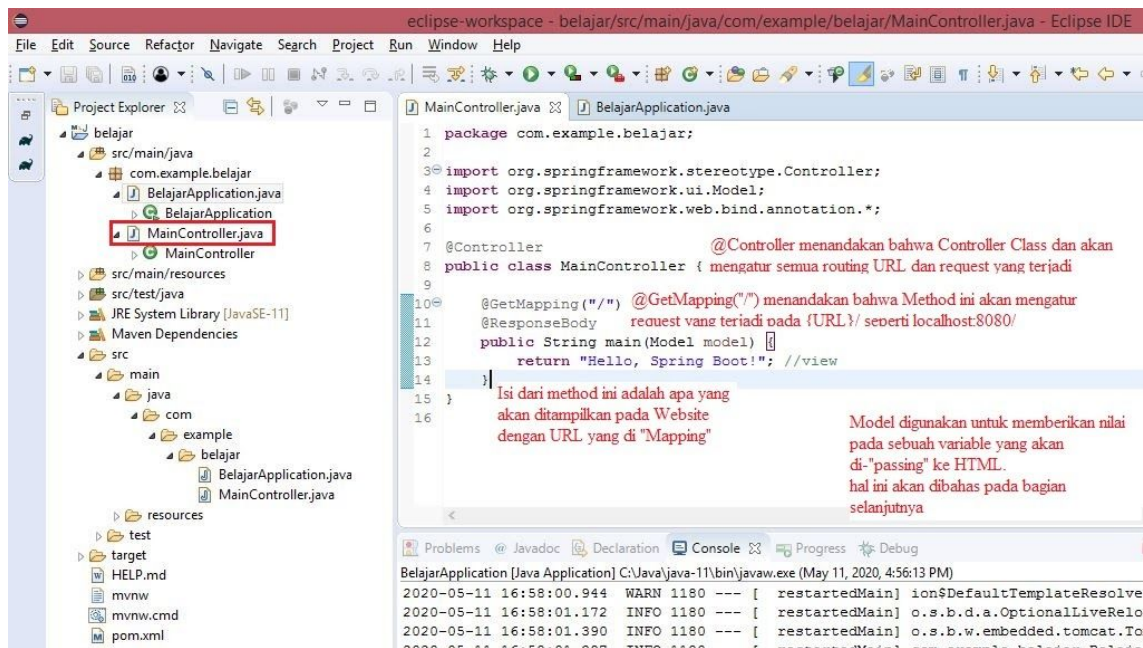
Materi

Note: Contoh yang ditampilkan pada bagian materi ini akan menggunakan Spring Project “belajar” yang sudah dibuat pada bagian sebelumnya.

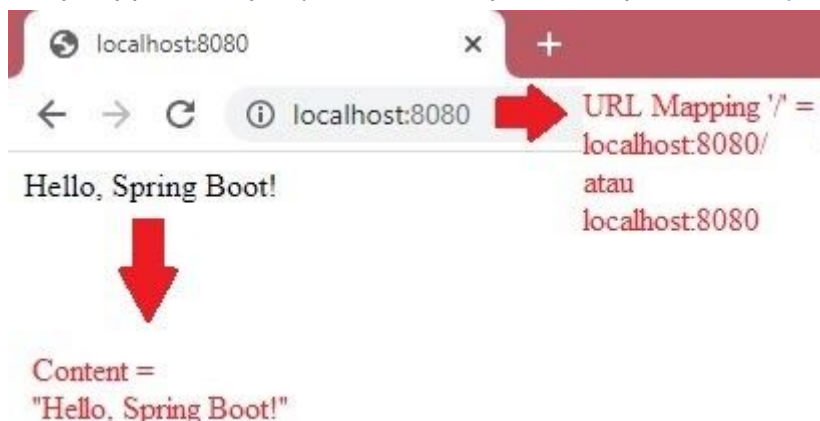
1. Controller in Spring

Controller digunakan untuk membuat routing URL pada suatu aplikasi Spring Boot dan bertugas untuk memproses request yang terjadi pada suatu URL.

Silakan coba membuat sebuah file `MainController.java` pada directory `src/main/java/com/example/belajar` sebagai berikut:



Setelah itu, run Spring Boot Application kalian (Dengan cara run pada file `BelajarApplication.java`) dan lihat tampilan web pada link <http://localhost:8080/>



2. Introducing Thymeleaf

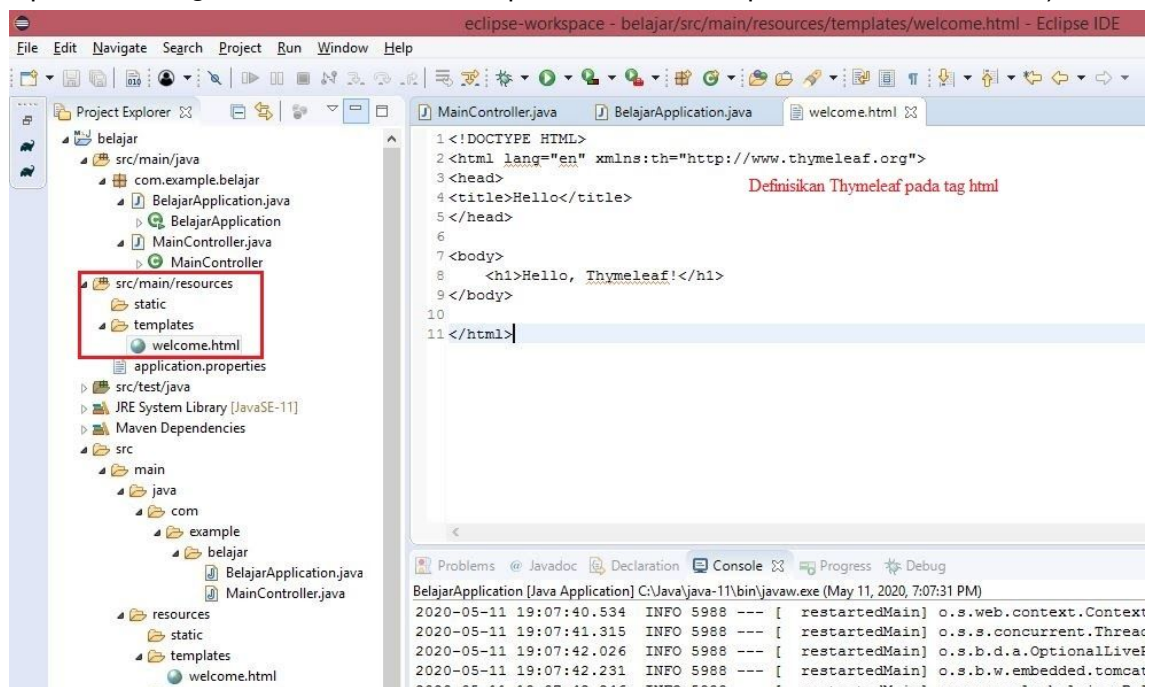
Thymeleaf merupakan Server Side Java Template Engine yang diperuntukan untuk membangun sebuah Aplikasi Web. Thymeleaf merupakan modern template engine yang support dengan HTML 5 dan sudah terintegrasi dengan Framework Java yaitu Spring. Dokumentasi dari Thymeleaf sangat baik dan relatif mudah dipahami.

Thymeleaf Standard Expressions:

- `${...}` : Variable expressions
- `*{...}` : Selection expressions
- `#{...}` : Message (i18n) expressions
- `@{...}` : Link/URL expressions
- `~{...}` : Fragment expressions

Untuk menggunakan thymeleaf, kita perlu mendefinisikannya di file HTML kita.

Sebagai contoh, coba kita buat file `welcome.html` di dalam `src/main/resources/templates` (Untuk edit file html pada Eclipse, kemungkinan diperlukan langkah berikut: klik kanan pada file html -> open with -> Text Editor):



Setelah itu, tambahkan tambahkan kode berikut ini pada MainController.java:

```
1 package com.example.belajar;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.ui.Model;
5 import org.springframework.web.bind.annotation.*;
6
7 @Controller
8 public class MainController {
9
10     @GetMapping("/")
11     @ResponseBody
12     public String main(Model model) {
13         return "Hello, Spring Boot!"; //view
14     }
15     URL Mapping = "/th"
16     @GetMapping("/th")
17     public String th(Model model) {
18         return "welcome";
19     }
20 }
```

Ketika tidak menggunakan anotasi `@ResponseBody`, maka akan dilakukan render file `welcome.html` pada template

Perhatikan bahwa kita tidak menggunakan anotasi `@ResponseBody` karena kita ingin melakukan render terhadap file `welcome.html` yang berada pada templates. Setelah itu, jalankan (Run) Spring Boot Application dan lihat tampilan pada web URL <http://localhost:8080/th>



Hello, Thymeleaf!  Isi dari file `welcome.html` pada template

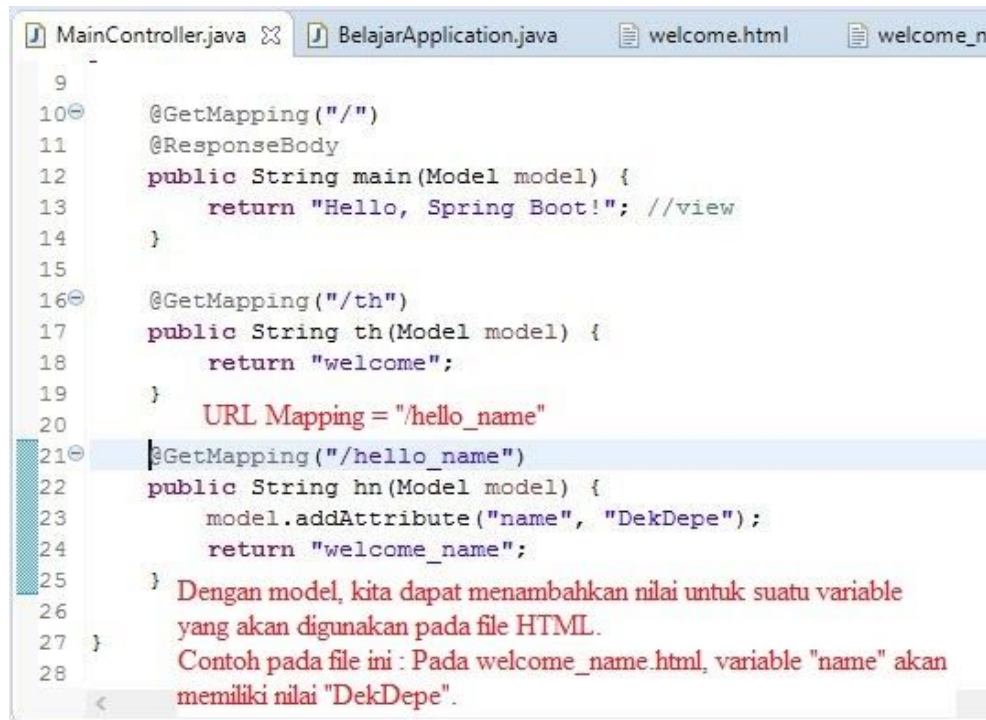
3. Content Passing from Java Code to Thymeleaf

Kita dapat melakukan passing content/parameter dari Java code ke Spring Boot website dengan menggunakan Model Object dan dengan bantuan Thymeleaf.

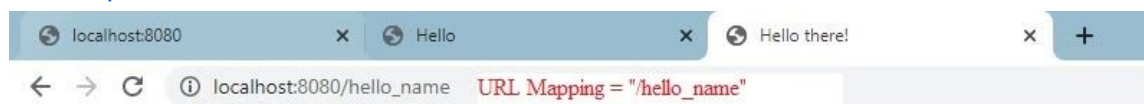
Sebagai latihan, silakan membuat file `welcome_name.html` di dalam folder `src/main/resources/templates` yang isinya sebagai berikut:



Selanjutnya, tambahkan kode berikut pada `MainController.java`:



Setelah itu, jalankan (Run) Spring Boot Application dan lihat tampilan pada web URL http://localhost:8080/hello_name



Hello, DekDepe!

"DekDepe" merupakan value dari variable "name" yang nilainya telah di-set menggunakan model pada `MainController.java`

Kita juga dapat melakukan passing ArrayList ke file HTML, seperti contoh berikut:
Buat File welcome_list.html di dalam src/main/resources/templates yang isinya:

```
MainController.java BelajarApplication.java welcome.html welcome_name.html welcome_list.html
1 <!DOCTYPE HTML>
2 <html lang="en" xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <title>Hello there!</title>
5 </head>
6
7 <body>
8   <h1 th:text="'Hello, ' + ${name} + '!'"></h1>
9   <h2>Favorite Coffee Beans</h2>
10  <ol>
11    <li th:each="coffeeBean : ${coffeeBeans}" th:text="${coffeeBean}"></li>
12  </ol>
13 </body>
14
15 </html>
16
```

Akan dibuat {coffeeBean} di mana coffeeBean adalah isi dari Variable coffeeBeans pada model di MainController.java

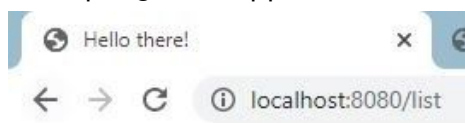
Selanjutnya, tambahkan kode berikut pada MainController.java:

```
MainController.java BelajarApplication.java welcome.html welco
24 @GetMapping("/hello_name")
25 public String hn(Model model) {
26   model.addAttribute("name", "DekDepe");
27   return "welcome_name";
28 }
29
30 @GetMapping("/list")
31 public String hl(Model model) {
32   List<String> coffeeBeans = new ArrayList<>();
33   coffeeBeans.add("Arabica");
34   coffeeBeans.add("Robusta");
35
36   model.addAttribute("name", "DekDepe");
37   model.addAttribute("coffeeBeans", coffeeBeans);
38   return "welcome_list";
39 }
40
41
42 }
43
```

Jangan lupa lakukan import java.util.ArrayList dan java.util.List

Model mendefinisikan nilai Variable coffeeBeans dengan sebuah ArrayList of String yang isinya ["Arabica", "Robusta"]

Run Spring Boot Application dan lihat tampilan pada URL <http://localhost:8080/list>



Hello, DekDepe!

Favorite Coffee Beans

1. Arabica
2. Robusta

Dibuat 2 buah yang merupakan tag list pada html yang berisi nilai-nilai pada coffeeBeans

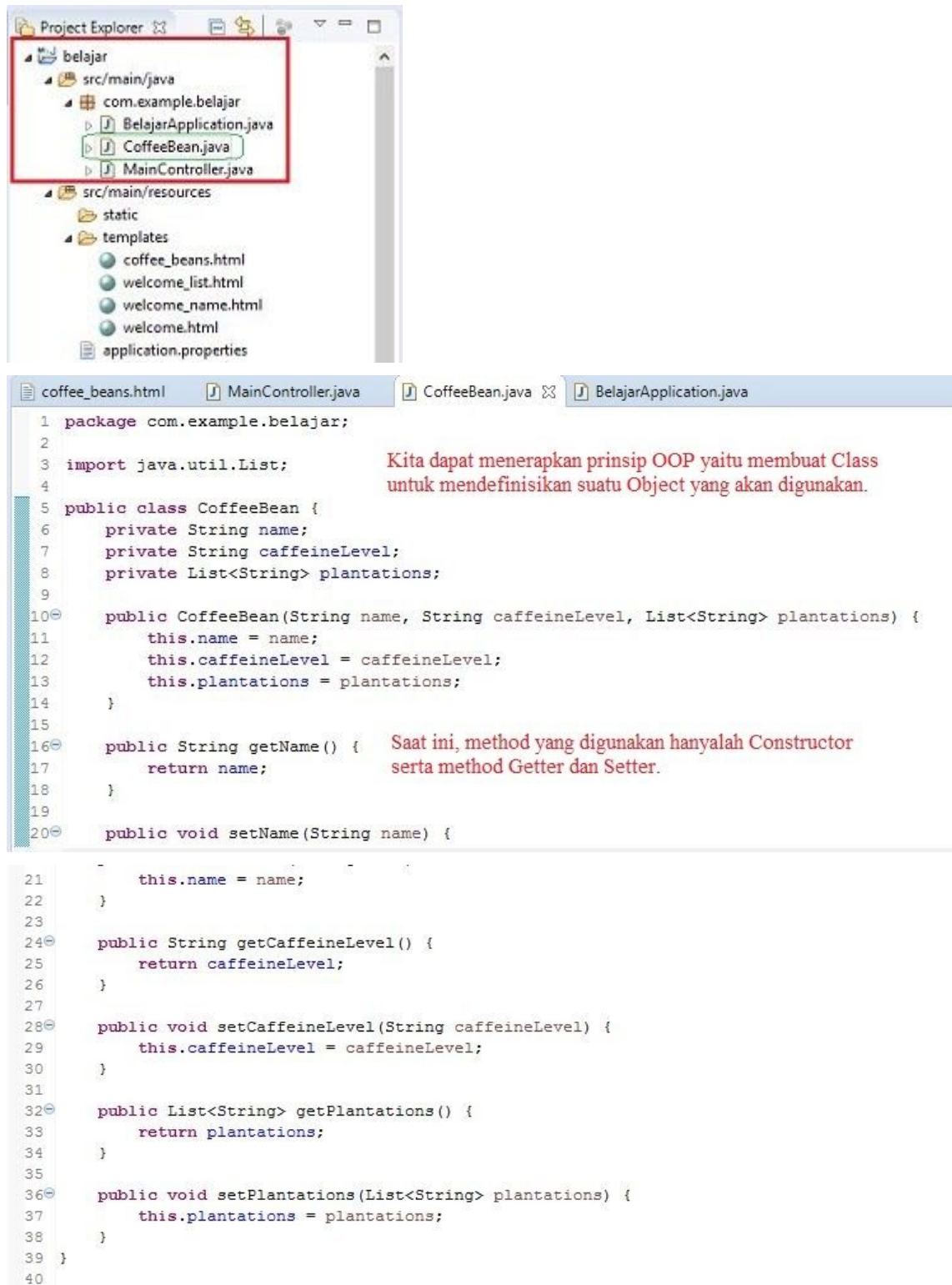
Selain pendefinisian variabel secara langsung pada MainController, kita juga dapat mendefinisikan variabel pada file java yang terpisah dan menggunakannya di MainController.java. Berikut adalah contohnya:

Buat File coffee_beans.html di dalam src/main/resources/templates yang isinya:

```
coffee_beans.html MainController.java CoffeeBean.java BelajarApplication.java
1<!DOCTYPE HTML>
2<html lang="en" xmlns:th="http://www.thymeleaf.org">
3<head>
4  <title>Coffee Beans Table</title>
5  <style>
6    td {
7      text-align: center;
8    }
9  </style>
10</head>
11
12<body>
13  <h2>Coffee Beans Table</h2>
14  <table border = "1">
15    <thead>
16      <tr>
17        <th> Name </th>
18        <th> Caffeine Level </th>
19        <th> Plantations </th>
20      </tr>
21    </thead>
22    <tbody>
23      <tr th:if="${coffeebeans.empty}">
24        <td colspan="3"> No Coffee Beans Available </td>
25      </tr>
26      <tr th:each="bean : ${coffeebeans}">
27        <td><span th:text="${bean.name}"></span></td>
28        <td><span th:text="${bean.caffeineLevel}"></span></td>
29        <td><span th:text="${#strings.listJoin(bean.plantations, ', ')}"></span></td>
30      </tr>
31    </tbody>
32  </table>
33</body>
34</html>
35
```

Nilai dari variable tersebut akan selalu didapat menggunakan model pada MainController.java.

Buat File CoffeeBean.java pada src/main/java/com/example/belajar yang isinya:



The screenshot shows an IDE with the Project Explorer on the left and the CoffeeBean.java file open in the editor. The Project Explorer shows the project structure with the following folders and files:

- belajar
 - src/main/java
 - com.example.belajar
 - BelajarApplication.java
 - CoffeeBean.java
 - MainController.java
 - src/main/resources
 - static
 - templates
 - coffee_beans.html
 - welcome_list.html
 - welcome_name.html
 - welcome.html
 - application.properties

The editor shows the following code for CoffeeBean.java:

```
1 package com.example.belajar;
2
3 import java.util.List;
4
5 public class CoffeeBean {
6     private String name;
7     private String caffeineLevel;
8     private List<String> plantations;
9
10    public CoffeeBean(String name, String caffeineLevel, List<String> plantations) {
11        this.name = name;
12        this.caffeineLevel = caffeineLevel;
13        this.plantations = plantations;
14    }
15
16    public String getName() {
17        return name;
18    }
19
20    public void setName(String name) {
21
22        this.name = name;
23    }
24
25    public String getCaffeineLevel() {
26        return caffeineLevel;
27    }
28
29    public void setCaffeineLevel(String caffeineLevel) {
30        this.caffeineLevel = caffeineLevel;
31    }
32
33    public List<String> getPlantations() {
34        return plantations;
35    }
36
37    public void setPlantations(List<String> plantations) {
38        this.plantations = plantations;
39    }
40 }
```

Kita dapat menerapkan prinsip OOP yaitu membuat Class untuk mendefinisikan suatu Object yang akan digunakan.

Saat ini, method yang digunakan hanyalah Constructor serta method Getter dan Setter.

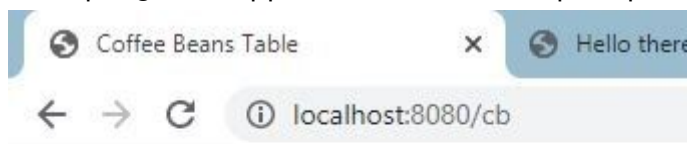
Selanjutnya, tambahkan kode berikut pada MainController.java:

```
coffee_beans.html | MainController.java | CoffeeBean.java | BelajarApplication.java
41 @GetMapping("/cb")
42 public String cb(Model model) {
43     List<CoffeeBean> coffeeBeans = new ArrayList<>();
44     // arabica
45     List<String> arabicaPlantations = new ArrayList<>();
46     arabicaPlantations.add("Latin America");
47     arabicaPlantations.add("Africa");
48     CoffeeBean arabica = new CoffeeBean("Arabica", "Medium", arabicaPlantations);
49     // robusta
50     List<String> robustaPlantations = new ArrayList<>();
51     robustaPlantations.add("Africa");
52     robustaPlantations.add("Java");
53     CoffeeBean robusta = new CoffeeBean("Robusta", "High", robustaPlantations);
54     coffeeBeans.add(arabica);
55     coffeeBeans.add(robusta);
56     model.addAttribute("coffeebeans", coffeeBeans);
57     return "coffee_beans";
58 }
59
60 }
```

Kita membuat ArrayList of CoffeeBean untuk digunakan pada File HTML kita.

Untuk saat ini, kita hanya memanfaatkan Constructor dari CoffeeBean.

Run Spring Boot Application dan lihat tampilan pada URL <http://localhost:8080/cb>



Coffee Beans Table

Name	Caffeine Level	Plantations
Arabica	Medium	Latin America, Africa
Robusta	High	Africa, Java

4. Content Passing Using URL Parameter

Selain dengan pendefinisian nilai variabel yang diisi langsung pada file MainController.java, kita juga dapat mendefinisikan nilai suatu variabel dengan mengambil nilai yang tertera di parameter pada URL.

Sebagai latihan, silakan membuat file greeting.html di dalam src/main/resources/templates yang isinya:

```
MainController.java BelajarApplication.java greeting.html
1 <!DOCTYPE HTML>
2 <html lang="en" xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <title>Hello there!</title>
5 </head>
6
7 <body>
8   <h1 th:text="'Hello, ' + ${name} + '!'"></h1>
9 </body>
10
11 </html>
```

Seperti latihan sebelumnya, nilai dari name didapat melalui model pada MainController.java

Setelah itu, tambahkan kode berikut pada MainController.java:

```
60 @GetMapping("/greeting")
61 public String helloParam(@RequestParam(defaultValue = "Stranger") String name, Model model) {
62     model.addAttribute("name", name);
63     return "greeting";
64 }
65
66 }
67
```

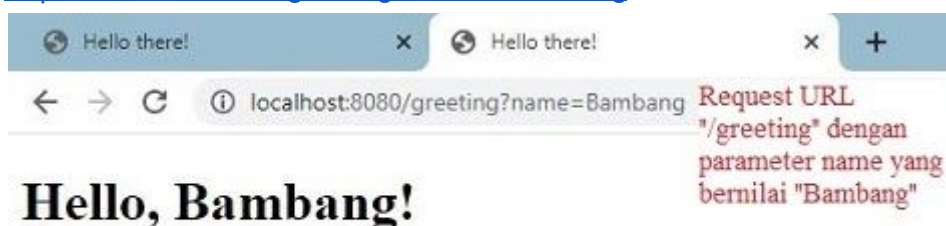
Pada bagian ini, kita meminta nilai untuk variabel name melalui parameter di URL kita. Apabila nilai dari name tidak didefinisikan pada parameter di URL, maka akan digunakan default value "Stranger" sebagai nilai dari variabel name.

Cara untuk memberikan nilai melalui parameter URL akan dibahas di bawah ini

Run Spring Boot Application dan lihat tampilan pada URL <http://localhost:8080/greeting>



Run Spring Boot Application dan lihat tampilan pada URL <http://localhost:8080/greeting?name=Bambang>



5. Form Submission Using Spring Boot

Kita juga bisa menambahkan form pada Project Spring kita. Form berguna untuk meminta nilai dari user dan menggunakannya untuk keperluan lain.

Berikut adalah latihan untuk membuat Form pada Spring Boot

Buat File coffee.html di dalam src/main/resources/templates yang isinya:

```
MainController.java  result.html  coffee.html  Coffee.java
1 <!DOCTYPE HTML>
2 <html xmlns:th="https://www.thymeleaf.org">
3 <head>
4   <title>Handling Form Submission</title>
5   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6 </head>
7 <body>
8   <h1>Form Create New Coffee</h1>
9   <form action="#" th:action="@{/createcoffee}" th:object="${coffee}" method="post">
10    <p>Coffee Name: <input type="text" th:field="*{name}" /></p>
11    <p>Description: <input type="text" th:field="*{description}" /></p>
12    <p><input type="submit" value="Submit Form" /> <input type="reset" value="Reset Form" /></p>
13  </form>
14 </body>
15 </html>
16
17
```

Membuat Form yang hasilnya akan disimpan pada variabel coffee pada Model

Isi field pertama disimpan sebagai atribut name, isi field kedua disimpan sebagai atribut description. Terdapat tombol submit yang bertuliskan "Submit Form" untuk melakukan proses POST yang akan diatur pada controller. Selain itu, juga terdapat tombol reset yang bertuliskan "Reset Form" untuk mereset seluruh field pada Form.

Buat File result.html di dalam src/main/resources/templates yang isinya:

```
MainController.java  result.html  coffee.html  Coffee.java
1 <!DOCTYPE HTML>
2 <html xmlns:th="https://www.thymeleaf.org">
3 <head>
4   <title>Result - Handling Form Submission</title>
5   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6 </head>
7 <body>
8   <h1>My Coffee:</h1>
9   <p th:text="'Coffee Name: ' + ${coffee.name}" />
10  <p th:text="'Description: ' + ${coffee.description}" />
11  <a href="/createcoffee">Create another Coffee</a>
12 </body>
13 </html>
14
15
```

Kita menggunakan variabel coffee yang didapat setelah melakukan method POST melalui Form.

Buat File Coffee.java di dalam src/main/java/com/example/belajar yang isinya:

```
1 package com.example.belajar;
2
3 public class Coffee {
4     private String name;
5     private String description;
6
7     public String getName() {
8         return name;
9     }
10    public void setName(String name) {
11        this.name = name;
12    }
13    public String getDescription() {
14        return description;
15    }
16    public void setDescription(String description) {
17        this.description = description;
18    }
19 }
20
```

Membuat Class Coffee yang didefinisikan atributnya dan membuat method Setter serta Getter untuk setiap atributnya

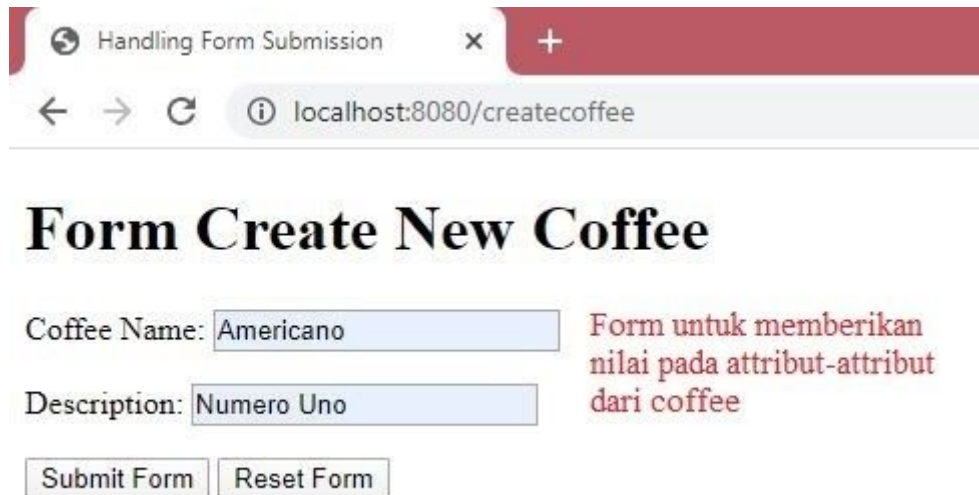
Setelah itu, tambahkan kode berikut pada MainController.java:

```
65
66    @GetMapping("/createcoffee")
67    public String coffeeForm(Model model) {
68        model.addAttribute("coffee", new Coffee());
69        return "coffee";
70    }
71
72    @PostMapping("/createcoffee")
73    public String coffeeSubmit(@ModelAttribute Coffee coffee) {
74        return "result";
75    }
76
```

Membuat variable baru yaitu coffee dengan Class Coffee yang akan diisi nilai dari atributnya melalui Form yang telah dibuat di coffee.html

Pada result.html, kita menggunakan variable coffee yang telah dikirimkan dengan metode POST saat tombol submit ditekan

Run Spring Boot Application, lalu lihat tampilan pada URL <http://localhost:8080/createcoffee> dan isi form tersebut dengan suatu nilai:



Handling Form Submission

localhost:8080/createcoffee

Form Create New Coffee

Coffee Name:

Description:

Form untuk memberikan nilai pada attribut-attribut dari coffee

Setelah itu, tekan tombol Submit Form dan lihat hasilnya:



Result - Handling Form Submission

localhost:8080/createcoffee

My Coffee:

Coffee Name: Americano

Description: Numero Uno

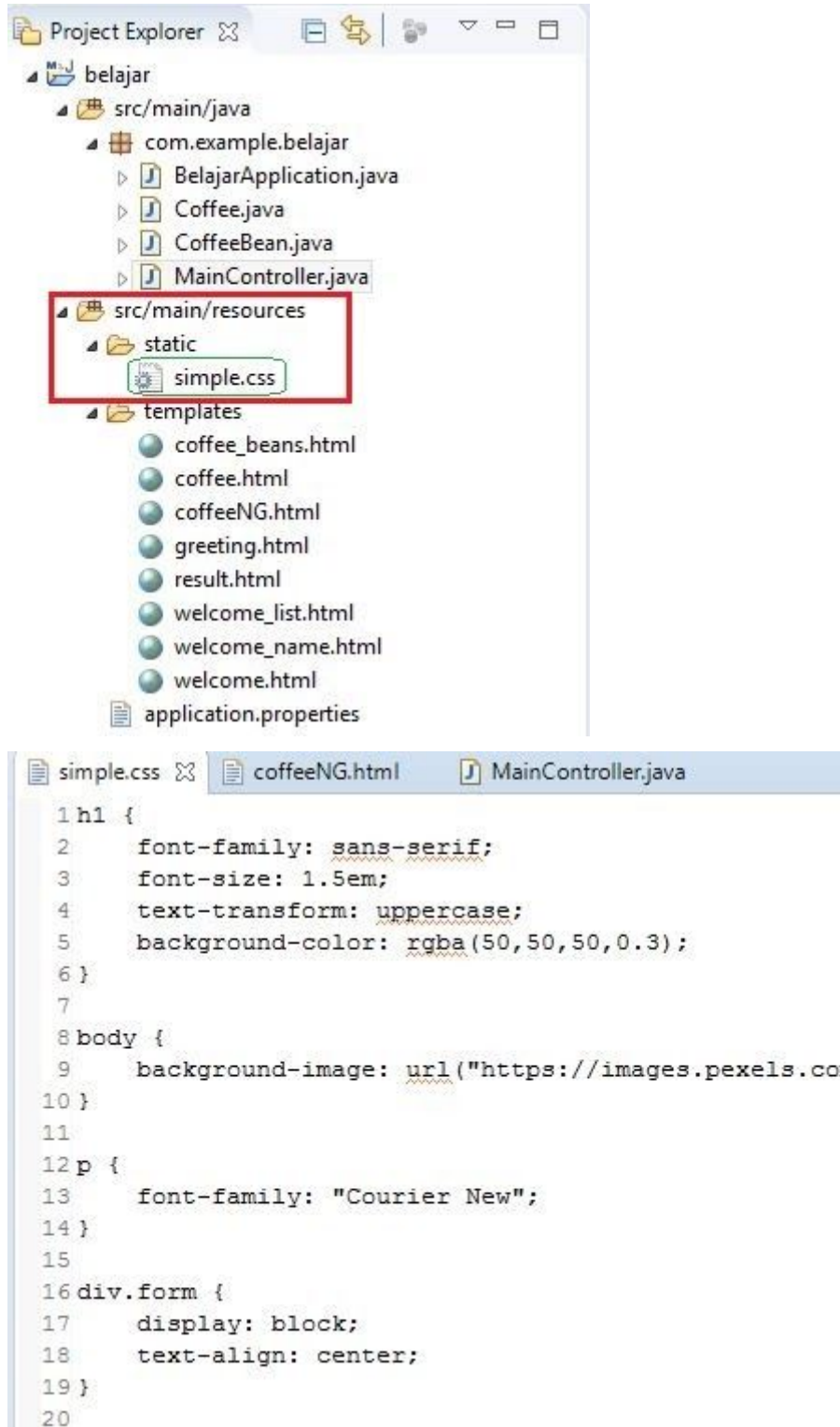
[Create another Coffee](#)

Setelah tombol submit ditekan, akan merender result.html yang berisikan variable coffee yang attributnya didapat dari isian form sebelumnya

6. Add CSS to Website

Kita bisa menambahkan File CSS untuk digunakan dalam website kita pada Spring Boot dengan cara sebagai berikut:

Buat simple.css di dalam src/main/resources/static yang isinya:




```

21 form {
22     display: inline-block;
23     margin-left: auto;
24     margin-right: auto;
25     text-align: left;
26 }
27
28 label {
29     display: inline-block;
30     float: left;
31     clear: left;
32     width: 125px;
33     text-align: left;
34 }
35
36 input {
37     display: inline-block;
38     float: left;
39 }
40

```

Link untuk background-image di atas:

<https://images.pexels.com/photos/6347/coffee-cup-working-happy.jpg?auto=compress&cs=tinysrgb&dpr=2&h=650&w=940>

Buat File coffeeNG.html di dalam src/main/resources/templates yang isinya:

```

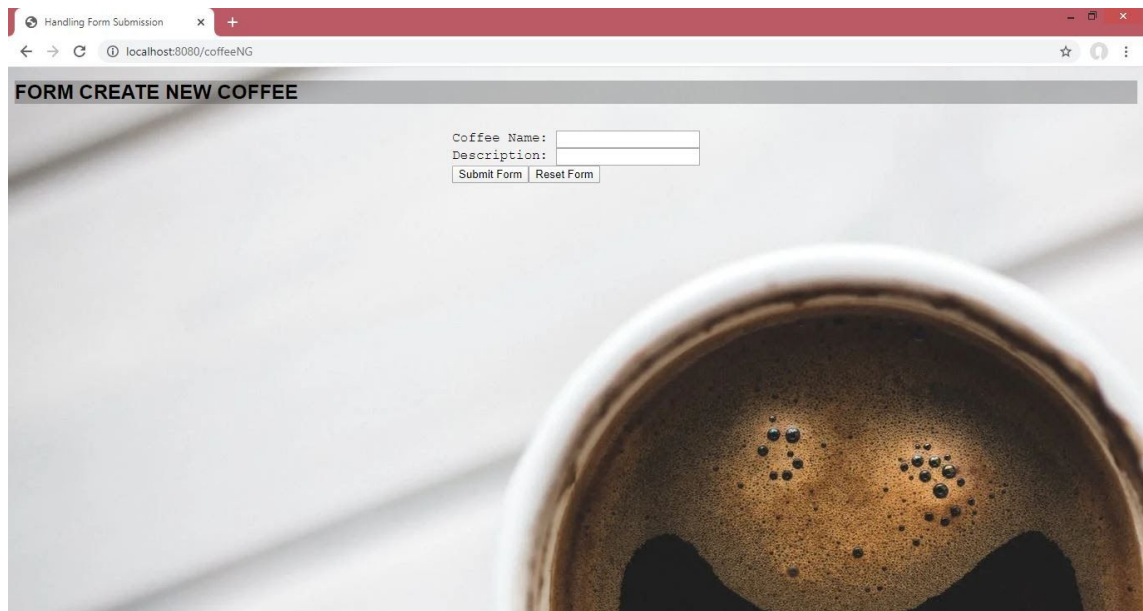
simple.css  coffeeNG.html  MainController.java
1 <!DOCTYPE HTML>
2 <html xmlns:th="https://www.thymeleaf.org">
3 <head>
4     <title>Handling Form Submission</title>
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6     <link th:href="@{/simple.css}" rel="stylesheet" />
7 </head>
8 <body>
9     <h1>Form Create New Coffee</h1>
10    <div class="form">
11        <form action="#" th:action="@{/createcoffee}" th:object="${coffee}" method="post">
12            <p><label>Coffee Name: </label><input type="text" th:field="*{name}" /></p>
13            <p><label>Description: </label><input type="text" th:field="*{description}" /></p>
14            <p><input type="submit" value="Submit Form" /> <input type="reset" value="Reset Form" /></p>
15        </form>
16    </div>
17 </body>
18 </html>
19
20

```

Setelah itu, tambahkan kode berikut pada MainController.java:

```
@GetMapping("/coffeeNG")
public String coffeeForm2(Model model) {
    model.addAttribute("coffee", new Coffee());
    return "coffeeNG";
}
```

Run Spring Boot Application, lihat tampilan di URL <http://localhost:8080/coffeeNG>



Referensi Tambahan

<https://code.visualstudio.com/docs/java/java-spring-boot>