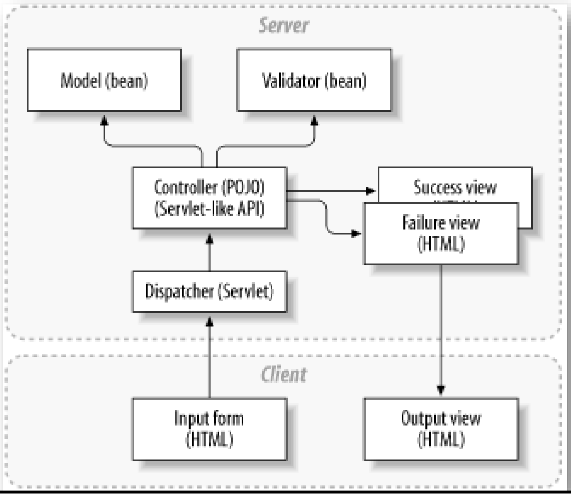
Spring MVC Web framework



1

1. **Spring Web MVC** **p 3**
   DispatcherServlet **p 4**

2. **Spring Boot** **p 6**

3. **First Example** **p 7**
   Creating a Spring Boot **p 8**
   pom.xml **p 10**
   SpringBootApplication **p 11**
   MVC
      Model **p 12**
      Controller **p 16**
      View **p 22**

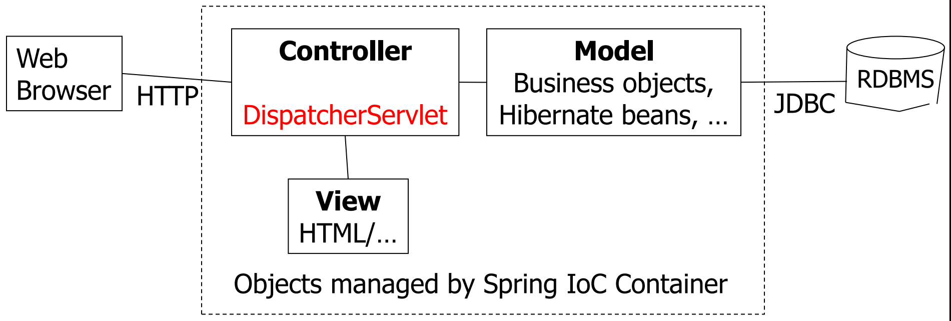
4. **Spring Boot Devtools / disable caching** **p 38**

5. **HomeController** **p 39**
6. **RequestMapping** **p 40**

2

1. Spring Web MVC

A part of the Spring Framework is **Spring Web MVC**,
an extensible MVC framework
for creating web applications.

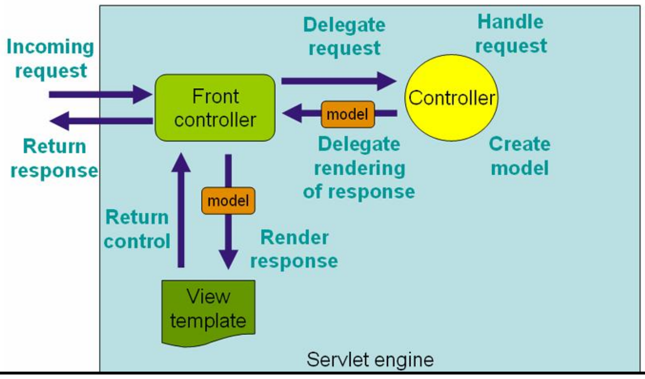

Web Browser
HTTP
Controller
DispatcherServlet
Model
Business objects, Hibernate beans, …
JDBC
RDBMS
View
HTML/…
Objects managed by Spring IoC Container

DispatcherServlet
is the entry point to the world of Spring Web MVC.

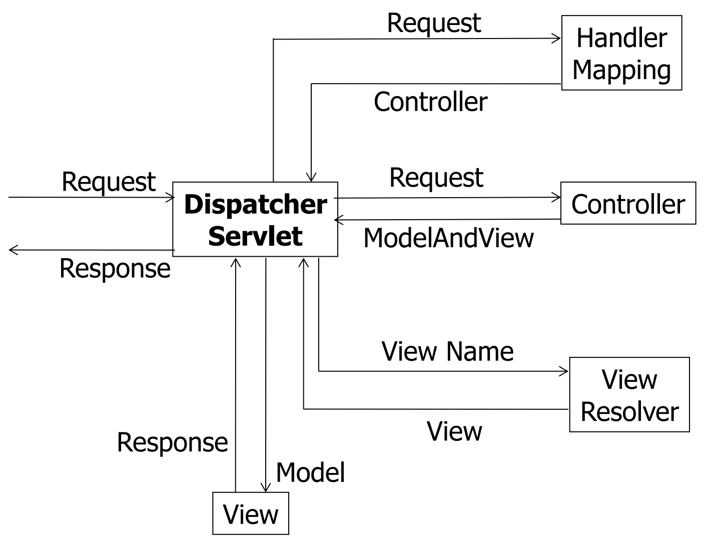
DispatcherServlet
is a **single front controller** servlet

= web application pattern

= a **single servlet** delegates responsibility for a request to other components of an application to perform actual processing.



Primary flow of request handling in Spring MVC



5

5

2. Spring Boot

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks, and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

http://spring.io/projects/spring-boot

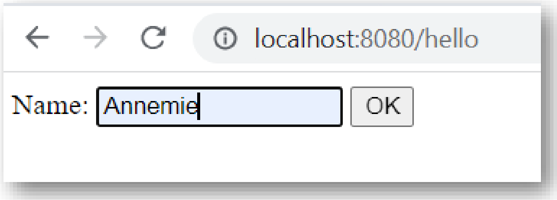
6

6

3. First Example

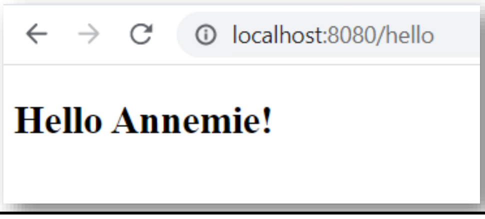
Example:

The application enables a user to enter her name in a text field, and upon clicking OK, the name is returned and displayed on a second page with a welcome greeting.



localhost:8080/hello

Name: Annemie



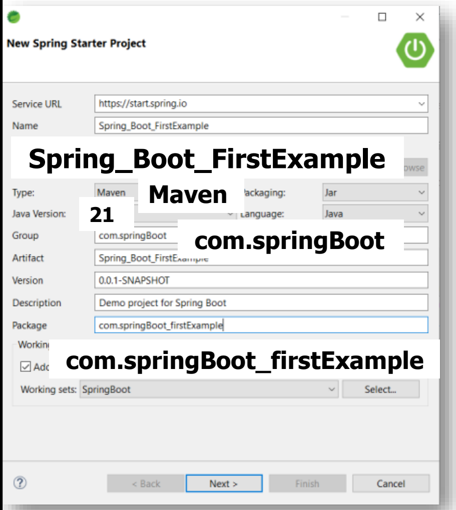
localhost:8080/hello

Hello Annemie!

7

Creating a Spring Boot

File -> New -> Spring Boot Starter



New Spring Starter Project

Service URL: https://start.spring.io
Name: Spring_Boot_FirstExample
Type: Maven
Packaging: Jar
Java Version:
Language: Java
Group: com.springBoot
Artifact: Spring_Boot_FirstExample
Version: 0.0.1-SNAPSHOT
Description: Demo project for Spring Boot
Package: com.springBoot_firstExample
Working sets: SpringBoot

Spring_Boot_FirstExample

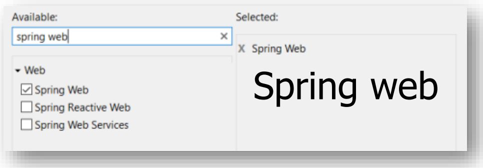
Maven

21

com.springBoot

com.springBoot_firstExample

Next

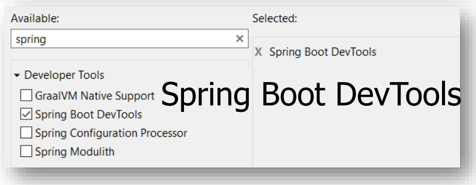


Available: spring web

Web
- ☑ Spring Web
- ☐ Spring Reactive Web
- ☐ Spring Web Services

Selected:
X Spring Web

Spring web

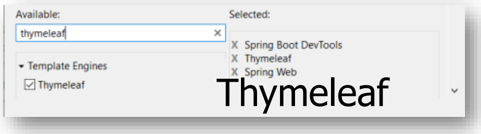


Available: spring

Developer Tools
- ☐ GraalVM Native Support
- ☑ Spring Boot DevTools
- ☐ Spring Configuration Processor
- ☐ Spring Modulith

Selected:
X Spring Boot DevTools

Spring Boot DevTools



Available: thymeleaf

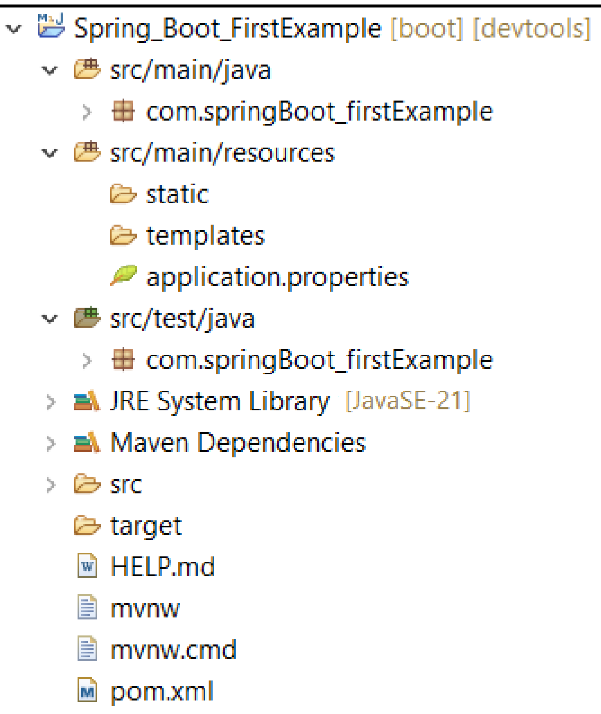
Template Engines
- ☑ Thymeleaf

Selected:
X Spring Boot DevTools
X Thymeleaf
X Spring Web

Thymeleaf

Finish

8



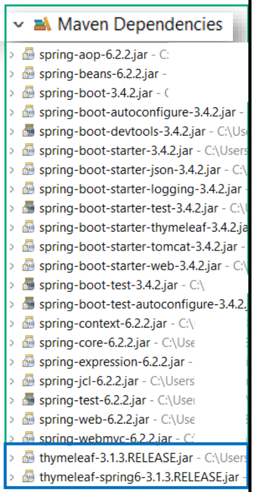
```
Spring_Boot_FirstExample [boot] [devtools]
  src/main/java
    com.springBoot_firstExample
  src/main/resources
    static
    templates
    application.properties
  src/test/java
    com.springBoot_firstExample
  JRE System Library [JavaSE-21]
  Maven Dependencies
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

9

9

pom.xml

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

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
</dependency>
```

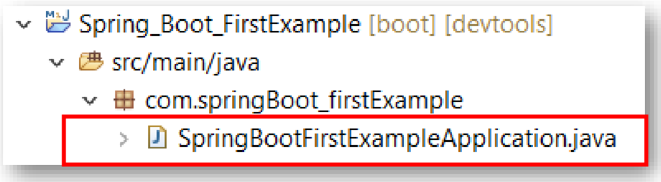


```
Maven Dependencies
  spring-aop-6.2.2.jar - C:
  spring-beans-6.2.2.jar -
  spring-boot-3.4.2.jar - (
  spring-boot-autoconfigure-3.4.2.jar -
  spring-boot-devtools-3.4.2.jar - C:\Us
  spring-boot-starter-3.4.2.jar - C:\Users
  spring-boot-starter-json-3.4.2.jar - C:\
  spring-boot-starter-logging-3.4.2.jar
  spring-boot-starter-test-3.4.2.jar - C:\
  spring-boot-starter-thymeleaf-3.4.2.ja
  spring-boot-starter-tomcat-3.4.2.jar -
  spring-boot-starter-web-3.4.2.jar - C:\
  spring-boot-test-3.4.2.jar - C:\
  spring-boot-test-autoconfigure-3.4.2.
  spring-context-6.2.2.jar - C:\
  spring-core-6.2.2.jar - C:\Use
  spring-expression-6.2.2.jar -
  spring-jcl-6.2.2.jar - C:\Users
  spring-test-6.2.2.jar - C:\Use
  spring-web-6.2.2.jar - C:\Use
  spring-webmvc-6.2.2.jar - C:
  thymeleaf-3.1.3.RELEASE.jar - C:\Users
  thymeleaf-spring6-3.1.3.RELEASE.jar -
```

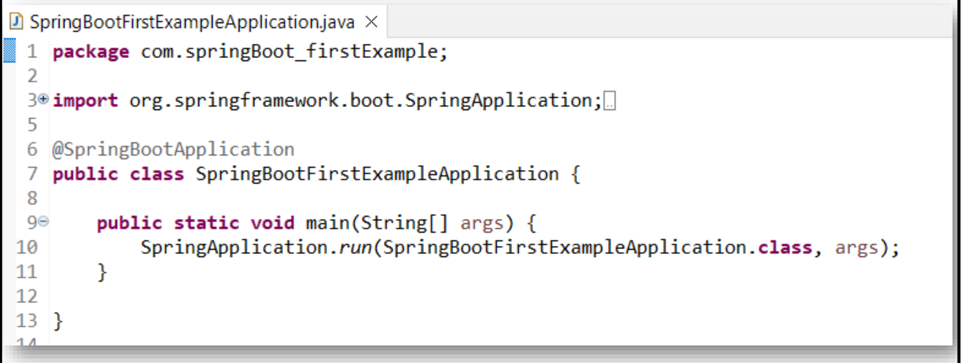
10

10

@SpringBootApplication



```
Spring_Boot_FirstExample [boot] [devtools]
  src/main/java
    com.springBoot_firstExample
      > SpringBootFirstExampleApplication.java
```



SpringBootFirstExampleApplication.java

```java
package com.springBoot_firstExample;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class SpringBootFirstExampleApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringBootFirstExampleApplication.class, args);
    }

}
```

11

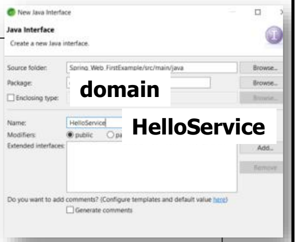
MVC

MODEL

Implementing a Service

Start by creating the **HelloService** interface.

```java
package domain;

public interface HelloService
{
    public String sayHello(String name);
}
```



New Java Interface

Java Interface

Create a new Java interface.

Source folder: Spring_Web_FirstExample/src/main/java

Package: **domain**

Enclosing type:

Name: HelloService **HelloService**

Modifiers: [✓ public]

Extended interfaces:

Do you want to add comments? (Configure templates and default value here)

Generate comments

12

Implementing a Service

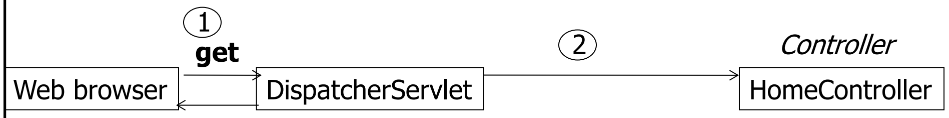
The **HelloServiceImpl** class performs a very **simple service**. It takes a name as a parameter and prepares and returns a String that includes the name.

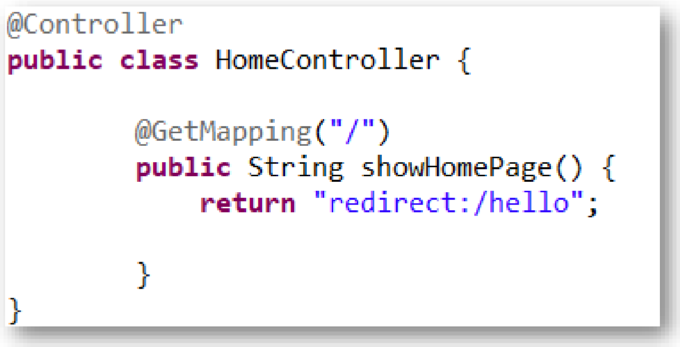
```
package domain;

public class HelloServiceImpl implements HelloService{

      @Override
     public String sayHello(String name) {
        return "Hello %s!".formatted(
                (name != null)?name:"");    }
}
```

13

13

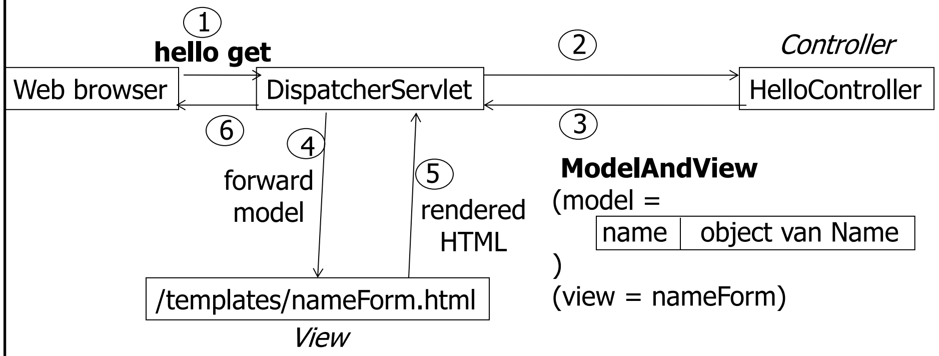
http://localhost:8080/

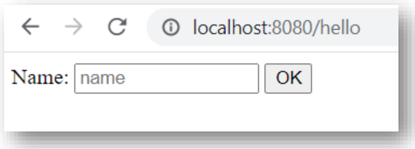




```
@Controller
public class HomeController {

        @GetMapping("/")
        public String showHomePage() {
            return "redirect:/hello";

        }
}
```

14





15

Controller

```
package com.springBoot.firstExample;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;


//This class is a controller class
@Controller
public class HelloController {
```

16

16

```
    // 'showFormPage'  is a request-handling method.
    // It handles requests whose path is /hello
    @GetMapping("/hello")
    public String showFormPage(Model model) {
        model.addAttribute("name", new Name());
        return "nameForm";
    }


}
```

17

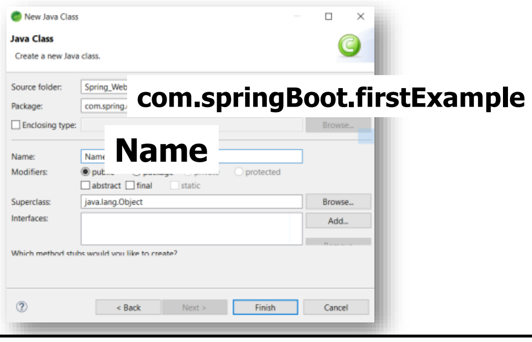
17

Implementing the Command Class

Note that an error is flagged for Name in the controller class.

You need to create the **Name class** as a **simple bean** **to hold information for each request.**

New > Java Class



New Java Class

Java Class

Create a new Java class.

Source folder: Spring_Web

Package: com.spring

com.springBoot.firstExample

Enclosing type:

Name: Name

Name

Modifiers: [✓ public] package private protected

abstract final static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

18

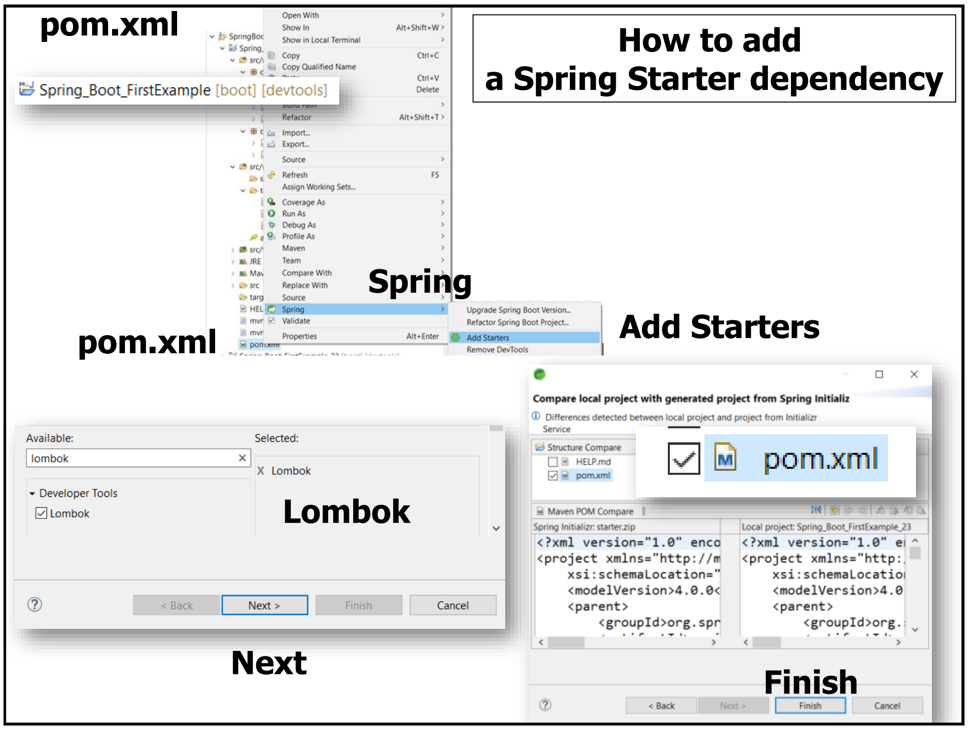
18

```
package com.springBoot_firstExample;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class Name {

        private String value;

}
```

19

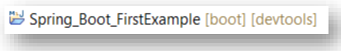
19

How to add a Spring Starter dependency



```
<?xml version="1.0" enco          <?xml version="1.0" e
<project xmlns="http://m          <project xmlns="http:
    xsi:schemaLocation="              xsi:schemaLocatio
    <modelVersion>4.0.0<              <modelVersion>4.0
    <parent>                          <parent>
        <groupId>org.spr                  <groupId>org.
```

20

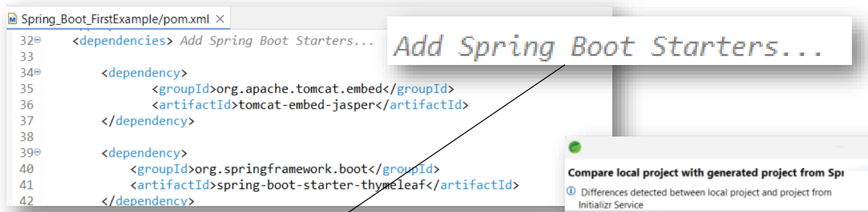


Spring_Boot_FirstExample [boot] [devtools]

OR

How to add a Spring Starter dependency

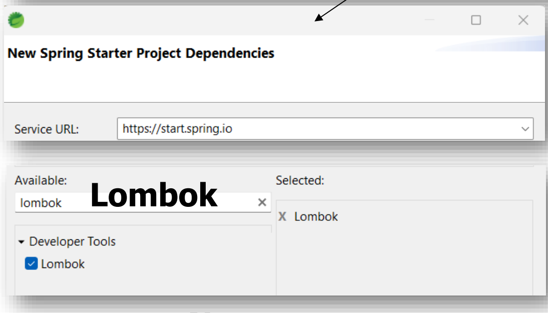
pom.xml



Spring_Boot_FirstExample/pom.xml

```
    <dependencies> Add Spring Boot Starters...

        <dependency>
                <groupId>org.apache.tomcat.embed</groupId>
                <artifactId>tomcat-embed-jasper</artifactId>
        </dependency>

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

Add Spring Boot Starters...



New Spring Starter Project Dependencies

Service URL: https://start.spring.io

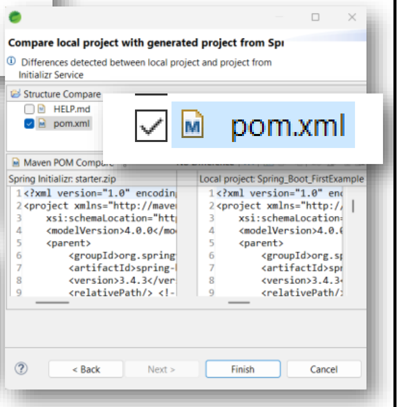
Available: lombok

Lombok

Selected: X Lombok

Developer Tools

☑ Lombok



Compare local project with generated project from Spi

Differences detected between local project and project from Initializr Service

Structure Compare

HELP.md

☑ pom.xml

☑ pom.xml

Maven POM Compare

Spring Initializr: starter.zip

```
<?xml version="1.0" encodin
<project xmlns="http://mave
    xsi:schemaLocation="htt
    <modelVersion>4.0.0</mo
    <parent>
        <groupId>org.spring
        <artifactId>spring-
        <version>3.4.3</ver
        <relativePath/> <!-
```

Local project: Spring_Boot_FirstExample

```
<?xml version="1.0" en
<project xmlns="http:/
    xsi:schemaLocation=
    <modelVersion>4.0.
    <parent>
        <groupId>org.sp
        <artifactId>spr
        <version>3.4.3
        <relativePath/
```

Next

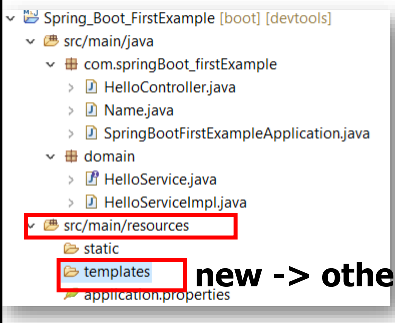
Finish

21

View

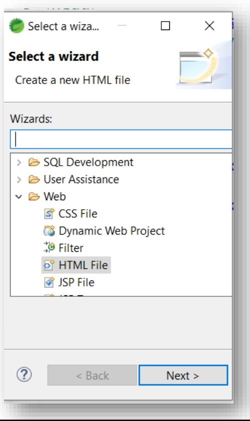
Create a HTML:

templates/nameForm.html: serves as the welcome page and allows a user to input a name.



```
Spring_Boot_FirstExample [boot] [devtools]
  src/main/java
    com.springBoot_firstExample
      HelloController.java
      Name.java
      SpringBootFirstExampleApplication.java
    domain
      HelloService.java
      HelloServiceImpl.java
  src/main/resources
    static
    templates
    application.properties
```

new -> other

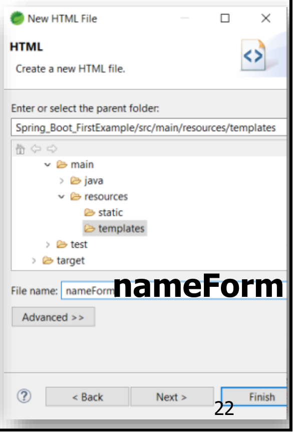


Select a wizard

Select a wizard

Create a new HTML file

Wizards:



New HTML File

HTML

Create a new HTML file.

Enter or select the parent folder:

Spring_Boot_FirstExample/src/main/resources/templates

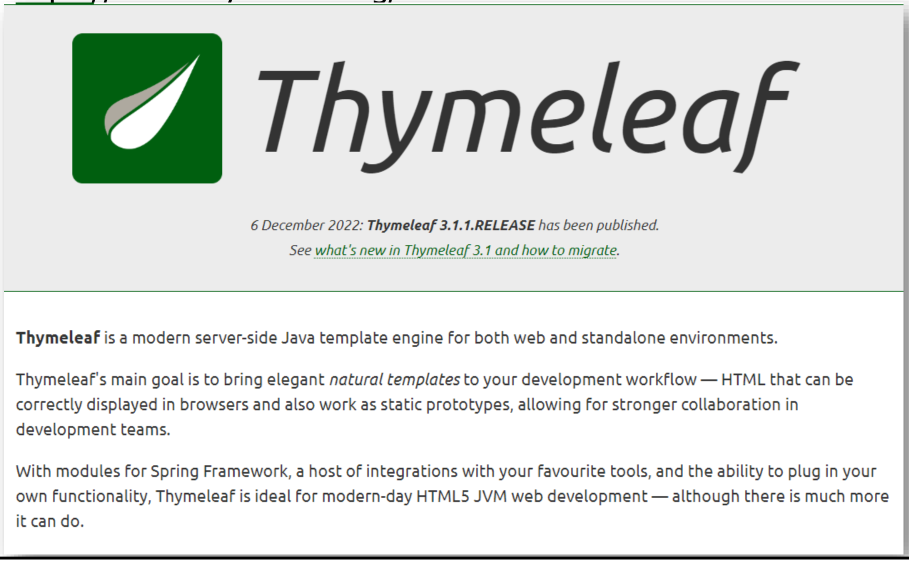
File name: nameForm

nameForm

22

Implementing the Views

https://www.thymeleaf.org/



Thymeleaf

6 December 2022: ***Thymeleaf 3.1.1.RELEASE*** *has been published.*

See what's new in Thymeleaf 3.1 and how to migrate.

Thymeleaf is a modern server-side Java template engine for both web and standalone environments.

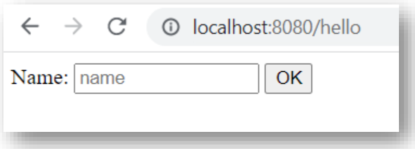
Thymeleaf's main goal is to bring elegant *natural templates* to your development workflow — HTML that can be correctly displayed in browsers and also work as static prototypes, allowing for stronger collaboration in development teams.

With modules for Spring Framework, a host of integrations with your favourite tools, and the ability to plug in your own functionality, Thymeleaf is ideal for modern-day HTML5 JVM web development — although there is much more it can do.

23

Implementing the Views

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
        <meta charset="ISO-8859-1">
        <title>Insert name</title>
</head>
<body>
  <form th:action="@{/hello}" th:object="${name}" method="post">

    Name:
    <input type="text" th:field="*{value}" size="15" placeholder="name"/>
    <button type="submit">OK</button>

  </form>

</body>
</html>
```



24

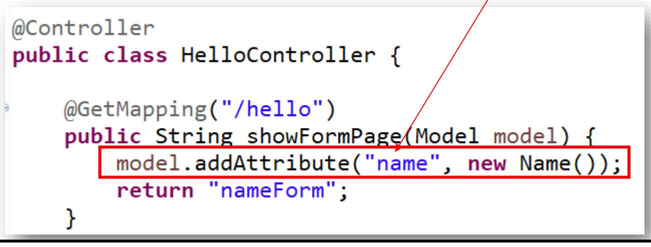
24

<form>

<form th:action="@{/hello}" th:object="${name}" method="post">

The <form> tag binds the **Name object** (identified by the **modelAttribute** attribute) that showFormPage (Model model) placed into the model to the various fields in the form.

It will be submitted as an **HTTP POST** request



```
@Controller
public class HelloController {

    @GetMapping("/hello")
    public String showFormPage(Model model) {
        model.addAttribute("name", new Name());
        return "nameForm";
    }
```

<input type>

<form th:action="@{/hello}" th:object="${name}" method="post">

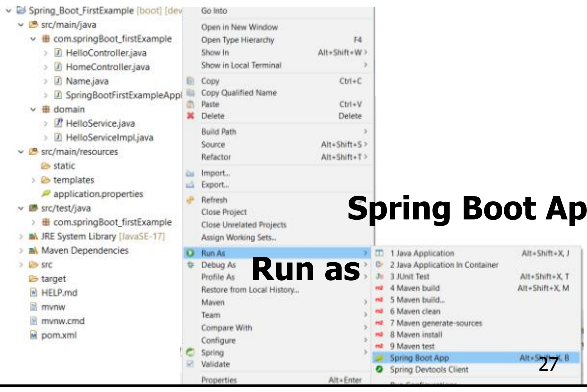
<input type="text" th:field="*{value}" size="15" placeholder="name"/>

The <input> tag has a **path** attribute that references the **property** of the Name object that the form is bound to.

```
public class Name {
    private String value;
    public String getValue() ...
    public void setValue(String value) ...
```

HomeController

```
package com.springBoot.firstExample;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
@Controller
public class HomeController {

        @GetMapping("/")
        public String showHomePage() {
                return "redirect:/hello";
        }
}
```

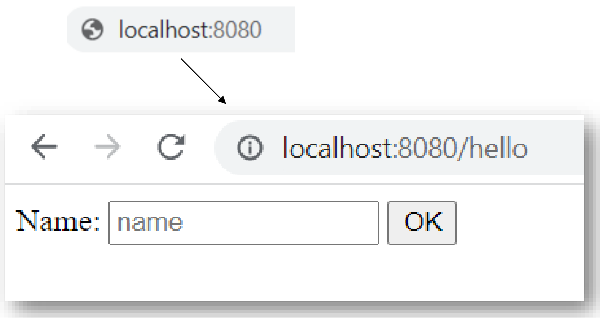


Spring Boot App

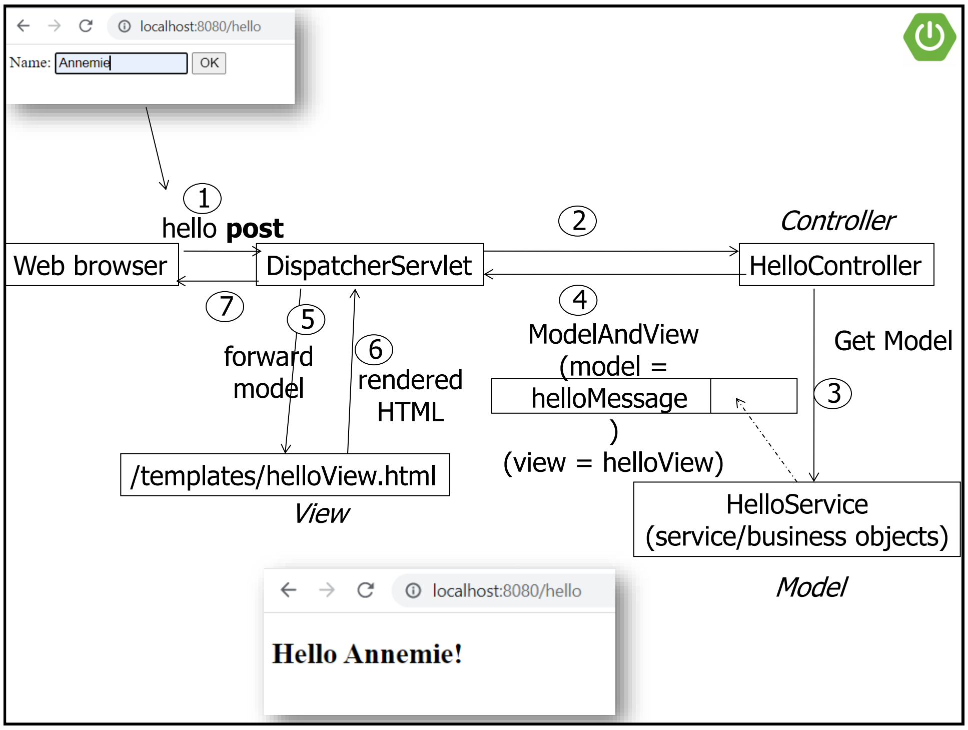
Run as

27

enter localhost:8080



28



Name: Annemie OK

Hello Annemie!

<form:form>

<form th:action="@{/hello}" th:object="${name}"
method="post">

HelloController:
@PostMapping("/hello")
public String onSubmit(...

Controller

```
package com.springBoot.firstExample;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import domain.HelloService;

//This class is a controller class
@Controller
public class HelloController {
```

31

31

```
//HelloService-bean will be injected when the controller
//is instantiated
@Autowired
private HelloService helloService;


// 'showHomePage'  is a request-handling method.
// It handles requests whose path is /hello
@GetMapping("/hello")
public String showFormPage(Model model) {
    model.addAttribute("name", new Name());
    return "nameForm";
}
```

32

32

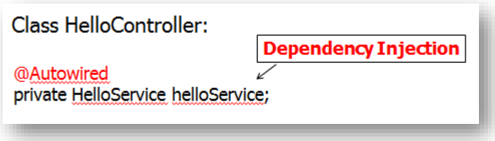
```
//This method will handle POST requests
@PostMapping("/hello")
public String onSubmit(Name name, Model model){

      model.addAttribute("helloMessage",
                   helloService.sayHello(name.getValue()));
    return "helloView";
 }
```

import org.springframework.ui.**Model**;
→ a ModelMap with the model attributes for the view

33

33

```
package com.springBoot.firstExample;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import domain.HelloService;
import domain.HelloServiceImpl;

@SpringBootApplication
public class SpringBootFirstExampleApplication {

     public static void main(String[] args) {
         SpringApplication.run(SpringBootFirstExampleApplication.class, args);
    }

    @Bean
    HelloService helloService() {
         return new HelloServiceImpl();
    }

}
```


Class HelloController:

Dependency Injection

```
@Autowired
private HelloService helloService;
```

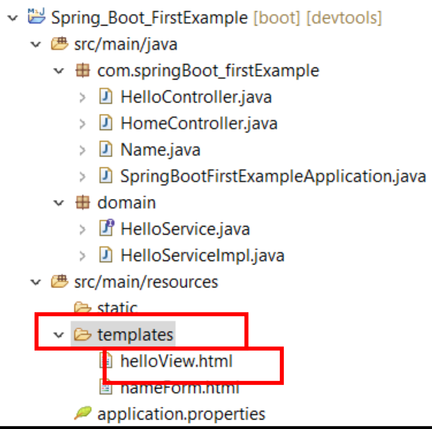
34

34

View

Create a HTML class:

templates/helloView.html: displays a greeting message that includes the input name.

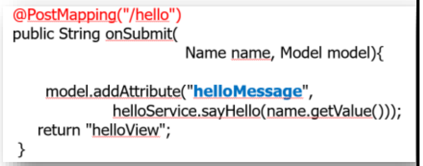


```
Spring_Boot_FirstExample [boot] [devtools]
  src/main/java
    com.springBoot_firstExample
      HelloController.java
      HomeController.java
      Name.java
      SpringBootFirstExampleApplication.java
    domain
      HelloService.java
      HelloServiceImpl.java
  src/main/resources
    static
    templates
      helloView.html
      nameForm.html
    application.properties
```

35

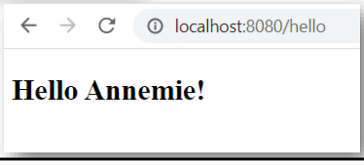
Implementing the Views

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Hello</title>
</head>
<body>

   <h2 th:text="${helloMessage}"></h2>

</body>
</html>
```



```
@PostMapping("/hello")
public String onSubmit(
                    Name name, Model model){

    model.addAttribute("helloMessage",
            helloService.sayHello(name.getValue()));
   return "helloView";
 }
```

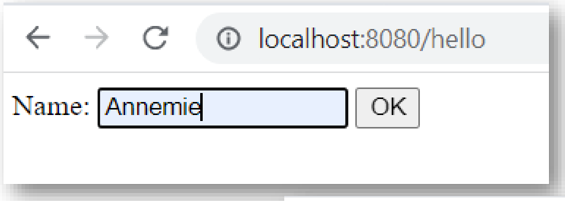
Jakarta Expression Language



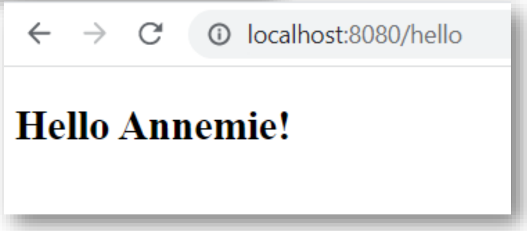
localhost:8080/hello

Hello Annemie!

localhost:8080


localhost:8080/hello

Name: Annemie OK


localhost:8080/hello

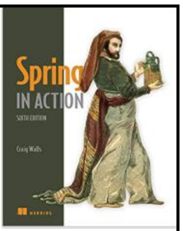
Hello Annemie!

By **spring-boot-devtools**, when you change code your application was automatically restarted.

37

37

4. Spring Boot Devtools



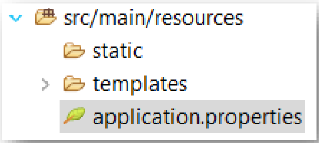
- **Automatic application restart when code changes**
- **Automatic browser refresh when browser-destined resources change**
- **Automatic disabling of template caching**
  - **Cached templates are not so great at development time**

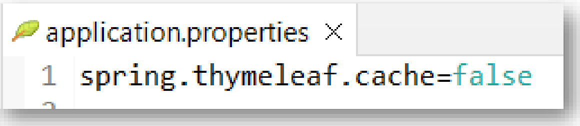
If you don't use Spring Boot Devtools

Thymeleaf:

By default: enable caching = true

To disable Thymeleaf caching:


src/main/resources
static
templates
application.properties

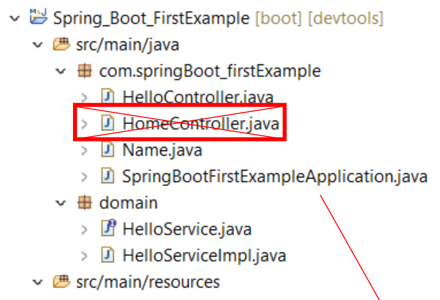

application.properties

```
spring.thymeleaf.cache=false
```

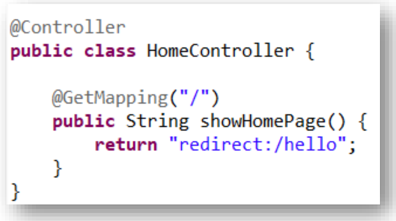
38

38

5. HomeController



```
Spring_Boot_FirstExample [boot] [devtools]
  src/main/java
    com.springBoot_firstExample
      HelloController.java
      ~~HomeController.java~~
      Name.java
      SpringBootFirstExampleApplication.java
    domain
      HelloService.java
      HelloServiceImpl.java
  src/main/resources
```



```java
@Controller
public class HomeController {

    @GetMapping("/")
    public String showHomePage() {
        return "redirect:/hello";
    }
}
```

```java
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
@SpringBootApplication
public class SpringBootFirstExampleApplication
                                        implements WebMvcConfigurer {
...
        @Override
        public void addViewControllers(ViewControllerRegistry registry) {
            registry.addRedirectViewController("/", "/hello");
        }
```

6. RequestMapping

```java
@Controller
@RequestMapping("/hello")
public class HelloController {

    @Autowired
    private HelloService helloService;

    @GetMapping//("/hello")
    public String showFormPage(Model model) {
        ...
    }

    @PostMapping//("/hello")
    public String onSubmit(Name name, Model model) {
        ...
    }
```

Using @RequestMapping("/hello") at the class level sets a base path for all endpoints in the controller. This way, @GetMapping and @PostMapping don't need to repeat "/hello", making the code cleaner.