

FIAP GRADUAÇÃO

SISTEMAS DE INFORMAÇÃO

MICROSERVICE AND WEB ENGINEERING

Prof^a. Aparecida Castello Rosa
profaparecida.rosa@fiap.com.br

Material de apoio:
PROF. PEDRO IVO CORREIA
PROF. LUCAS FURLANETO

Agenda

- Maven
- Introdução Spring MVC
- Arquitetura MVC
- Criando um projeto Spring MVC
- Annotations
- *Thymeleaf*

Objetivos

- Entender o funcionamento do padrão arquitetural MVC.
- Conhecer o funcionamento do Spring MVC.
- Entender Maven, pom.xml.
- Desenvolver primeira aplicação Spring MVC com *Thymeleaf* – Controller e View – Hello World

Apache Maven

I Maven

- Apache Maven é uma ferramenta de automação de compilação utilizada na maioria das vezes em projetos Java.
- Também é utilizada para construir e gerenciar projetos escritos em C#, Ruby, Scala e outras linguagens.
- O Maven é uma ferramenta para fazer duas coisas importantes:
 1. Padronizar a estrutura do projeto;
 2. Realizar o download de dependências.



MVC (Model-View-Controller)

I MVC (Model-View-Controller)

O Design Pattern **Front Controller**

Definição de **MVC**

- **Model:** Define o modelo ou domínio da aplicação
 - Regras de negócios
 - Persistência de dados
- **View:** Interação com usuário
- **Controller:** Componente intermediário entre **View** e **Model**
 - Recebe requisições (requests)
 - Envia respostas (responses)
 - Interage com a camada model

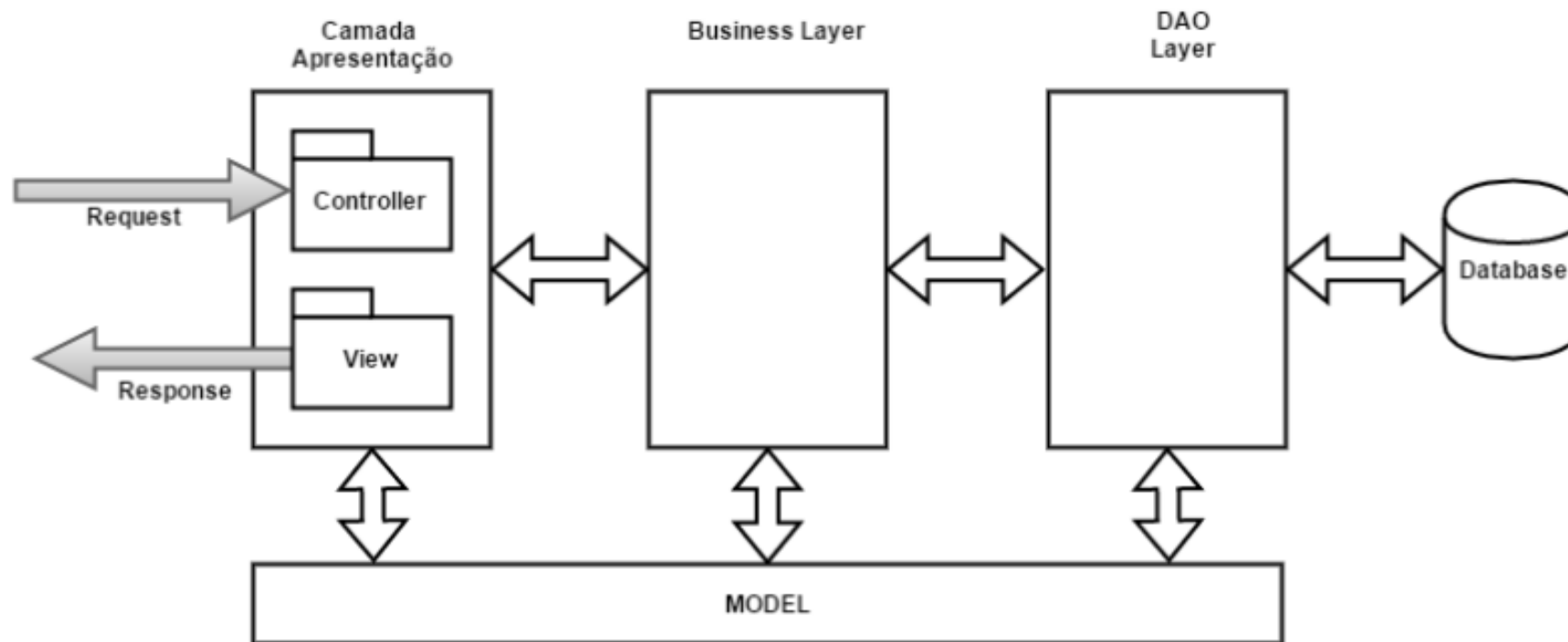
Spring MVC

Spring MVC

- O framework Spring, é um dos frameworks Java mais conhecido e utilizado no mercado.
- Ele implementa um grande número de funcionalidades, como injeção de dependência, persistência de dados e uma implementação para o **padrão MVC** para a criação de **aplicações WEB**.

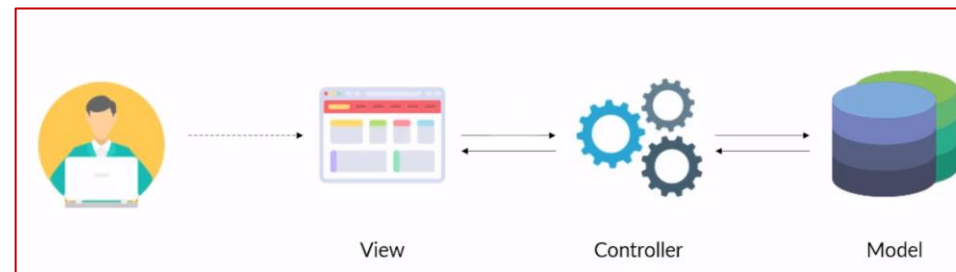
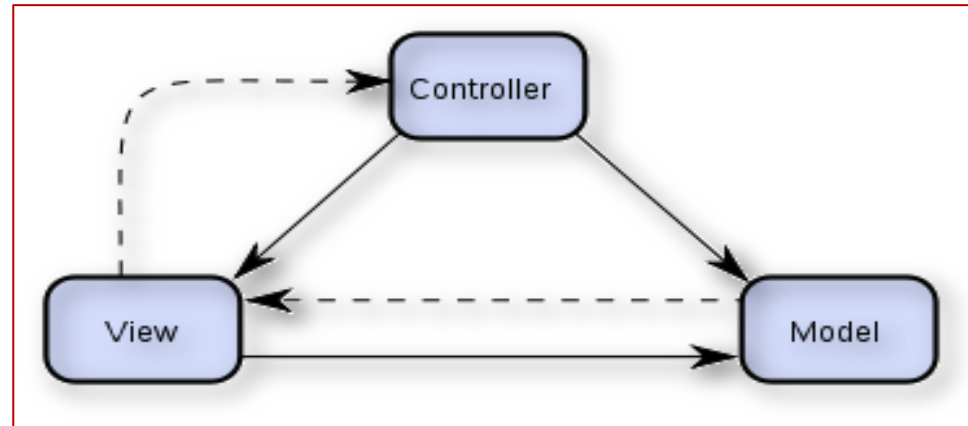


Arquitetura Java - Camadas da Aplicação Web



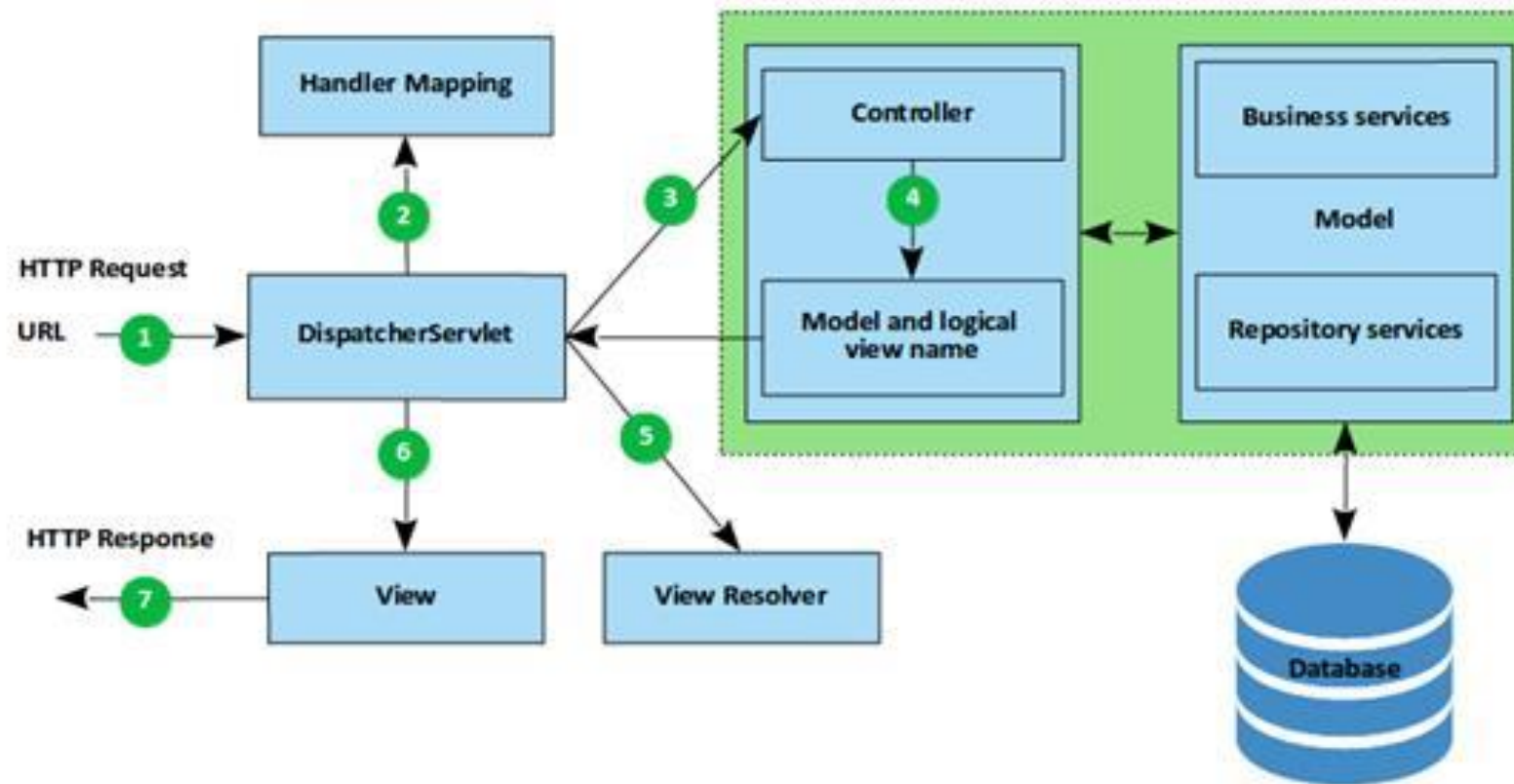
Arquitetura Java - Pattern Arquitetural - MVC

- Model - View - Controller



Arquitetura Java - Funcionamento do Spring MVC ^{FIAP}

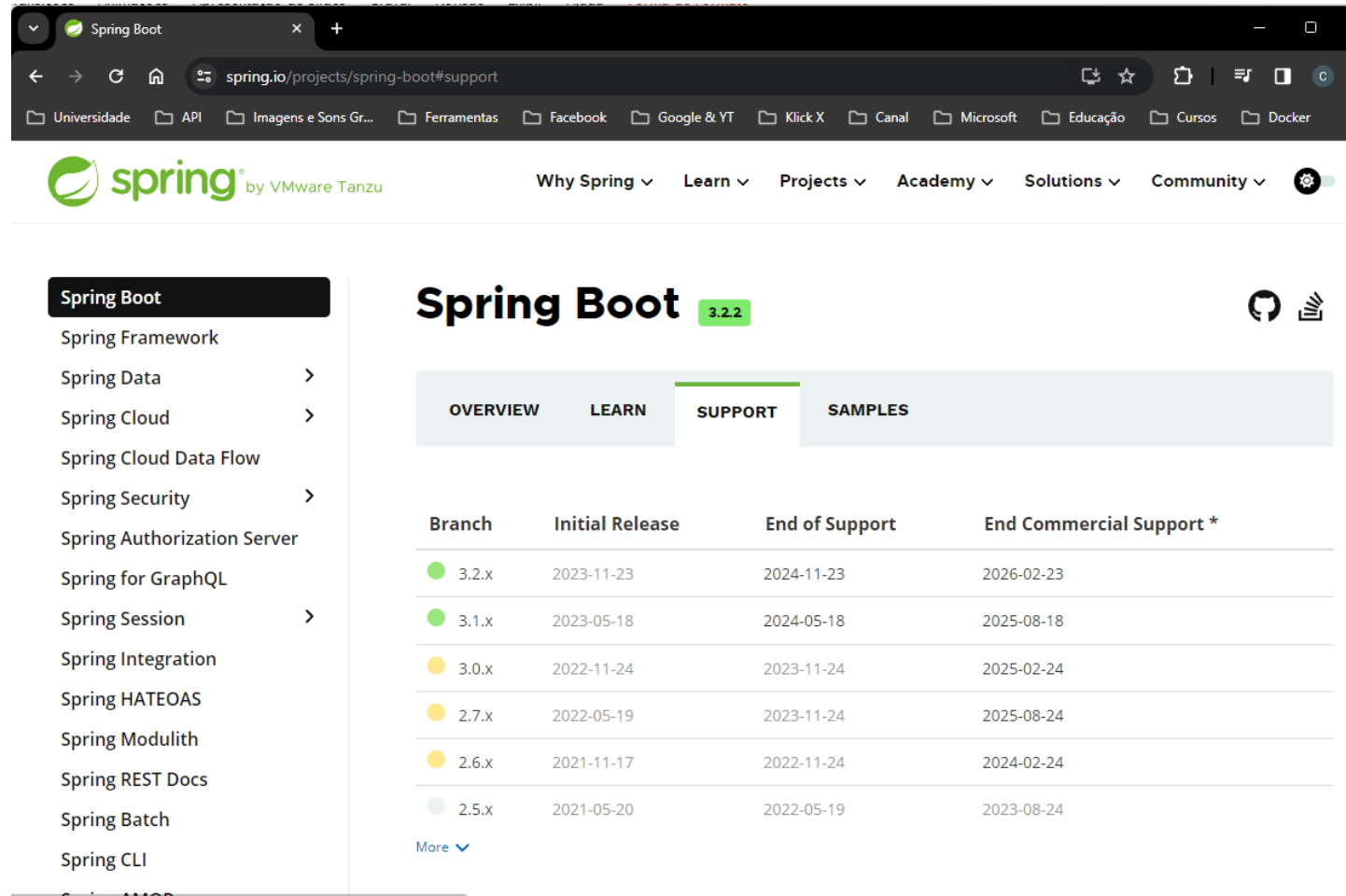
- Model - View - Controller



Spring

Spring Boot
Spring Framework

Spring Boot - versão



The screenshot shows the Spring Boot support page on the spring.io website. The page is titled "Spring Boot" with a version indicator "3.2.2". The "SUPPORT" tab is selected in the navigation bar. A table lists the support dates for various Spring Boot versions, including 3.2.x, 3.1.x, 3.0.x, 2.7.x, 2.6.x, and 2.5.x. The table columns are Branch, Initial Release, End of Support, and End Commercial Support *.

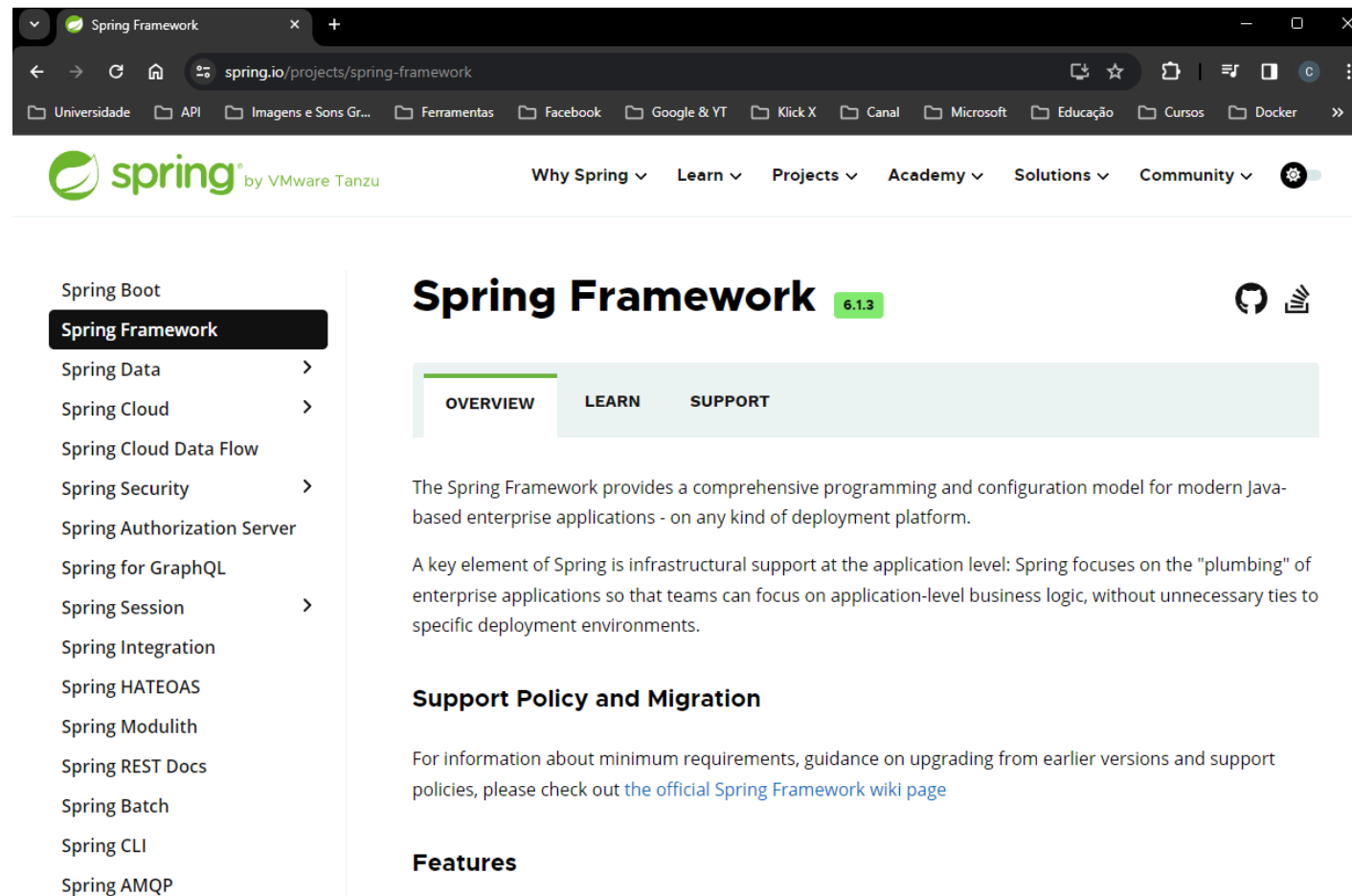
Branch	Initial Release	End of Support	End Commercial Support *
3.2.x	2023-11-23	2024-11-23	2026-02-23
3.1.x	2023-05-18	2024-05-18	2025-08-18
3.0.x	2022-11-24	2023-11-24	2025-02-24
2.7.x	2022-05-19	2023-11-24	2025-08-24
2.6.x	2021-11-17	2022-11-24	2024-02-24
2.5.x	2021-05-20	2022-05-19	2023-08-24

<https://spring.io/projects/spring-boot#support>

AULA 03 – Spring MVC e Thymeleaf – Controller e View – Hello World

Prof.^a Aparecida F. Castello Rosa – profaparecida.rosa@fiap.com.br

Spring Framework - versão



<https://spring.io/projects/spring-framework>

AULA 03 – Spring MVC e Thymeleaf – Controller e View – Hello World

Prof.^a Aparecida F. Castello Rosa – profaparecida.rosa@fiap.com.br

■ Spring Boot 3 e Spring Framework 6 - Mudanças

- **Spring Boot 3.X** requer no mínimo a versão **LTS Java 17**, ou seja, se for utilizar uma versão do Java anterior a essa não conseguirá utilizar o Spring Boot 3.
- O **Spring Framework 6.0** também requer o **Java 17** ou superior para compatibilidade e também no mínimo o Jakarta EE versão 9.
- A nomenclatura dos pacotes *javax*, que foram renomeados para **jakarta**, como, por exemplo: *jakarta.validation*; *jakarta.persistence*; etc.

Spring Boot 3.0 Migration Guide

The screenshot shows the GitHub repository page for the Spring Boot 3.0 Migration Guide. The repository is owned by 'spring-projects' and is named 'spring-boot'. It is a public repository with 40.2k forks and 71.5k stars. The page title is 'Spring Boot 3.0 Migration Guide' by Andy Wilkinson, edited 3 weeks ago with 32 revisions. The document is intended to help migrate applications to Spring Boot 3.0. It includes sections for 'Before You Start', 'Upgrade to the Latest 2.7.x Version', and 'Review Dependencies'. The 'Upgrade to the Latest 2.7.x Version' section advises upgrading to the latest 2.7.x version before starting the migration. The 'Review Dependencies' section notes that the move to Spring Boot 3 will upgrade dependencies and might require work on the user's end, suggesting a review of dependency management for 2.7.x and 3.0.x. A sidebar on the right shows 'Pages' (210), 'Release Notes' (v3.2, v3.1, v3.0, v2.7, Older Versions), and 'Migration Guides' (v2.7 -> v3.0).

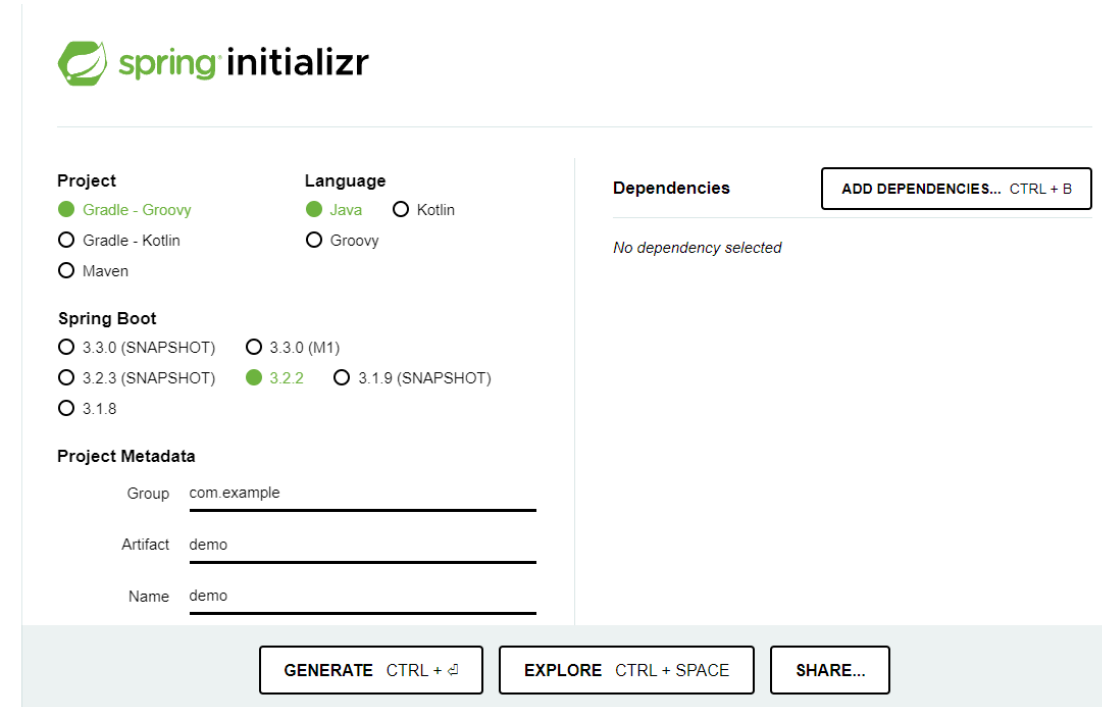
<https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-3.0-Migration-Guide>

AULA 03 – Spring MVC e Thymeleaf – Controller e View – Hello World

Prof.^a Aparecida F. Castello Rosa – profaparecida.rosa@fiap.com.br

Referências Spring

- <https://spring.io/quickstart>
- <https://start.spring.io/>



The image shows the Spring Initializr web form, which is used to generate a Spring project. The form is divided into several sections: Project, Language, Spring Boot, Project Metadata, and Dependencies. The Project section has radio buttons for Gradle - Groovy (selected), Gradle - Kotlin, and Maven. The Language section has radio buttons for Java (selected), Kotlin, and Groovy. The Spring Boot section has radio buttons for 3.3.0 (SNAPSHOT), 3.3.0 (M1), 3.2.3 (SNAPSHOT), 3.2.2 (selected), 3.1.9 (SNAPSHOT), and 3.1.8. The Project Metadata section has input fields for Group (com.example), Artifact (demo), and Name (demo). The Dependencies section has a button to add dependencies and a message indicating no dependency is selected. At the bottom, there are buttons for GENERATE, EXPLORE, and SHARE...

spring initializr

Project

☒ Gradle - Groovy
☐ Gradle - Kotlin
☐ Maven

Language

☒ Java ☐ Kotlin
☐ Groovy

Spring Boot

☐ 3.3.0 (SNAPSHOT) ☐ 3.3.0 (M1)
☐ 3.2.3 (SNAPSHOT) ☒ 3.2.2 ☐ 3.1.9 (SNAPSHOT)
☐ 3.1.8

Project Metadata

Group
Artifact
Name

Dependencies **ADD DEPENDENCIES... CTRL + B**

No dependency selected

GENERATE CTRL + G **EXPLORE CTRL + SPACE** **SHARE...**

■ Preparando o Ambiente - Recursos

- Java **JDK 17 LTS** ou superior
- Versão do **Spring Boot 3.X.X**
- **Spring Initializr** - Initializr generates spring boot project with just what you need to start quickly!

<https://start.spring.io/>

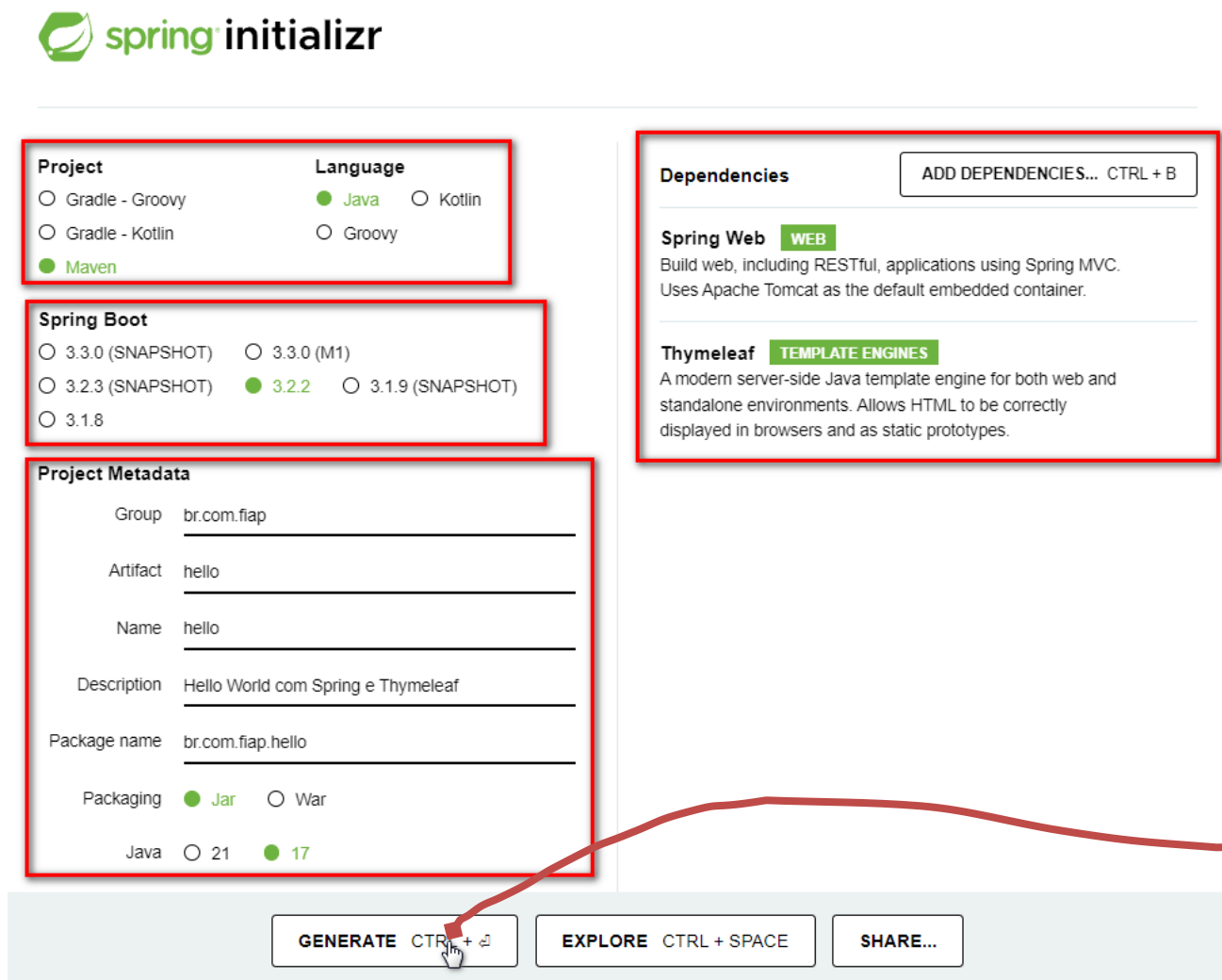
■ Preparando o Ambiente – IDE's

- Spring Tool Suite - <https://spring.io/tools>
- Eclipse JEE
- IntelliJ IDEA Community Edition - www.jetbrains.com/pt-br/idea/download/?section=windows
- VSCode - <https://code.visualstudio.com/>

Criando o projeto com Spring Initializr

Hello World

Projeto - Spring

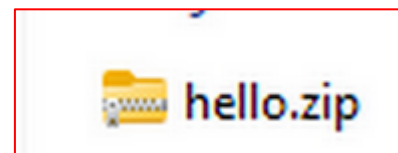


The image shows the Spring Initializr web form for creating a new project. The form is divided into several sections, each highlighted with a red border:

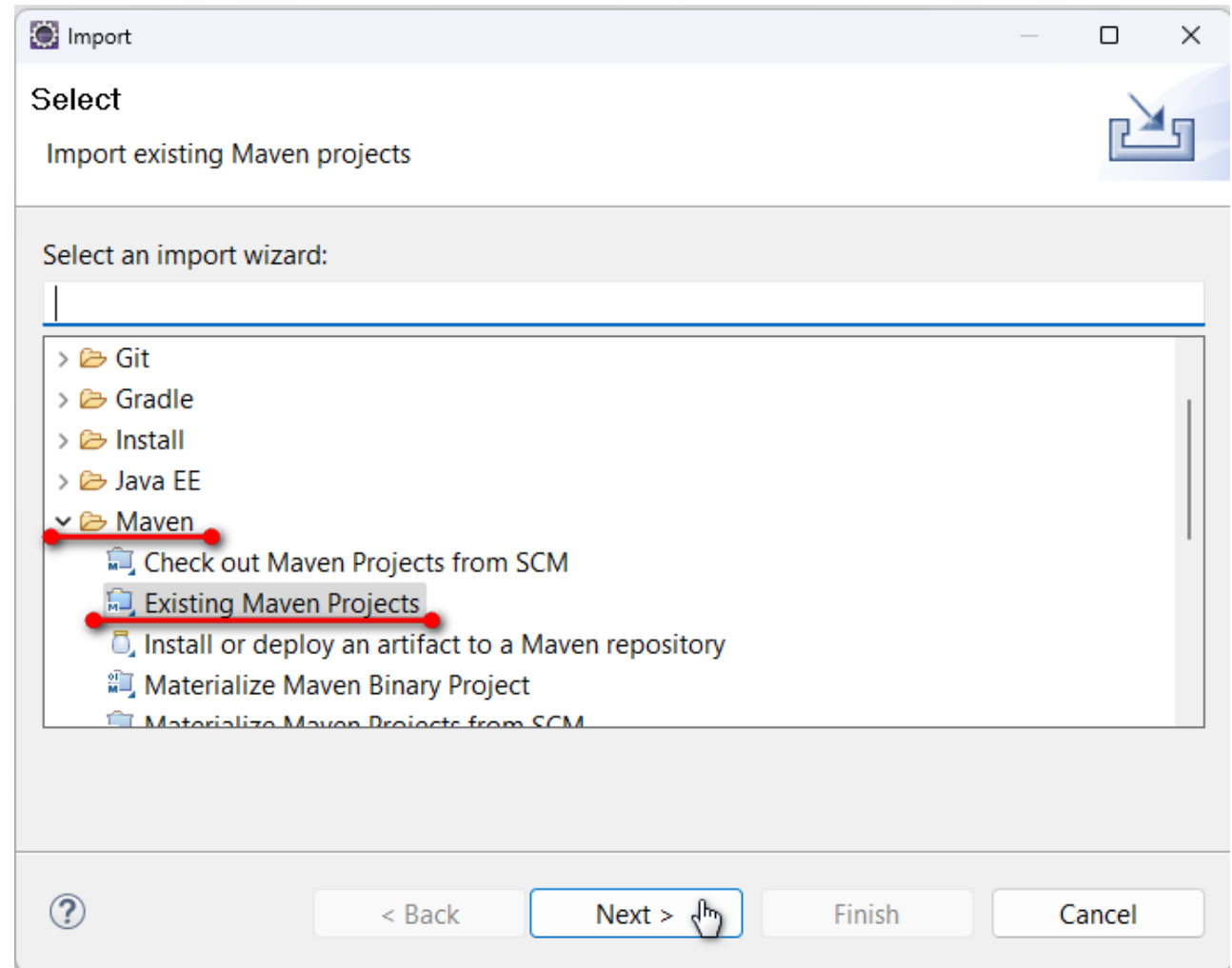
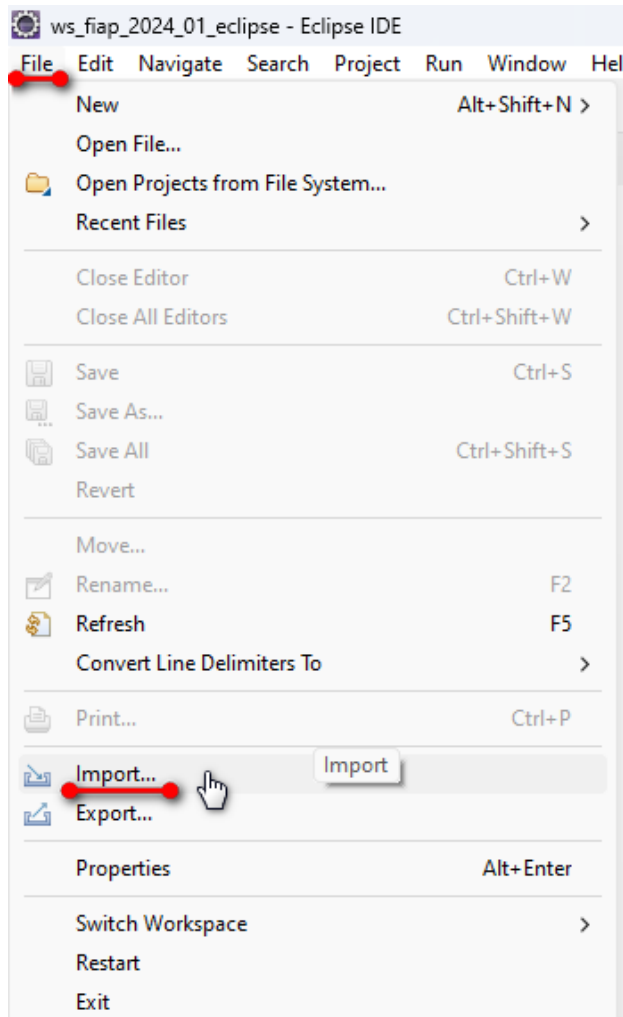
- Project:** Includes radio buttons for **Gradle - Groovy**, **Gradle - Kotlin**, and **Maven** (selected).
- Language:** Includes radio buttons for **Java** (selected), **Kotlin**, and **Groovy**.
- Spring Boot:** Includes radio buttons for versions **3.3.0 (SNAPSHOT)**, **3.3.0 (M1)**, **3.2.3 (SNAPSHOT)**, **3.2.2** (selected), **3.1.9 (SNAPSHOT)**, and **3.1.8**.
- Project Metadata:** Includes text input fields for **Group** (br.com.fiap), **Artifact** (hello), **Name** (hello), **Description** (Hello World com Spring e Thymeleaf), and **Package name** (br.com.fiap.hello). It also includes radio buttons for **Packaging** (**Jar** selected, War) and **Java** (**21**, **17** selected).
- Dependencies:** Includes a button **ADD DEPENDENCIES... CTRL + B** and two sections: **Spring Web** (WEB) and **Thymeleaf** (TEMPLATE ENGINES).

At the bottom, there are three buttons: **GENERATE CTRL + G**, **EXPLORE CTRL + SPACE**, and **SHARE...**. A red arrow points from the **GENERATE** button to a folder icon labeled **hello.zip** in a separate box on the right.

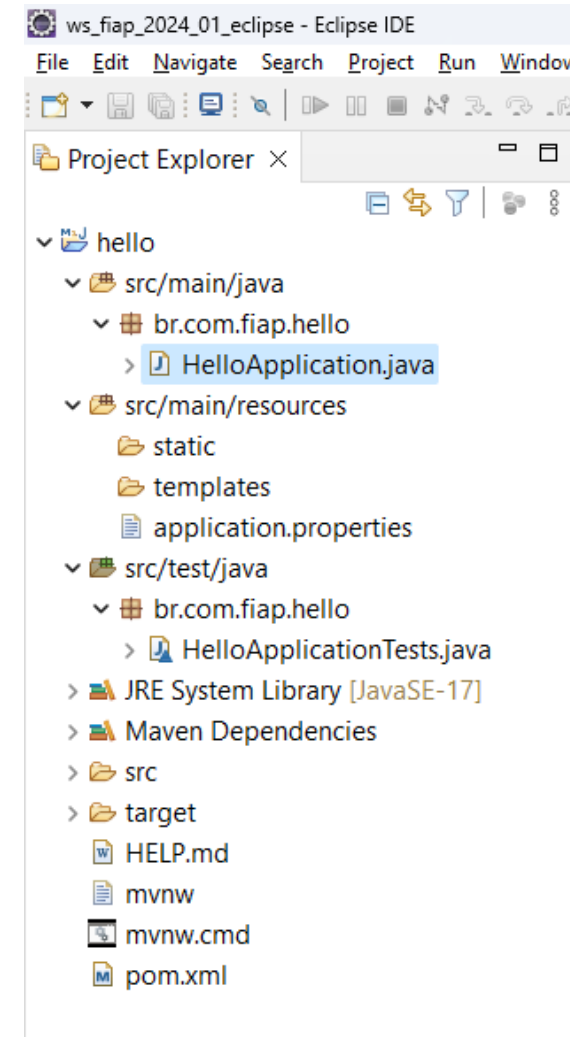
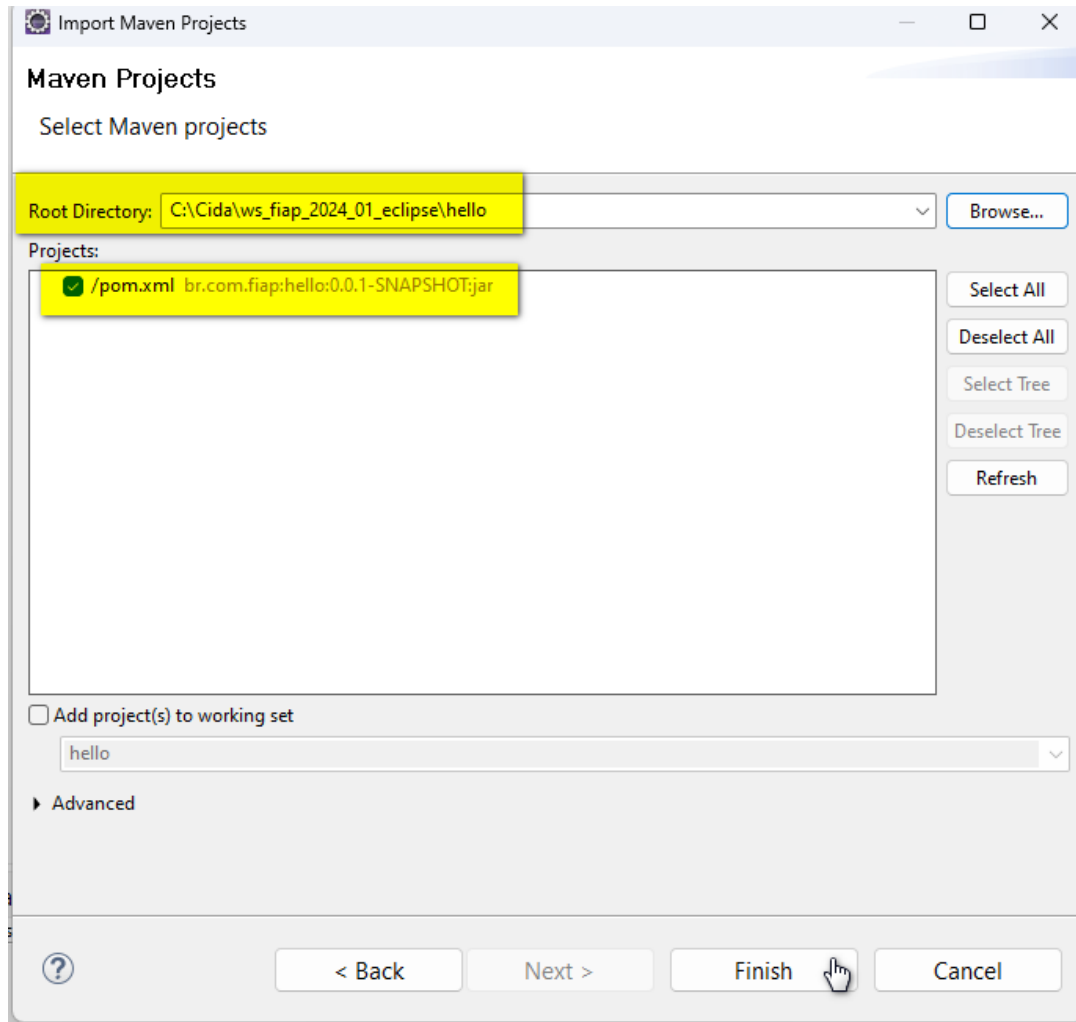
- Descompactar o arquivo em uma pasta.
- Importar/Abrir no IDE.

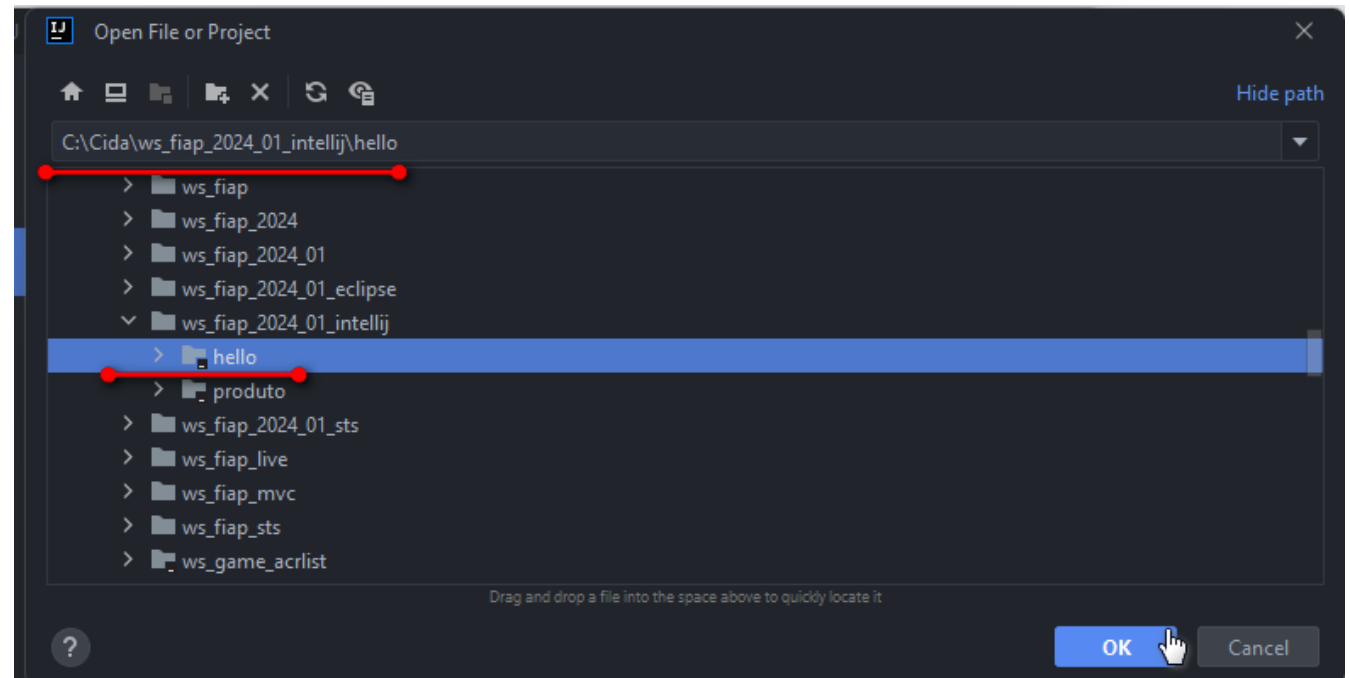
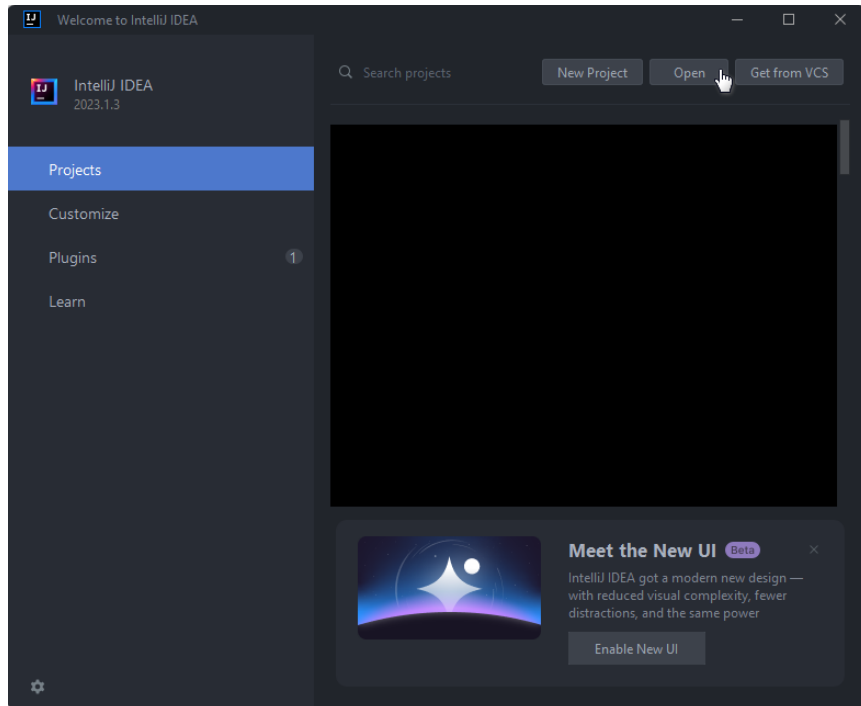


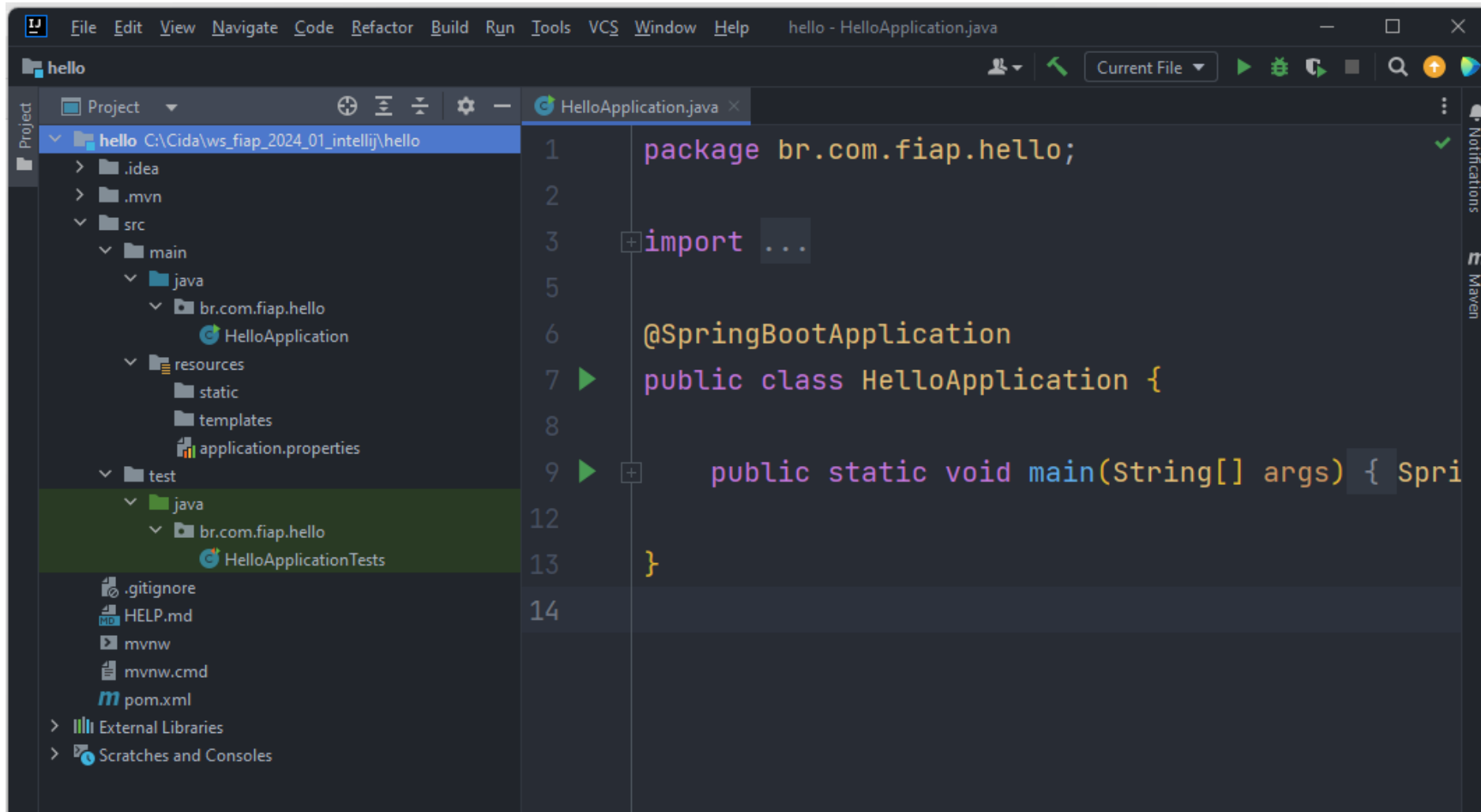
Eclipse IDE & Spring Tool Suite (STS)



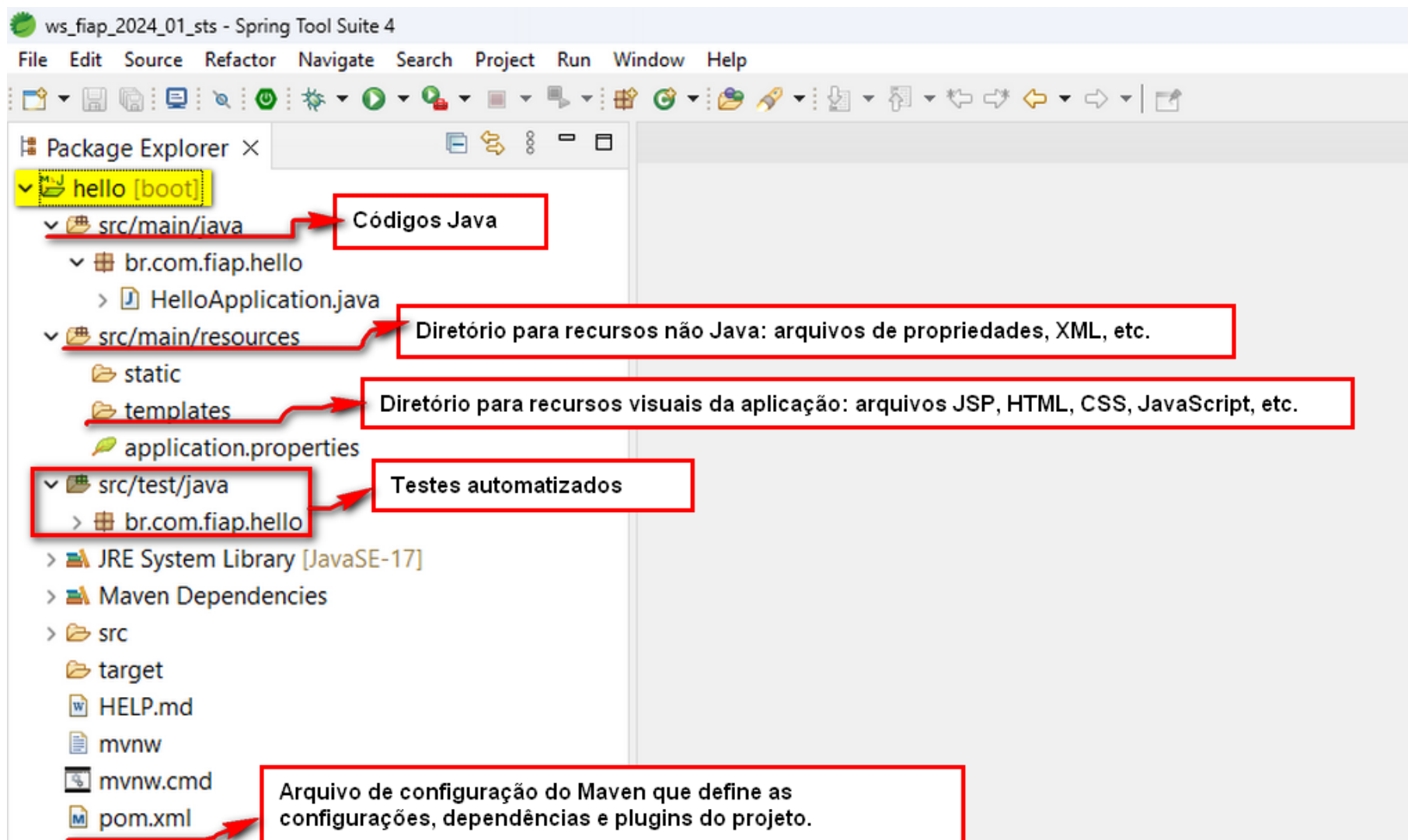
Eclipse IDE & Spring Tool Suite (STS)







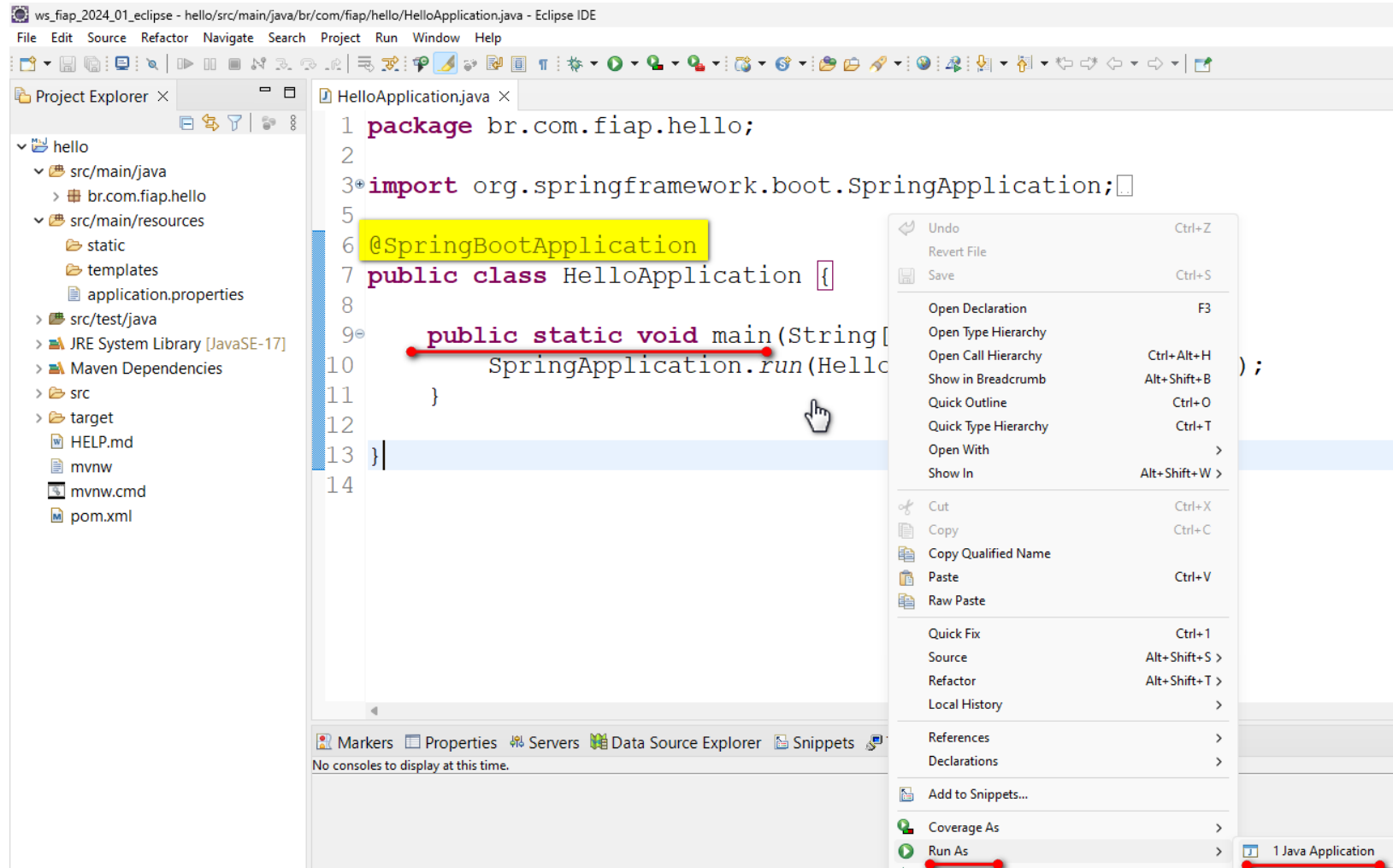
Estrutura do projeto



! pom.xml


```
hello/pom.xml x
2<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4  <modelVersion>4.0.0</modelVersion>
5  <parent>
6    <groupId>org.springframework.boot</groupId>
7    <artifactId>spring-boot-starter-parent</artifactId>
8    <version>3.2.2</version>
9    <relativePath/> <!-- lookup parent from repository -->
10 </parent>
11 <groupId>br.com.fiap</groupId>
12 <artifactId>hello</artifactId>
13 <version>0.0.1-SNAPSHOT</version>
14 <name>hello</name>
15 <description>Hello World com Spring e Thymeleaf</description>
16 <properties>
17   <java.version>17</java.version>
18 </properties>
19 <dependencies>
20   <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter-thymeleaf</artifactId>
23   </dependency>
24   <dependency>
25     <groupId>org.springframework.boot</groupId>
26     <artifactId>spring-boot-starter-web</artifactId>
27   </dependency>
28
29   <dependency>
30     <groupId>org.springframework.boot</groupId>
```

Executando a Aplicação



Tomcat

- O Tomcat é utilizado pelo Spring por padrão.



```

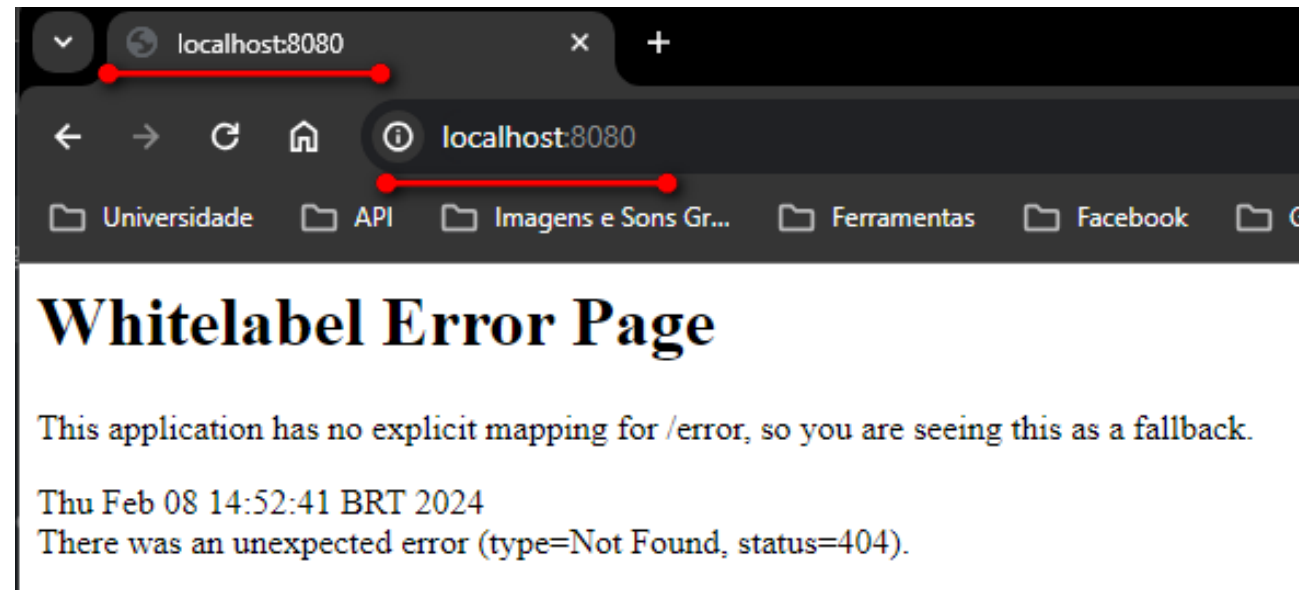
:: Spring Boot ::
(v3.2.2)

2024-02-08T14:50:07.998-03:00 INFO 24644 --- [main] br.com.fiap.hello.HelloApplication : Starting HelloApplication using Java 17.0.7 with PID 24644
2024-02-08T14:50:08.010-03:00 INFO 24644 --- [main] br.com.fiap.hello.HelloApplication : No active profile set, falling back to 1 default profile
2024-02-08T14:50:11.592-03:00 INFO 24644 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2024-02-08T14:50:11.623-03:00 INFO 24644 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-02-08T14:50:11.623-03:00 INFO 24644 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.18]
2024-02-08T14:50:11.799-03:00 INFO 24644 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-02-08T14:50:11.802-03:00 INFO 24644 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1.161 seconds
2024-02-08T14:50:12.965-03:00 INFO 24644 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''
2024-02-08T14:50:12.992-03:00 INFO 24644 --- [main] br.com.fiap.hello.HelloApplication : Started HelloApplication in 6.161 seconds (process running for 6.161 seconds)

```

Testando Tomcat

- O Tomcat está no ar.
- Essa página não é mensagem de erro, é que não temos nenhuma página HTML no projeto para apresentar.



Spring *Annotations*

Spring – Annotations

@Component

@Service

@Controller

@Repository

- Ao anotar uma classe com algum desses estereótipos, o Spring entende que essa classe é um *bean* e será gerenciada por ele.
- Um *bean* é um objeto que é instanciado, montado e gerenciado por um contêiner do Spring por meio da Inversão de Controle (IoC) e Injeção de Dependências.

Spring – Annotations

Anotação	Objetivo
@Component	Qualquer tipo de bean gerenciado pelo Spring
@Service	Classes com regras de negócio
@Autowired	Injeção de variável anotada

Spring – Annotations

Anotação (Data)	Objetivo
@Entity	Identifica a classe como mapeamento de tabela no BD
@Id	Identifica o atributo que representa a coluna para PK na tabela
@Repository	Classes de acesso a banco de dados
@Column	Altera propriedades da coluna no banco de dados
@Table	Altera o nome da table no banco de dados
@GeneratedValue	Usa uma estratégia para geração automática de número identificador único no atributo

Spring – Annotations

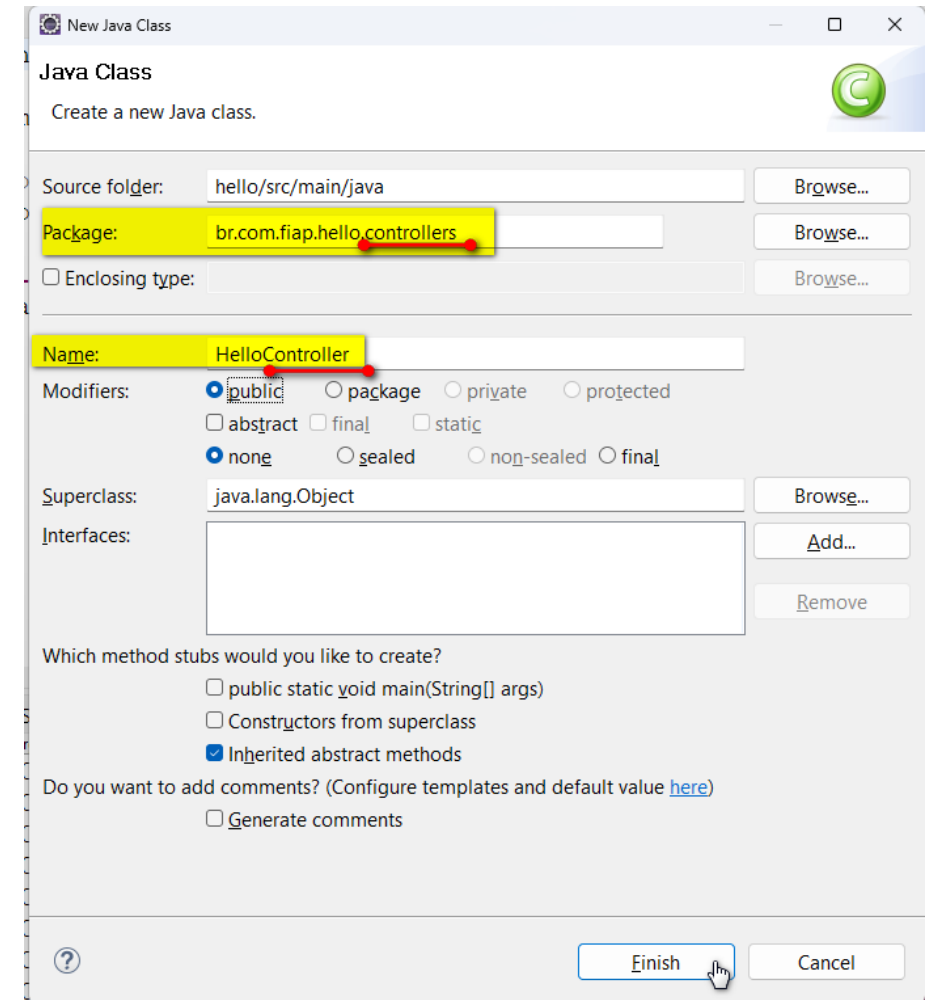
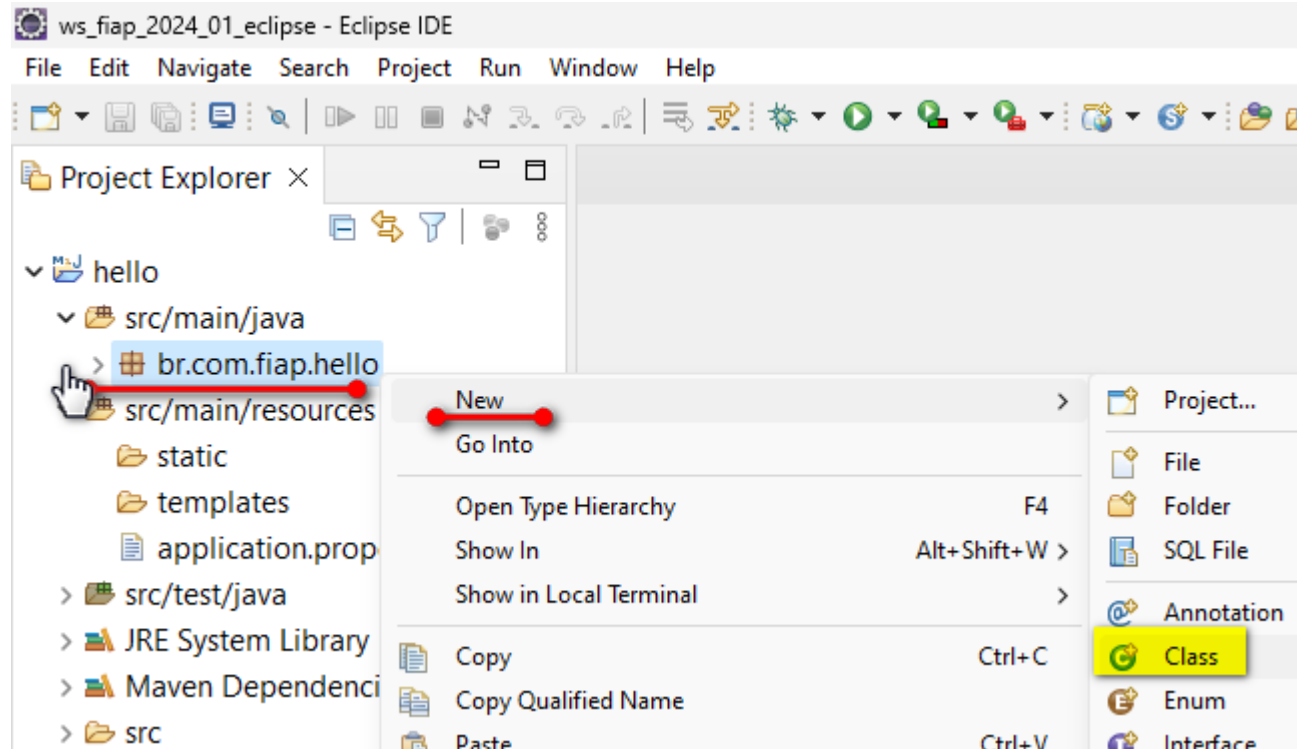
Anotação MVC (request)	Objetivo
@Controller	Classes de recursos que fornecem endpoints
@RequestMapping	Mapear URLs de acesso ao controller e aos métodos
@PathVariable	Extrair da URL um parâmetro incluído como path da URL
@RequestParam	Capturar um parâmetro de consulta (query param) GET
@ModelAttribute	Usado como variável de retorno de métodos
@Valid	Validação de dados de objetos em parâmetros

Spring – Annotations

Anotação API (request)	Objetivo
@RestController	Classes de recursos que fornecem endpoints
@GetMapping	Anota o método para responder como endpoint HTTP GET (select)
@PutMapping	Anota o método para responder como endpoint HTTP PUT (update completo)
@PostMapping	Anota o método para responder como endpoint HTTP POST (insert)
@DeleteMapping	Anota o método para responder como endpoint HTTP DELETE (delete)
@RequestMapping	Mapear URLs de acesso ao controller e aos métodos
@PathVariable	Extrair da URL um parâmetro incluído como path da URL
@RequestParam	Capturar um parâmetro de consulta (query param) GET
@Valid	Validação de dados de objetos em parâmetros

Controller

! HelloController

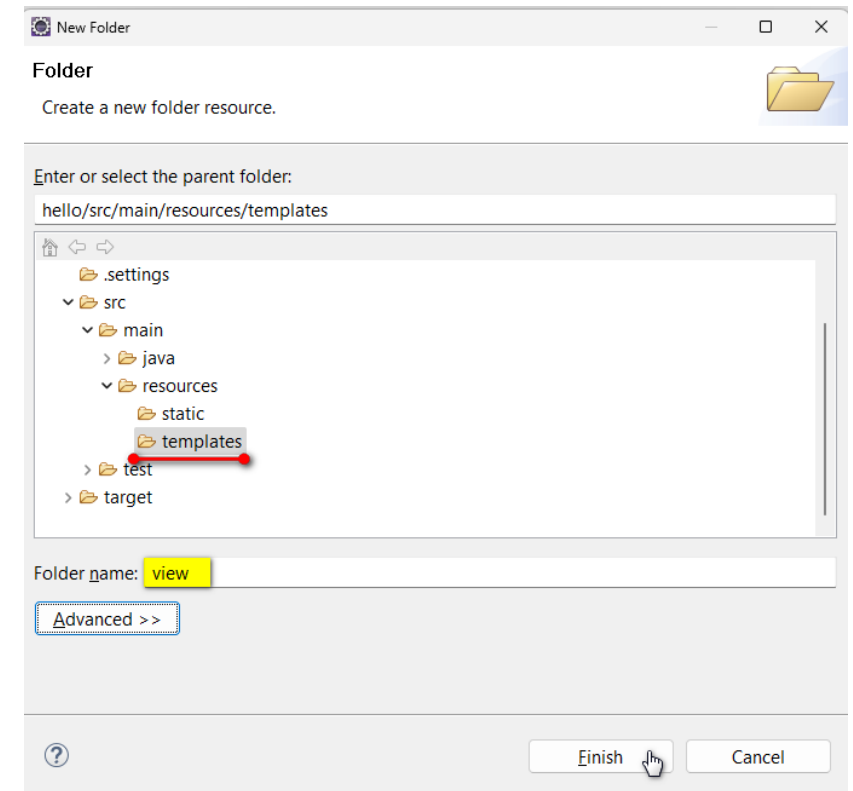
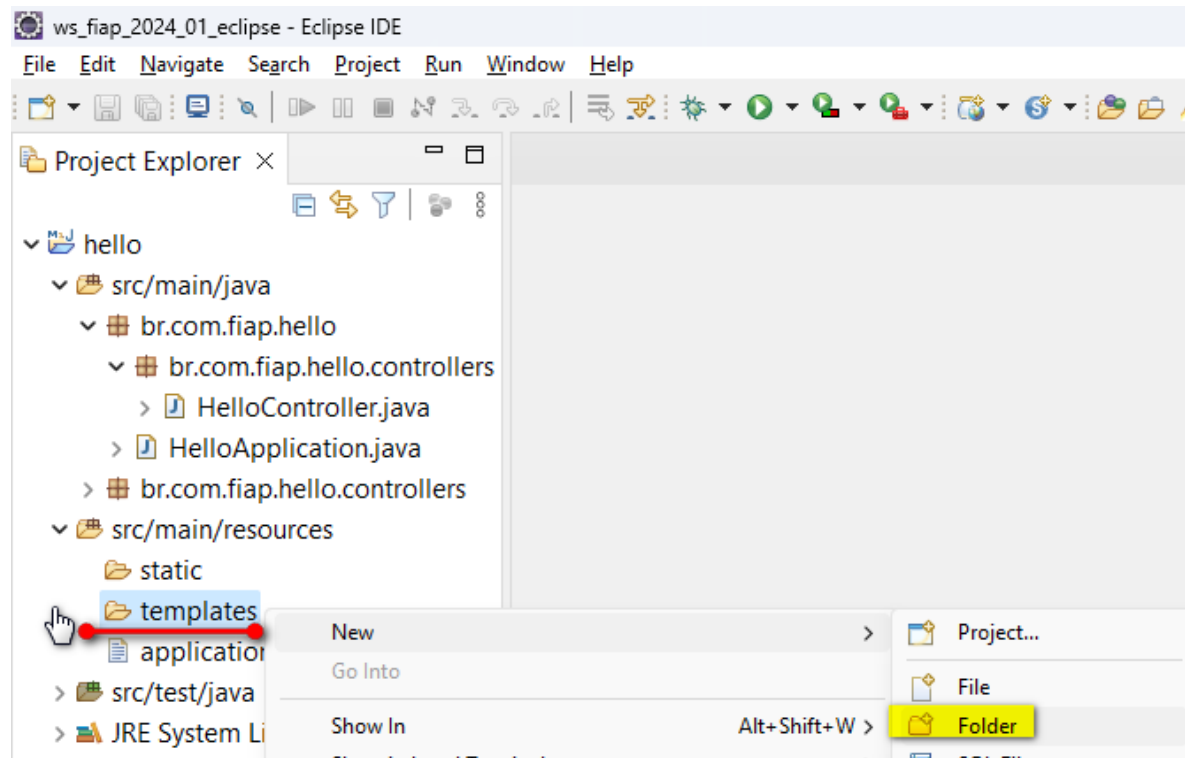


Class HelloController

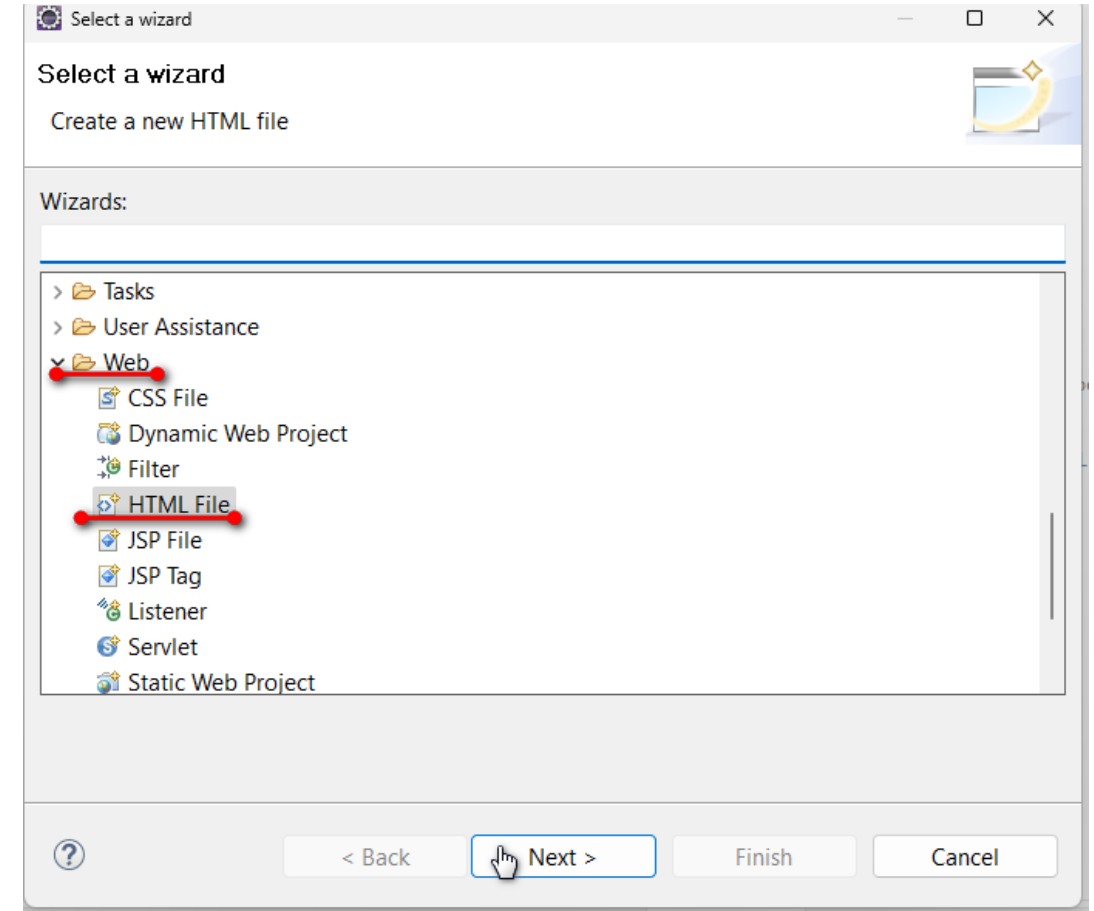
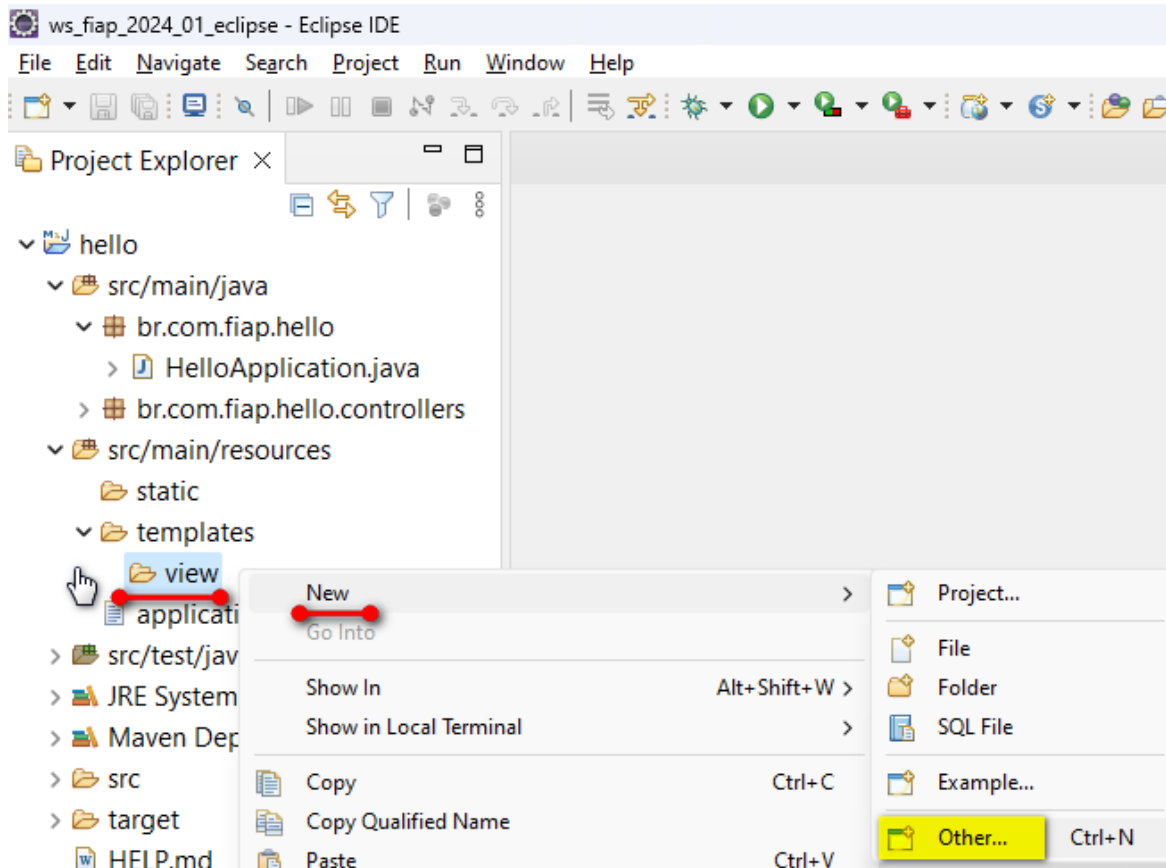
```
HelloController.java ×  
1 package br.com.fiap.hello.controllers;  
2  
3 import org.springframework.stereotype.Controller;  
4  
5 @Controller // define que essa classe é gerenciada pelo Spring  
6 public class HelloController {  
7  
8 }
```

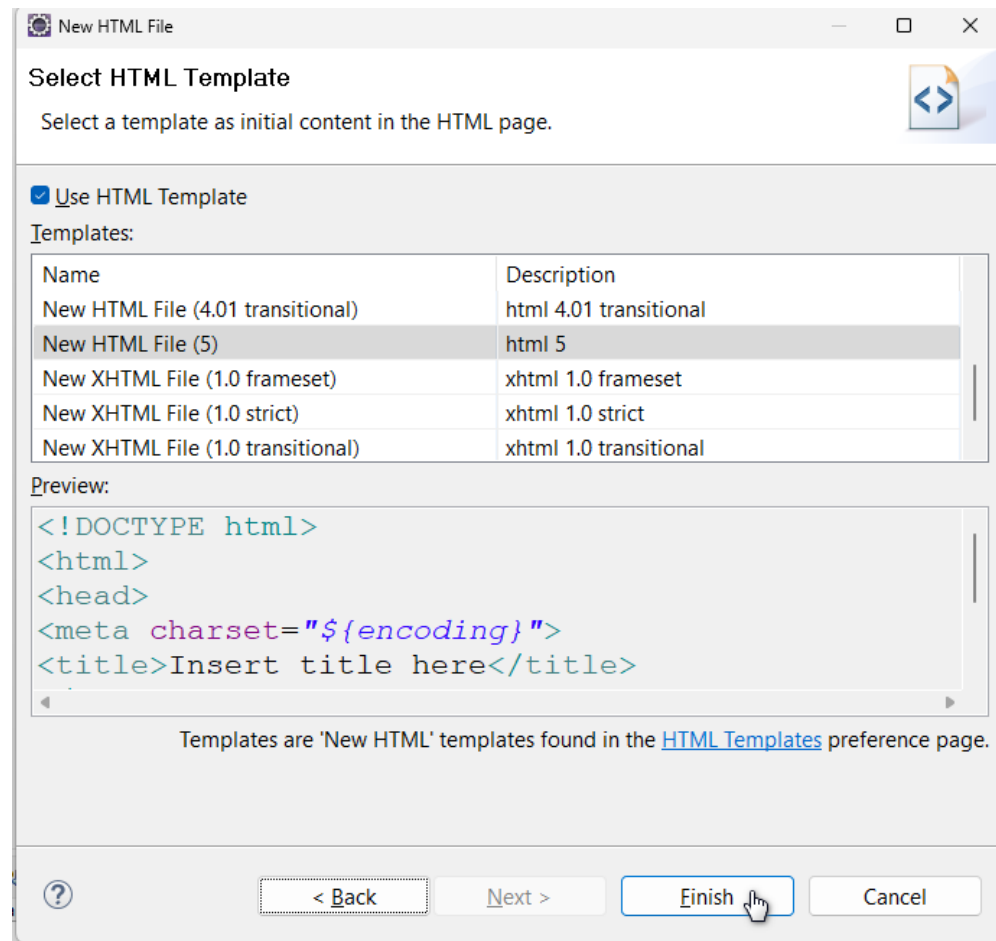
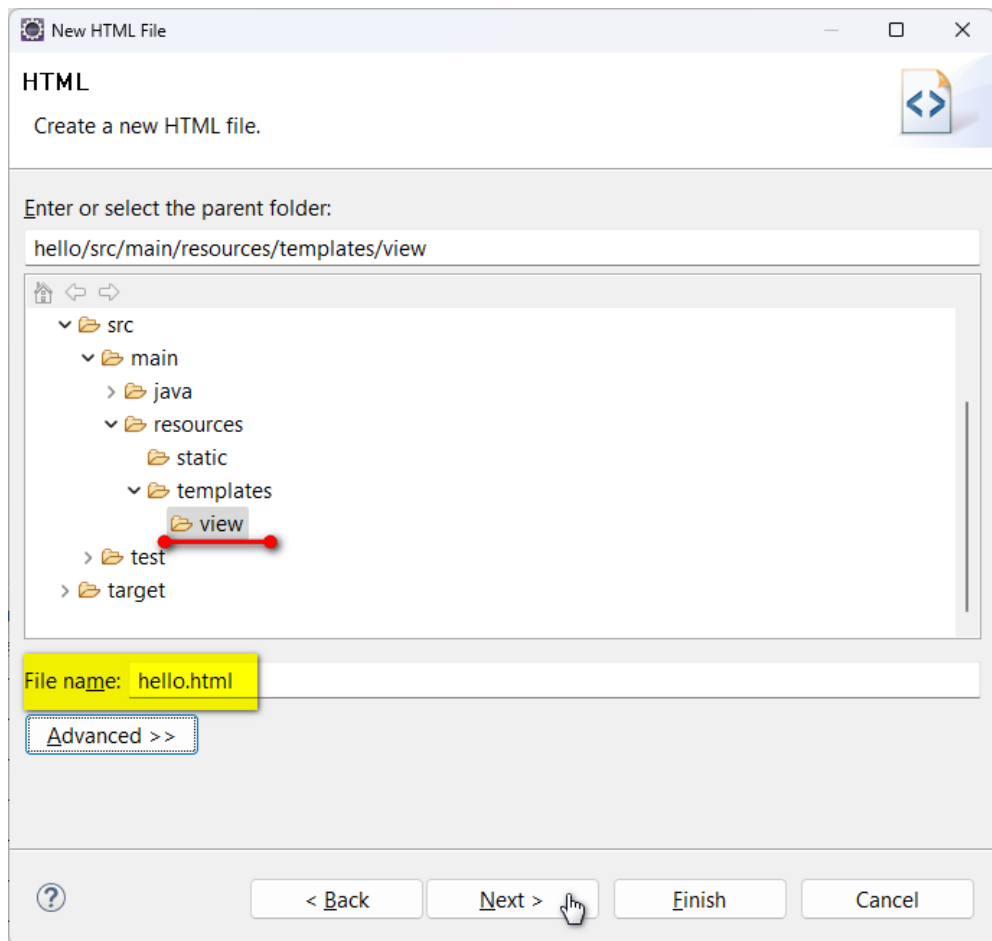
View

- Em aplicações Spring Boot os arquivos de *template* ficam dentro da pasta **src/main/resources/templates**.



View





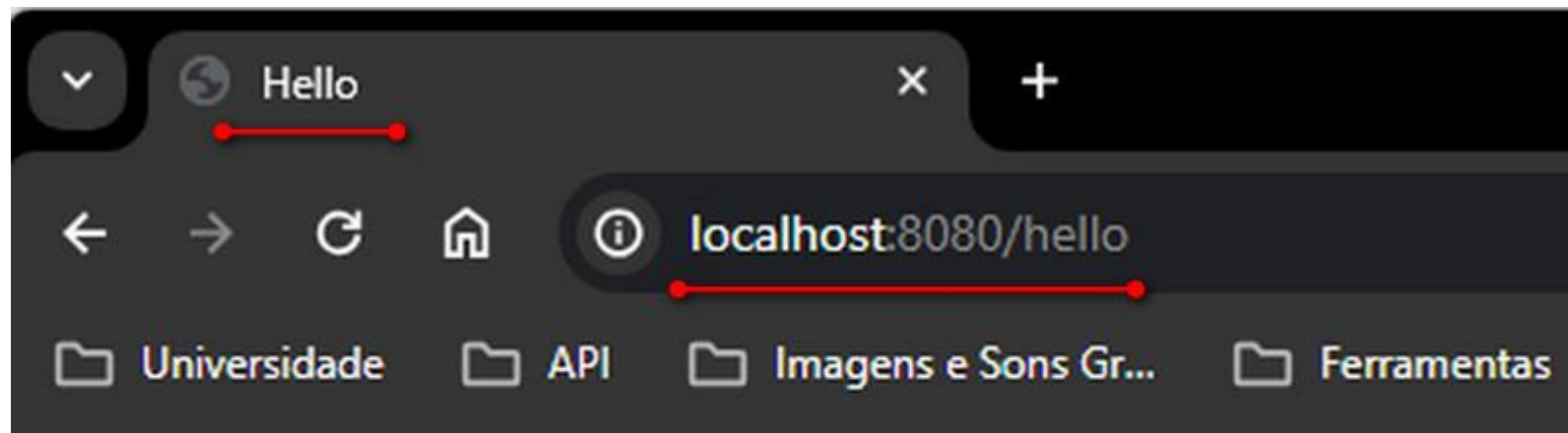
View - hello.html

```
hello.html x
1 <!DOCTYPE html>
2 <html lang="pt-br">
3
4 <head>
5     <meta charset="ISO-8859-1">
6     <title>Hello</title>
7 </head>
8
9 <body>
10
11     <h1>Hello, World</h1>
12     <p>Vamos de Spring MVC e Thymeleaf </p>
13
14 </body>
15
16 </html>
```

Alterar class HelloController

```
hello.html | HelloController.java x
1 package br.com.fiap.hello.controllers;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.RequestMapping;
6
7 @Controller // define que essa classe é gerenciada pelo Spring
8 @RequestMapping("/hello") // mapeando a URL da aplicação
9 public class HelloController {
10
11     @GetMapping()
12     public String mostrarMensagem() {
13         /*
14          * ação desejada
15          * devolve o caminho para View
16          * e a página HTML que será exibida
17          */
18         return "view/hello";
19     }
20
21 }
```

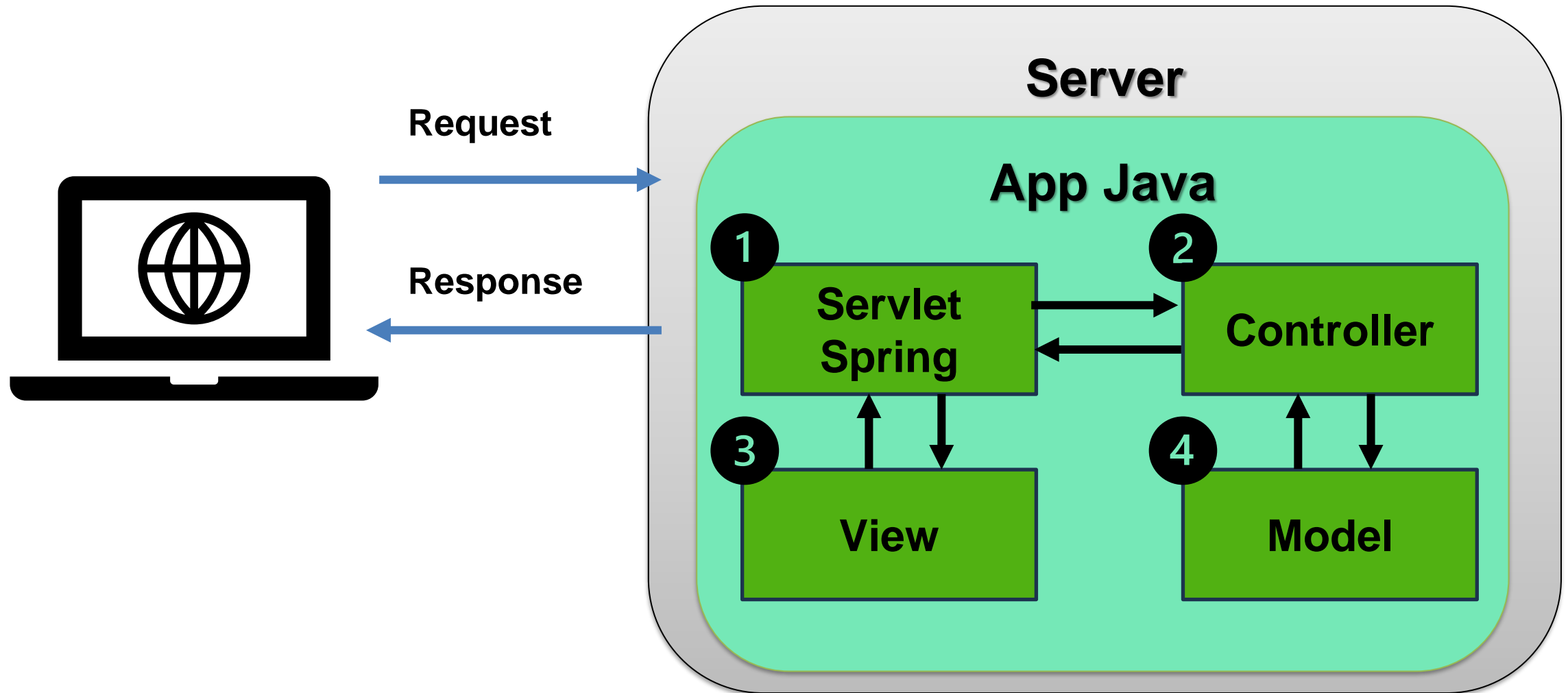
Executando a aplicação



Hello, World

Vamos de Spring MVC e Thymeleaf

Spring - Ciclo de uma Requisição HTTP



Thymeleaf

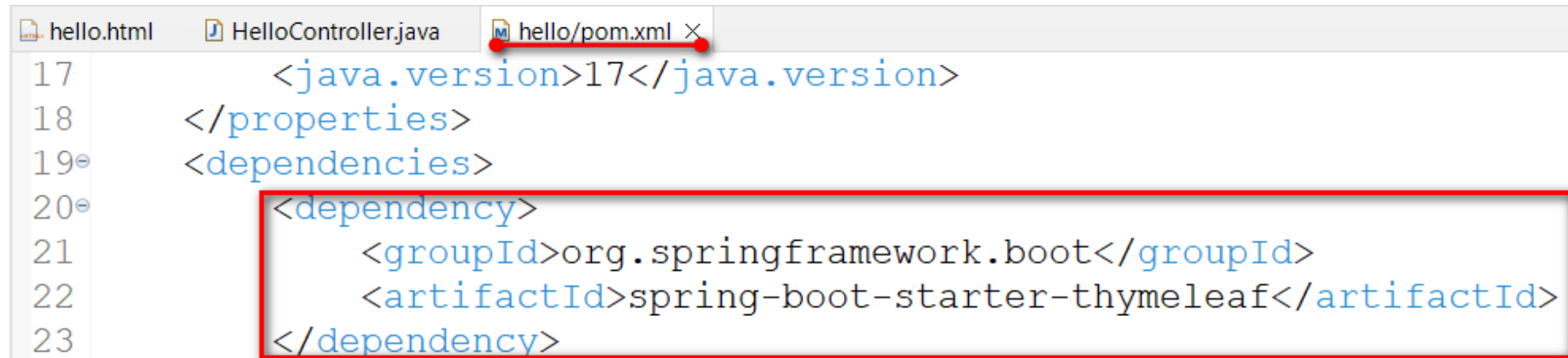
■ *Thymeleaf*

- O *Thymeleaf* é uma **template engine** para projetos Java que facilita a criação de páginas **HTML**.
- Ele serve para gerar páginas **HTML** no lado servidor de forma **dinâmica**, permitindo a troca de informações entre o código Java e as página HTML, de tal forma ele garante que o desenvolvedor consiga criar *templates* de forma mais fácil para suas aplicações.

<https://www.thymeleaf.org/>



- O **Thymeleaf** nos auxilia na integração entre a **Controller** e a **View**. Enviando informações do servidor para o cliente.



```
hello.html HelloController.java hello/pom.xml x
17     <java.version>17</java.version>
18 </properties>
19 <dependencies>
20 <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter-thymeleaf</artifactId>
23 </dependency>
```



Importando a biblioteca *Thymeleaf*

No arquivo hello.html



```
1 <!DOCTYPE html>
2 <html lang="pt-br" xmlns:th="http://thymeleaf.org">
3
4 <head>
5     <meta charset="ISO-8859-1">
6     <title>Hello</title>
7 </head>
8
9 <body>
10
11     <h1>Hello, World</h1>
12     <p>Vamos de Spring MVC e Thymeleaf </p>
13
14 </body>
15
16 </html>
```

th é o apelido definido para a variável
th abreviação de Thymeleaf

■ *Thymeleaf* – Expression Language

- Para referenciar uma variável com o *Thymeleaf*, utilizamos uma marcação especial, chamada **Expression Language - EL**
- Ela é representada pelo símbolo **`${}`**.
- Tudo o que estiver entre a abertura e fechamento de chaves, é interpretado pelo ***Thymeleaf*** como variável.

hello.html

hello.html x HelloController.java

```
1 <!DOCTYPE html>
2 <html lang="pt-br" xmlns:th="http://thymeleaf.org">
3
4 <head>
5     <meta charset="ISO-8859-1">
6     <title>Hello</title>
7 </head>
8
9 <body>
10
11     <h1>Hello, World</h1>
12
13     <h2 th:text="${mensagem}">Thymeleaf</h2>
14
15     <p>Vamos de Spring MVC e Thymeleaf </p>
16
17 </body>
18
19 </html>
20
```

xmlns:th="http://thymeleaf.org"

th - apelido definido para a variável.
th - abreviação de Thymeleaf

<h2 th:text="\${mensagem}">Thymeleaf</h2>

EL - Expression Language

Thymeleaf - tags

O `th` utilizado é do **Thymeleaf**, ele é um tipo de variável utilizado para receber informações da controller.

`th:each` : para fazer interações com objetos enviados pela *controller*.

`th:text` : para exibir informação.

Customizando a Aplicação

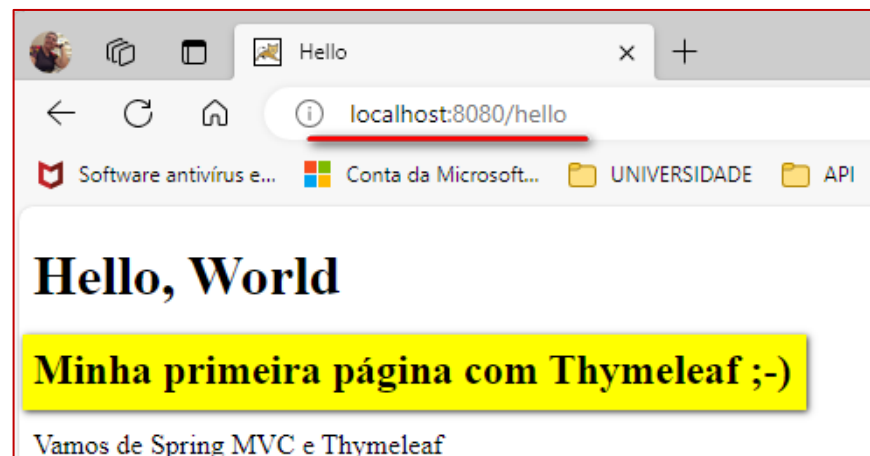
Alterar class HelloController

```
hello.html HelloController.java X
1 package br.com.fiap.hello.controllers;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.ui.Model;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestMapping;
7
8 @Controller // define que essa classe é gerenciada pelo Spring
9 @RequestMapping("/hello") // mapeando a URL da aplicação
10 public class HelloController {
11
12     @GetMapping()
13     public String mostrarMensagem(Model model) {
14
15         /*
16          model adiciona um atributo na página nomeado mensagem, e o objeto
17          relacionado que neste caso é a string.
18          Assim, enviamos uma informação para a página.
19          O controlador é quem envia a variável para a página,
20          e não a página que acessa informação no controlador.
21          */
22
23         model.addAttribute("mensagem", "Minha primeira página com Thymeleaf ;-) ");
24
25         /*
26          ação desejada
27          devolve o caminho para View
28          e a página HTML que será exibida
29          */
30         return "view/hello";
31     }
32 }
```

model.addAttribute

Usado como retorno do método

Executando a Aplicação



class HelloController

```
model.addAttribute("mensagem", "Minha primeira página com Thymeleaf ;-) ");
```

hello.html

```
<h2 th:text="${mensagem}">Thymeleaf</h2>
```

Alterar classe HelloController

```
hello.html *HelloController.java x
1 package br.com.fiap.hello.controllers;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.ui.Model;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RequestParam;
8
9 @Controller
10 @RequestMapping("/hello")
11 public class HelloController {
12
13     @GetMapping()
14     public String mostrarMensagem(@RequestParam(name="nome",
15                                     required=false,
16                                     defaultValue="FIAP")
17                                   String nome, Model model) {
18
19
20         model.addAttribute("nome", nome);
21
22
23         return "view/hello";
24     }
25 }
```

@RequestParam

Capturar um parâmetro de consulta (query param) GET

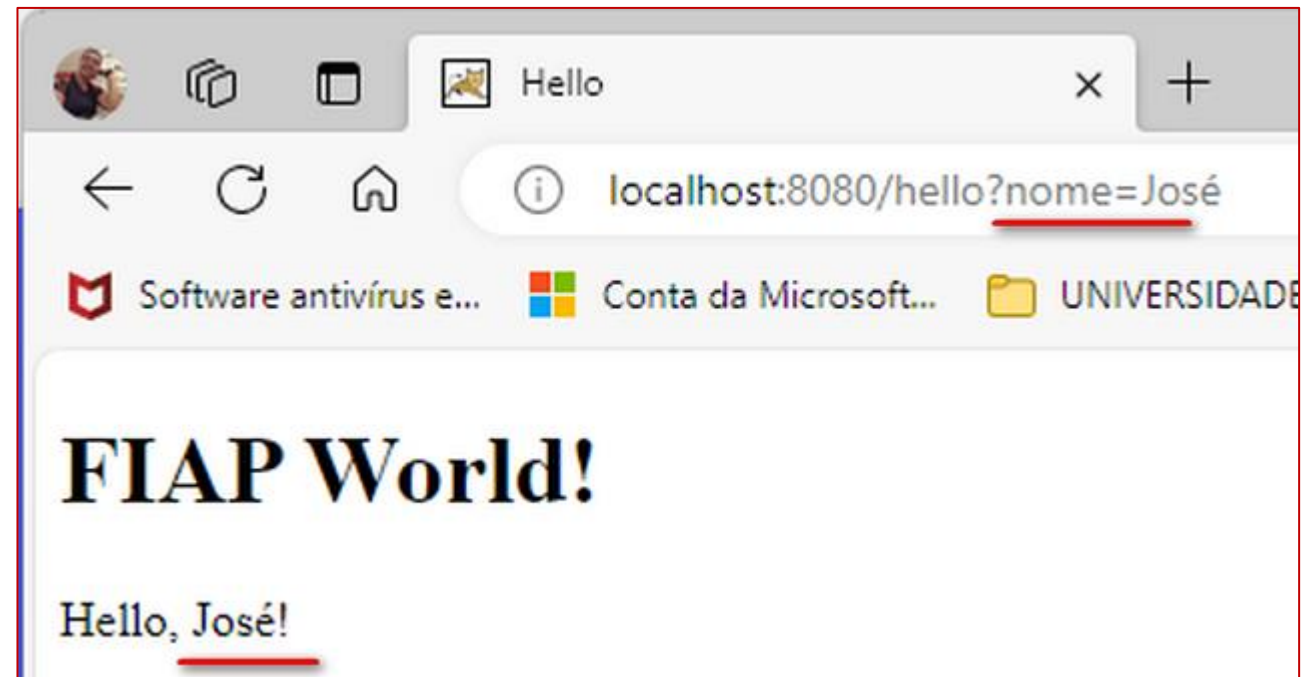
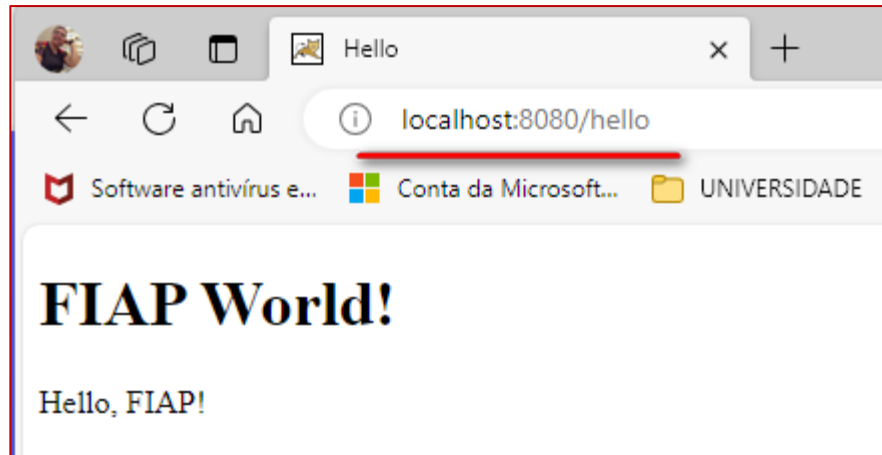
Alterar hello.html

```
hello.html x HelloController.java
1 <!DOCTYPE html>
2 <html lang="pt-br" xmlns:th="http://thymeleaf.org">
3
4 <head>
5     <meta charset="ISO-8859-1">
6     <title>Hello</title>
7 </head>
8
9 <body>
10
11     <h1>FIAP World!</h1>
12
13     <p th:text="|Hello,  ${nome} |"></p>
14
15     <!-- OU
16         <p th:text=" ${nome} ">Nome</p>
17
18     -->
19
20 </body>
21
22 </html>
23
```

■ Parâmetro na URL

- Passar o nome como parâmetro na query da URL
`http://localhost:8080/hello?nome=Cida`
- O valor do parâmetro ***nome*** sobrescreve (*override*) o valor **default FIAP** e é refletido na resposta pelo conteúdo alterado.

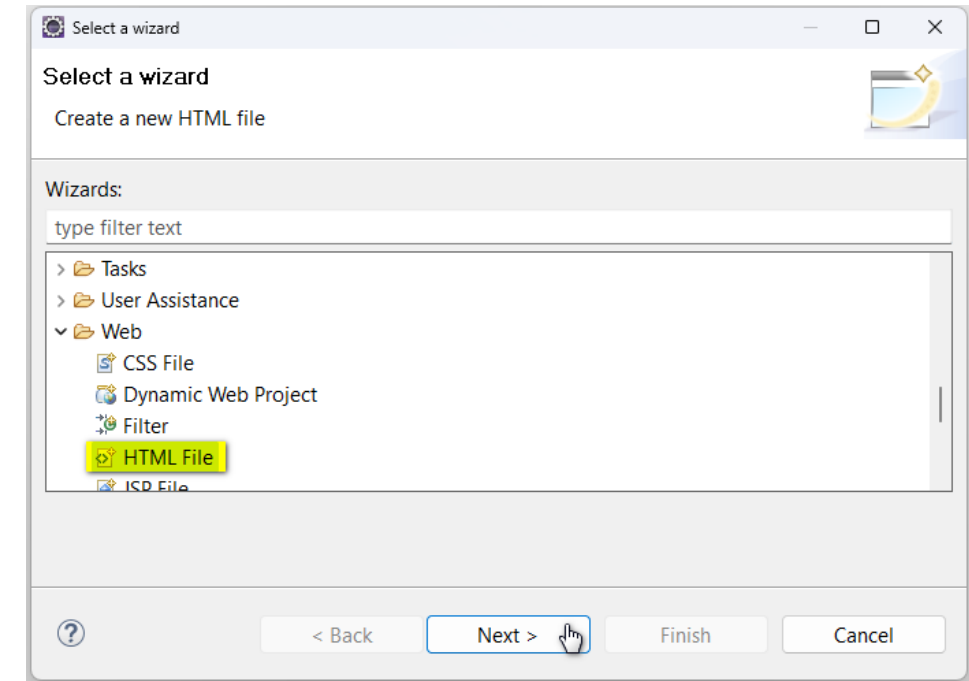
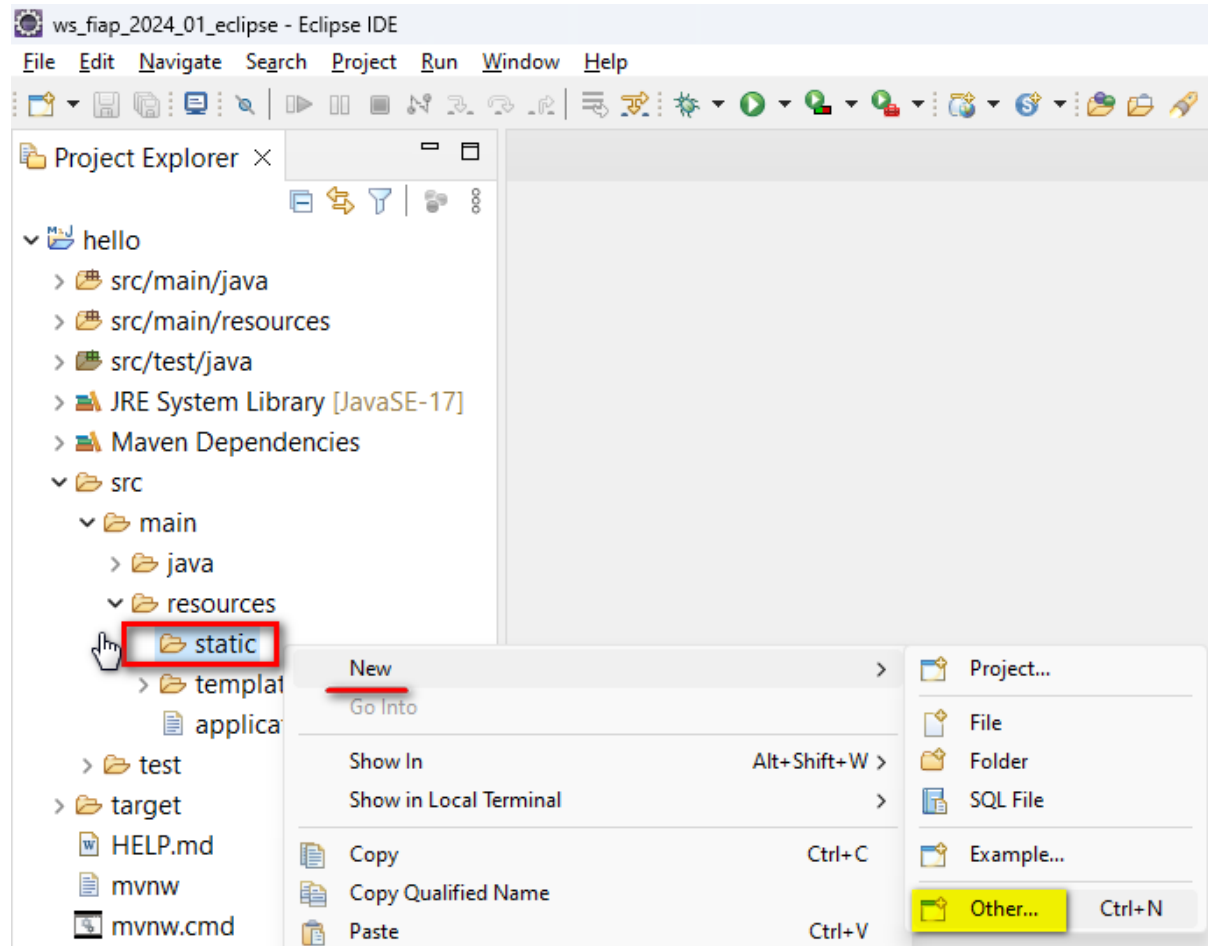
Executando a Aplicação



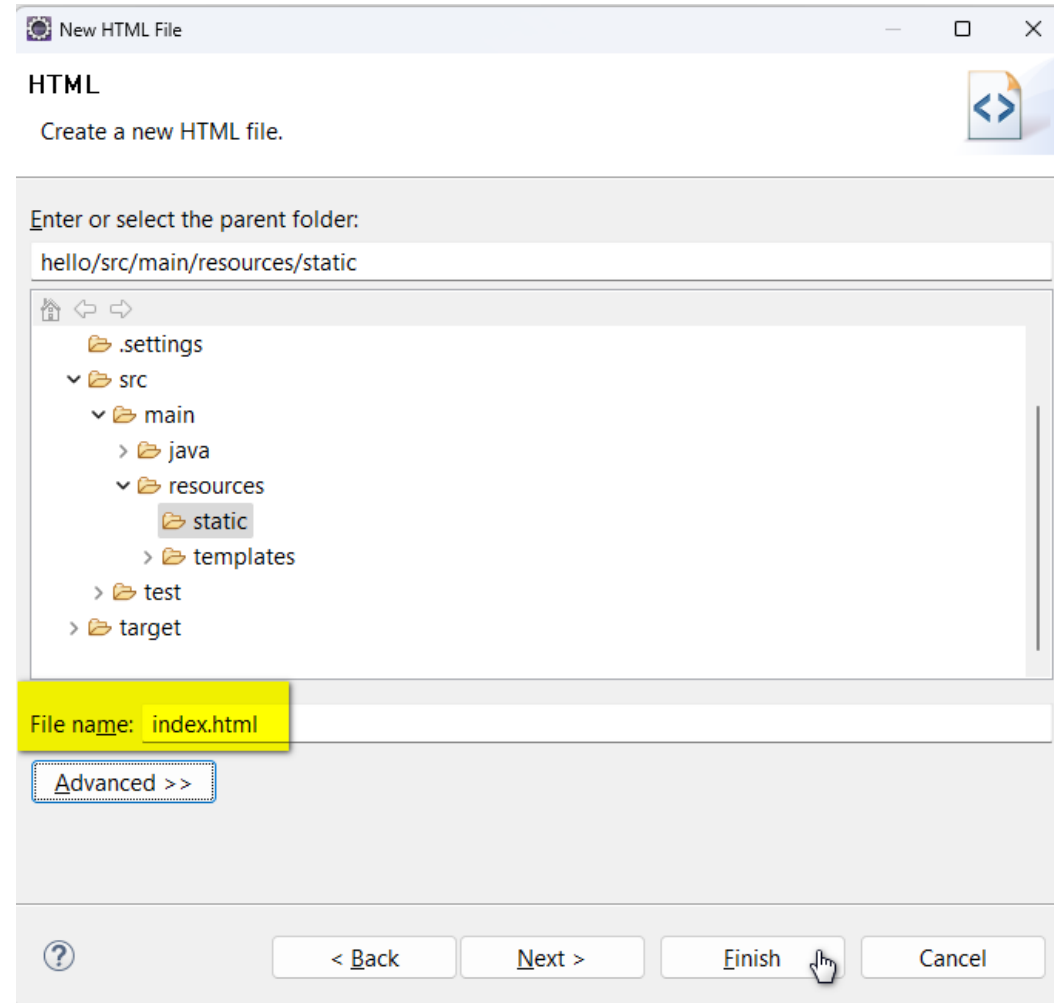
I Adicionando a Home Page

- Podemos utilizar recursos estáticos, incluindo HTML, JavaScript e CSS, na aplicação Spring Boot, e devemos coloca-los na pasta ***static***.
- Por padrão, o Spring Boot fornece conteúdo estático de recursos no ***classpath /static*** (ou ***/public***).
- O recurso ***index.html*** é especial porque, se existir, é usado como uma "página de boas-vindas" ou página inicial, o que significa que é servido como recurso raiz, ou seja, em ***http://localhost:8080/***.

Adicionando a Home Page – index.html



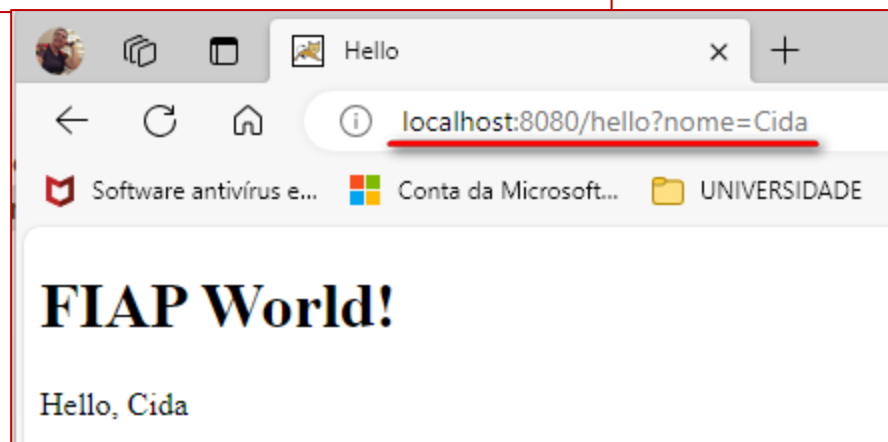
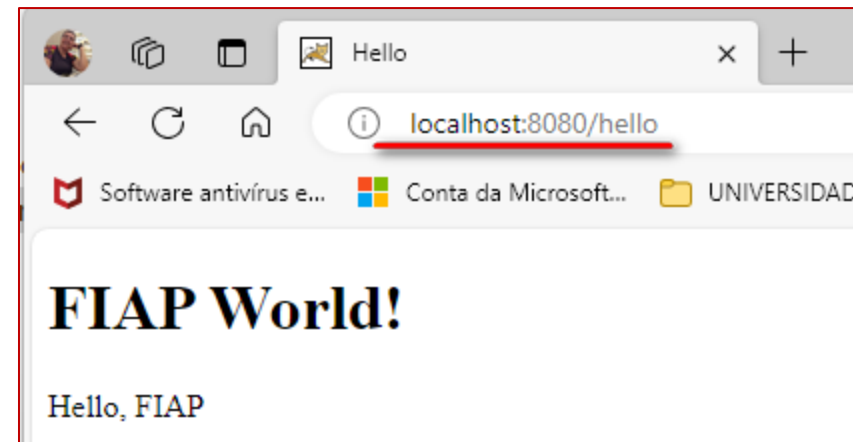
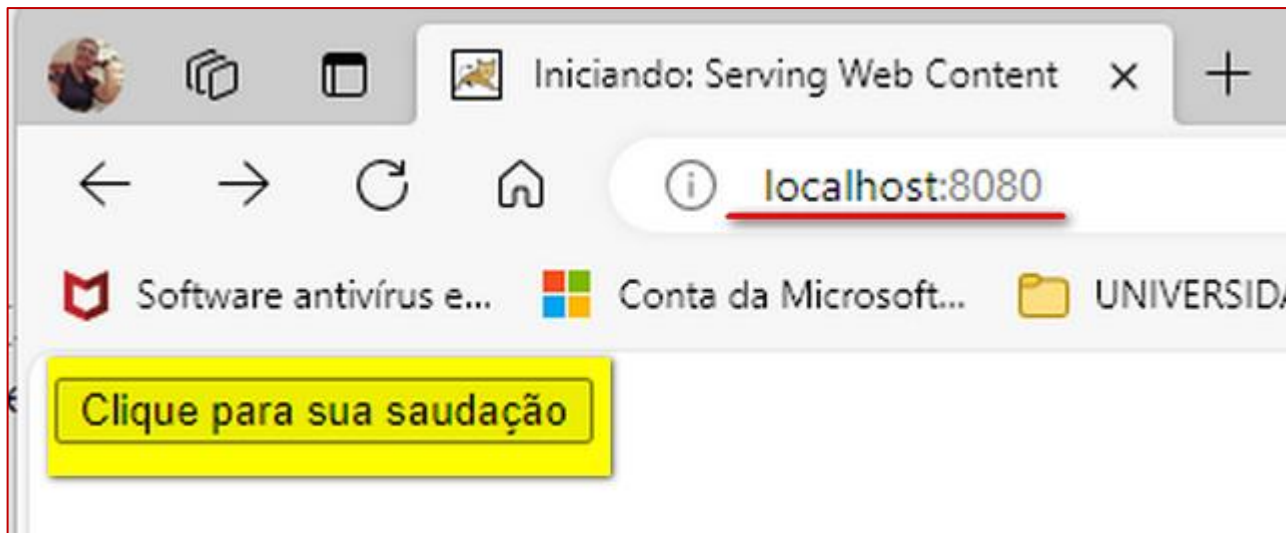
Adicionando a Home Page – index.html



Home Page – index.html

```
*index.html x
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5     <meta charset="UTF-8">
6     <title>Iniciando: Serving Web Content</title>
7 </head>
8
9 <body>
10
11     <a href="/hello"><button>Clique para sua saudação</button></a>
12
13 </body>
14
15 </html>
```

Executar a Aplicação





Copyright © 2024

Prof^a. Aparecida de Fátima Castello Rosa

Copyright © 2020

Prof^o. Pedro Ivo Correia e Prof^o. Lucas Furlaneto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).