# PARTE 1

Agora é com você! Modifique sua aplicação de linha de comando para web e crie seu primeiro controlador com a rota localhost:8080/series. Para isso, você precisará fazer algumas alterações no pom.xml e criar uma nova classe Controller.

VER OPINIÃO DO INSTRUTOR

### Opinião do instrutor

No nosso pom.xml, precisaremos adicionar a dependência starter-web, deixando o arquivo da seguinte forma:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.1</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>br.com.alura</groupId>

<artifactId>screenmatch</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>screenmatch</name>

<description>Primeiro projeto Spring sem web</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-web</artifactId>**

**</dependency>**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.15.2</version>

</dependency>

<dependency>

<groupId>com.theokanning.openai-gpt3-java</groupId>

<artifactId>service</artifactId>

<version>0.14.0</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

COPIAR CÓDIGO

Além disso, para mapear nossa primeira rota HTTP, precisaremos de um controlador. Iremos seguir as boas práticas e criar um pacote exclusivo para os controladores, chamado controller. Nele, iremos criar a classe SerieController. Ela deve ser anotada com @RestController, que permite que a classe seja gerenciada pelo Spring e que ele reconheça que ela é um controlador.

Ainda no controller, criaremos o método obterSeries(), que deve ser anotado com @GetMapping(“/series”), indicando para qual url nossa aplicação está mapeada. Esse método retornará, por enquanto, uma String: "Aqui vão ser listadas as séries". Dessa forma, a classe controller fica assim:

**package** br.com.alura.screenmatch.controller;

**import** org.springframework.web.bind.**annotation**.GetMapping;

**import** org.springframework.web.bind.**annotation**.RestController;

@RestController

**public** **class** **SerieController** {

@GetMapping("/series")

**public** String obterSeries() {

**return** "Aqui vão ser listadas as séries";

}

}

COPIAR CÓDIGO

Por fim, para que a aplicação realmente rode na web, nossa classe ScreenmatchApplication deve ser modificada. Podemos duplicá-la, chamar a antiga classe de ScreenmatchApplicationSemWeb, e comentar o código dela. Iremos modificar a nova classe. Vamos retirar todas as referências à linha de comando, como a implementação do CommandLineRunner, a sobrescrita do método run() e o uso do repository. Nossa classe ScreenmatchApplication fica assim:

package br.com.alura.screenmatch;

**import** br.com.alura.screenmatch.principal.Principal;

**import** br.com.alura.screenmatch.repository.SerieRepository;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** **ScreenmatchApplication** {

**public** **static** void **main**(String[] args) {

**SpringApplication**.**run**(**ScreenmatchApplication**.class, args);

}

}

COPIAR CÓDIGO

Feito isso, realizamos o primeiro passo e já conseguimos visualizar algo acontecendo no navegador! Os próximos passos são visualizar as séries tanto em formato de JSON quanto como imagens no front-end. Vamos lá? :)