

AULA 08 - SPRING WEB

Disciplina de Backend - Professor Ramon Venson - SATC 2024



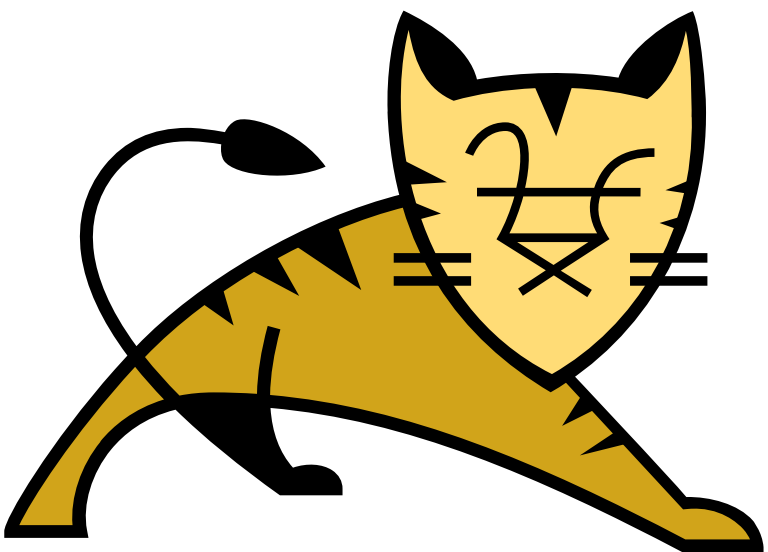
Spring Web

- Parte integrante do Spring Framework
- Recursos para lidar com solicitações HTTP e gerenciar controladores.
- Pode manipular também visualizações.

Spring Boot e Spring Initializr

- O Spring Boot é um framework opinativo para construir aplicações Spring em produção com configuração mínima.
- O Spring Initializr é uma ferramenta baseada na web para gerar projetos Spring Boot com dependências e configurações personalizadas.
- Ele simplifica o processo de inicialização de um projeto Spring Boot, fornecendo uma interface amigável para selecionar dependências e gerar modelos de projeto.





Servidor de Aplicação

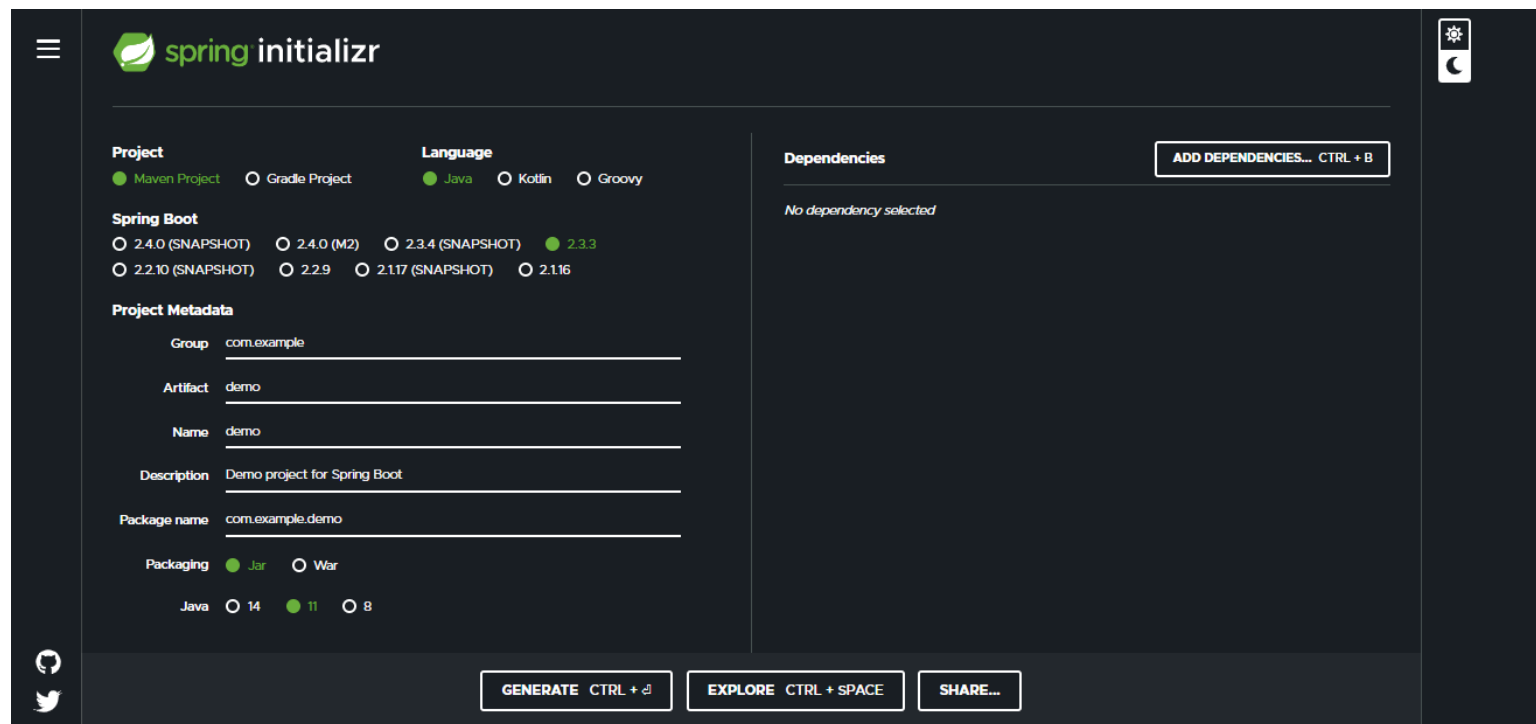
- Ambiente de execução para aplicações Web.
- Gerencia solicitações e coloca as aplicações em contato com a camada TCP/IP.
- Por padrão, o Spring Boot usa o Tomcat, um servidor de aplicação de código aberto e leve.

Como criar a primeira rota para minha aplicação Spring Web

1. Crie um novo projeto Spring Boot usando o Spring Initializr.
2. Defina uma classe de controlador com a anotação `@RestController`.
3. Defina um método de manipulador dentro da classe do controlador para lidar com solicitações HTTP.
4. Use `@GetMapping("/")` ou outras anotações de mapeamento para especificar o mapeamento de URL para o método de manipulador.
5. Execute a aplicação e acesse a rota definida em um navegador da web.

Criando um novo projeto

- Acesse o site [Spring Initializr](#)
- Configuração padrão:
 - Projeto: Maven
 - Spring Boot: 3+
 - Packaging: Jar
 - Java: 17



The screenshot shows the Spring Initializr web application interface. The header includes the Spring logo and the text "spring initializr". The main content area is divided into several sections:

- Project:** Includes radio buttons for "Maven Project" (selected) and "Gradle Project".
- Language:** Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Includes radio buttons for various versions: "2.4.0 (SNAPSHOT)", "2.4.0 (M2)", "2.3.4 (SNAPSHOT)", "2.3.3" (selected), "2.2.10 (SNAPSHOT)", "2.2.9", "2.1.17 (SNAPSHOT)", and "2.1.16".
- Project Metadata:** Includes input fields for "Group" (com.example), "Artifact" (demo), "Name" (demo), "Description" (Demo project for Spring Boot), and "Package name" (com.example.demo).
- Packaging:** Includes radio buttons for "Jar" (selected) and "War".
- Java:** Includes radio buttons for "14", "17" (selected), and "8".
- Dependencies:** Includes a button "ADD DEPENDENCIES... CTRL + B" and the text "No dependency selected".

At the bottom, there are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...".

Rodando e configurando porta de rede

- Descompacte a pasta `*.zip` baixada e abra o projeto
- Adicione a seguinte linha ao arquivo `applications.properties` para mudar a porta da aplicação

```
server.port = 8080 //8080 é a porta padrão
```

Criando um Controller

Crie uma nova classe e adicione um método para criar um novo mapeamento de rota.

HelloWorldController.java

```
@RestController
public class HelloWorldController {
    @GetMapping("/")
    public String olaMundo() {
        return "Olá mundo";
    }
}
```

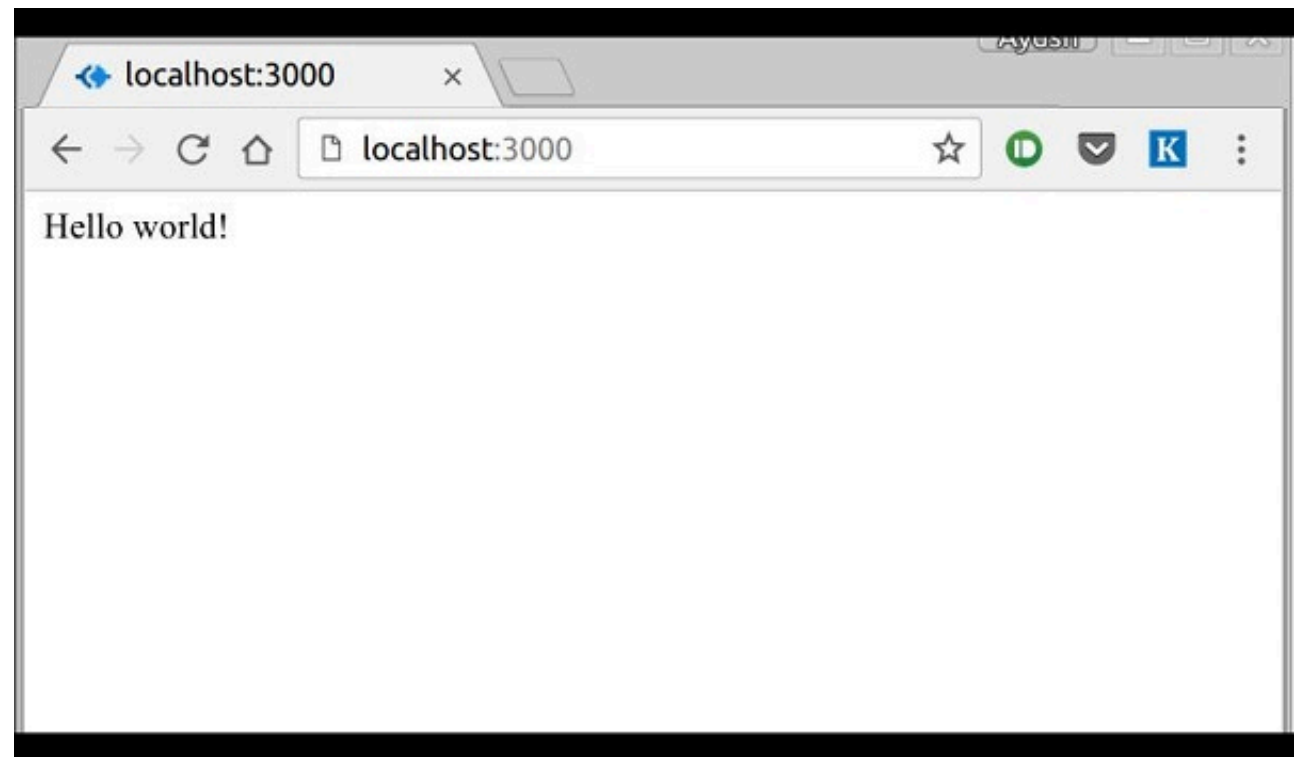
Não esqueça das anotações @RestController e @GetMapping("/"). Você pode substituir o `/` por qualquer outro nome.

Acessando a rota

Após rodar sua aplicação e utilizando um navegador ou qualquer outro cliente HTTP, realize a requisição para o endereço:

```
http://localhost:8080/
```

Lembre-se de substituir a porta `8080` caso tenha configurado outra e adicionar a rota definida no `@GetMapping("/")`



Criando um Service

O Controller geralmente é responsável apenas por receber e validar requisições, assim como enviar repostas. Para padronizar nossa rota, vamos adicionar um novo service para transferir as responsabilidades de regra de negócios para a sua devida camada:

HelloWorldService.java

```
@Service
public class HelloWorldService {
    public String gerarOlaMundo() {
        return "Ola, mundo!"
    }
}
```

Agora vamos refatorar o nosso `HelloWorldController.java`:

```
@RestController
public class HelloWorldController {
    @Autowired
    private HelloWorldService helloWorldService;

    @GetMapping("/")
    public String olaMundo() {
        return helloWorldService.gerarOlaMundo();
    }
}
```

Não esqueça de criar um atributo na classe controller e usar a anotação `@Autowired` para que o Spring gerencie automaticamente essa dependência.

Explorando requisições

O Spring Web empacota e gerencia os dados de cada uma das requisições que nosso web service recebe. Para acessar alguns destes dados, podemos utilizar:

- `@RequestParam` para ler parâmetros de query
- `@RequestBody` para ler o corpo de uma requisição
- `@RequestHeader` para ler cabeçalhos de uma requisição
- `@PathVariable` para ler variáveis de caminho da requisição

@RequestParam

Essa anotação extrai o conteúdo de uma query string da URL, como por exemplo:

`https://localhost:8080/?nome=Faustao`

```
@GetMapping("/")  
public String olaMundo(@RequestParam nome) {  
    return helloWorldService.gerarOlaMundo() + nome; // Olá, Mundo! Faustao  
}
```

@RequestBody

Essa anotação extrai o conteúdo do corpo da requisição.

```
@GetMapping("/")  
public String olaMundo(@RequestBody nome) {  
    return helloWorldService.gerarOlaMundo() + nome;  
}
```

@RequestHeaders

Essa anotação extrai o conteúdo dos cabeçalhos da requisição.

```
@GetMapping("/")  
public String olaMundo(@RequestHeaders nome) {  
    return helloWorldService.gerarOlaMundo() + nome;  
}
```

@PathVariable

Essa anotação extrai o conteúdo de uma variável definida no caminho da requisição, como por exemplo: `https://localhost:8080/galvao`

```
@GetMapping("/{nome}")  
public String olaMundo(@PathVariable("nome") nome) {  
    return helloWorldService.gerarOlaMundo() + nome; // Ola, Mundo! Galvao  
}
```


Lendo variáveis de ambiente

Podemos fazer com que o Spring injete automaticamente variáveis de ambiente dentro de variáveis do nosso programa. Para isso usamos a anotação `@Value()`.

```
@RestController
public class HelloWorldController {
    @Value("${app.nome}")
    private String nome;

    @GetMapping("/")
    public String helloWorld() {
        return helloWorldService.gerarOlaMundo() + nome; // Ola, Mundo! MeuApp
    }
}
```

Podemos usar o arquivo `application.properties` para definir essas variáveis de ambiente num arquivo, automaticamente:

```
spring.application.name=demo  
server.port = 8000  
app.nome = "MeuApp"
```

O que aprendemos hoje

- O Spring Web é uma parte do Spring Framework para construção de aplicações web.
- O Spring Boot e o Spring Initializr simplificam o processo de criação de projetos Spring Boot.
- Aprendemos como criar a primeira rota para uma aplicação Spring Web usando controladores e mapeamentos de solicitação.