

TÓPICO 08 - SPRING WEB

Disciplina de Backend - Professor Ramon Venson - SATC 2025.2



Spring Framework

O Spring Framework é um framework modular de aplicações na plataforma Java que facilita o desenvolvimento de aplicações empresariais.

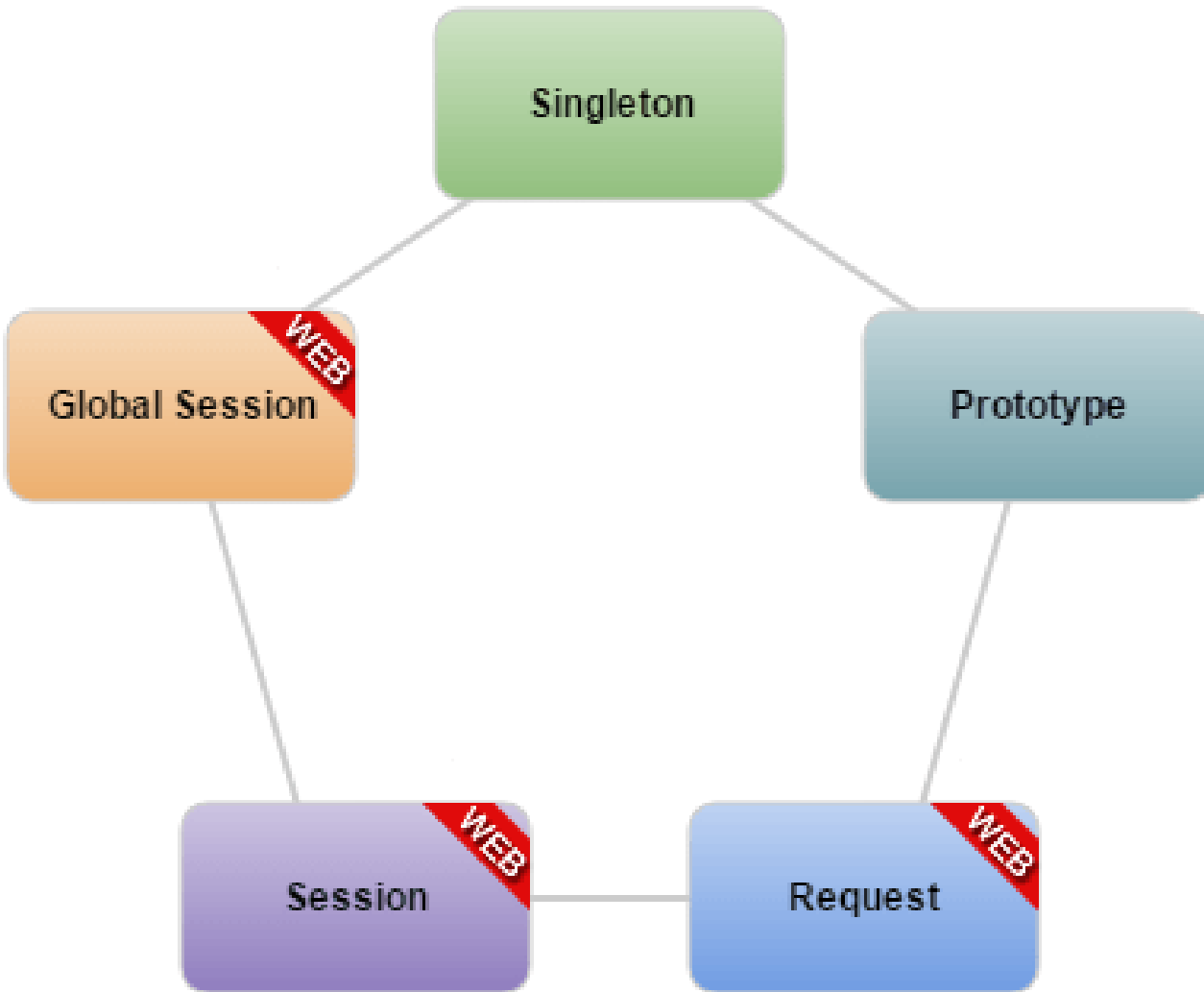
Ele utiliza o padrão de projeto Inversion of Control (IoC) para facilitar a criação e controle de aplicações.

História do Spring

Foi desenvolvido por Rod Johnson, em 2003, e lançado em 2004.

Surgiu como substituto para as especificações J2EE (javax), e hoje implementa e estende as especificações Jakarta.





Spring Beans

Um Spring Bean é um objeto gerenciado pelo Spring Container.

Essa é a unidade básica do Spring Framework, que representa entidades de dados, serviços e controladores.



Spring Web

O Spring Web é um modulo do Spring Framework que fornece recursos para criar aplicativos da web.

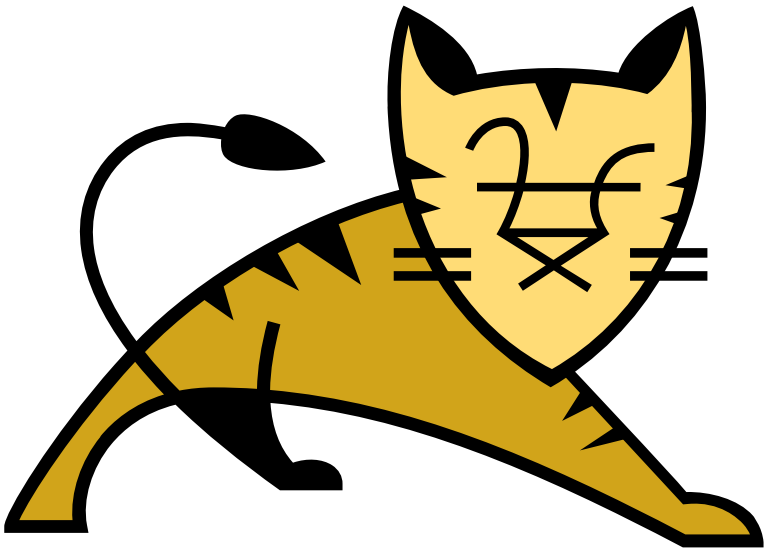
Sem ele seria necessário implementar e integrar um conjunto de bibliotecas e frameworks para lidar com solicitações HTTP, gerenciar controladores e visualizações.

Spring Boot e Spring Initializr

Existem também dois módulos importantes relacionados ao Spring:

- 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.





Servidor de Aplicação

Para executar uma aplicação Spring Web, é necessário um servidor de aplicação.

Usando o Spring Boot, ele já vem com um servidor de aplicação integrado, o `Tomcat`.

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 ou 21

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";
    }
}
```

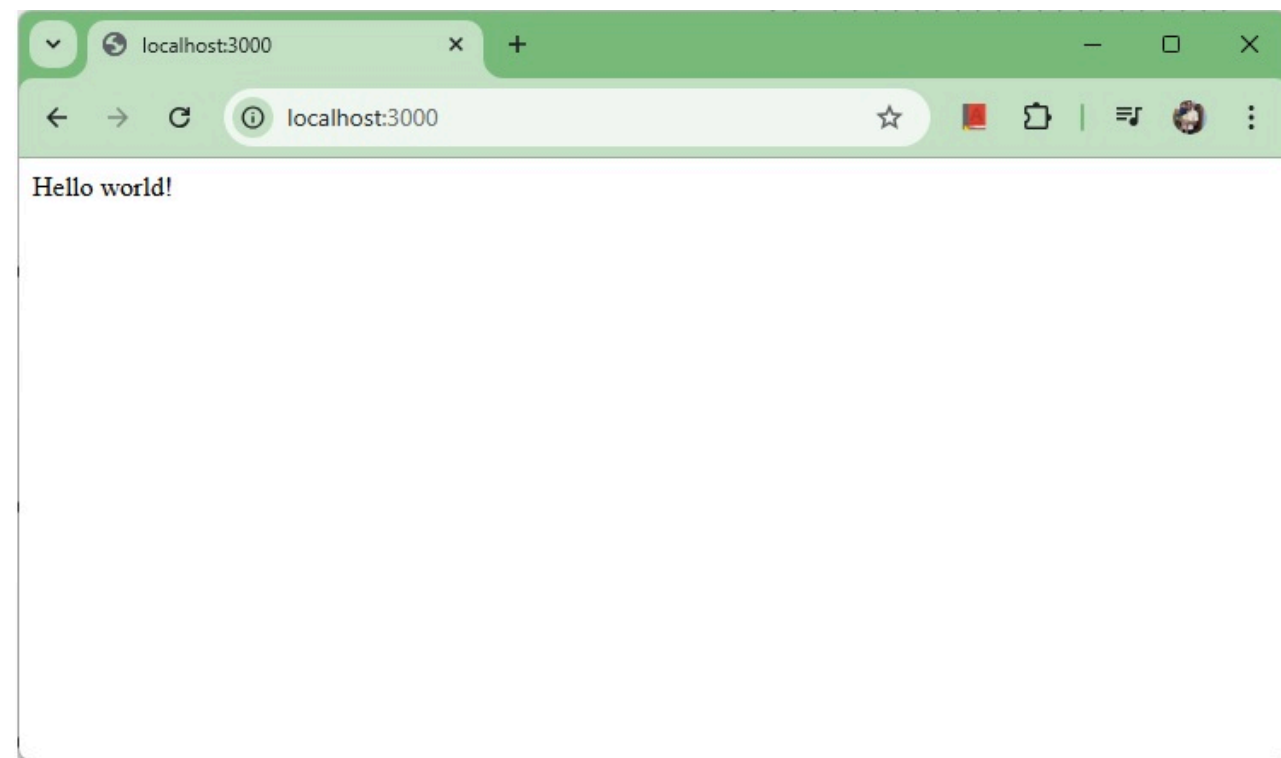
Não esqueça das anotações `@RestController` e `@GetMapping("/")`. Você pode substituir o `/` por qualquer outro caminho.

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 String nome) {  
    return "Olá, " + nome; // Olá, Faustao  
}
```


@RequestBody

Essa anotação extrai o conteúdo do corpo da requisição.

```
@GetMapping("/")  
public String olaMundo(@RequestBody nome) {  
    return "Olá, " + nome;  
}
```

@RequestHeaders

Essa anotação extrai o conteúdo dos cabeçalhos da requisição.

```
@GetMapping("/")  
public String olaMundo(@RequestHeaders nome) {  
    return "Olá, " + 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 String 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"
```