

# Linguagem de Programação Orientada a Objetos I

Spring

Prof. Tales Bitelo Viegas

<https://fb.com/ProfessorTalesViegas>

# Introdução

- ▶ Spring é um conjunto de projetos que resolvem várias situações do cotidiano de um programador, ajudando a criar aplicações Java com simplicidade e flexibilidade
- ▶ Surgiu como uma alternativa ao Java EE, e seus criadores se preocuparam para que ele fosse o mais simples e leve possível.
- ▶ Todos os projetos são Open Source, com código-fonte no Github

# Spring x Java EE

- ▶ Spring não é 100% concorrente do Java EE, visto que usa tecnologias que estão dentro da especificação.
- ▶ Entretanto, como não faz parte da especificação, novos projetos são lançados e testados muito mais rapidamente

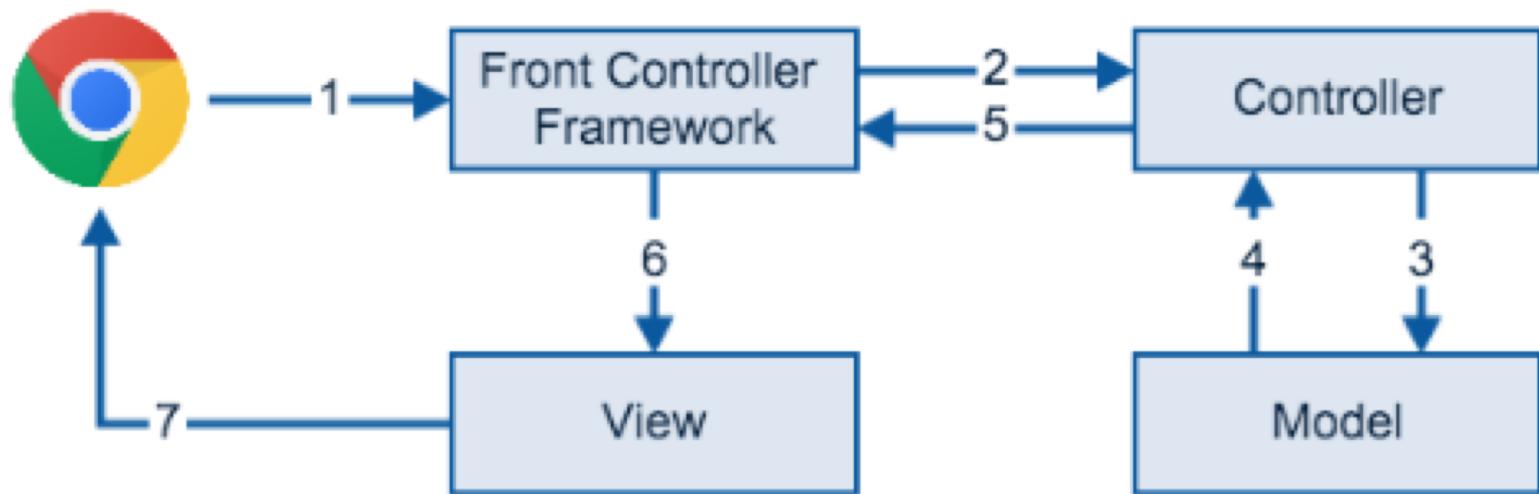
# Spring Boot

- ▶ Responsável pela análise da configuração do código.
- ▶ Analisa o código-fonte criado e configura todo o ambiente necessário automaticamente
- ▶ <https://projects.spring.io/spring-boot/>

# Spring MVC

- ▶ Framework que auxilia no desenvolvimento de aplicações web robustas
- ▶ <https://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html>
- ▶ Model–View–Controller

# Spring MVC



# Spring MVC

- ▶ Controller
  - Responsável por tratar a requisição
- ▶ Model
  - Responsável pelo acesso a banco de dados, validações e regras de negócio
- ▶ View
  - Responsável por desenhar/renderizar/transformar em uma saída (HTML/JSON/XML) os dados que serão enviados para o usuário

# Gradle

- ▶ Ferramenta para auxiliar no gerenciamento das dependências de um projeto.
- ▶ Responsável por buscar dependências, realizar o build do projeto, definir configurações de deploy, entre outras tarefas
- ▶ <https://gradle.org/>

# Spring Initializr

- ▶ <http://start.spring.io>
- ▶ Site mantido pelo Spring que possibilita criar um modelo básico de aplicação Spring
- ▶ Possibilita definir os detalhes da aplicação, escolher os frameworks que serão utilizados e gerar o modelo.

# Na prática

- ▶ Vamos criar um projeto de exemplo.
- ▶ Para isto, acesse o site do Spring Initializer e configure a sua aplicação com o grupo "br.edu.ulbra" e o artefato "sample"
- ▶ Selecione a opção Gradle
- ▶ Nas dependências coloque Web e Thymeleaf
- ▶ Clique no botão Generate Project
- ▶ Baixe o projeto, descompacte o arquivo ZIP e importe ele como um projeto Gradle no Eclipse/IntelliJ

# IntelliJ

- ▶ Import Project
- ▶ Selecionar a pasta raiz do projeto (sample)
- ▶ Selecionar Import Project from External Model e selecionar o Gradle
- ▶ Selecionar a opção "Use auto import"
- ▶ Escolher o SDK do projeto (Caso não esteja habilitado, criar um novo SDK utilizando o botão +/JDK e procurando a pasta de instalação do Java JDK 1.8 no computador)
- ▶ Finish
- ▶ Aguardar a importação das dependências e configuração do Projeto
- ▶ Após carregar todas as dependências, ele deve gerar uma opção para executar o projeto

# Eclipse

- ▶ File/Import
- ▶ Selecionar Gradle/Existing Gradle Project
- ▶ Selecionar a pasta raiz do projeto (sample)
- ▶ Next/Next/Finish
- ▶ Aguardar a importação das dependências e configuração do Projeto
- ▶ Para executar, selecionar a classe SampleApplication e executar como Java Application

# Na prática

- ▶ Espere importar todas as dependências necessárias
- ▶ Após, execute o projeto. Ele deve inicializar o servidor de aplicação, com o deploy realizado e disponibilizar a aplicação em:
  - <http://localhost:8080>
- ▶ Entretanto, como não colocamos página alguma, não teremos nada

# Criando um Controller

- ▶ Crie uma nova classe em Java com o nome IndexController
- ▶ Coloque no package  
br.edu.ulbra.sample.controller
- ▶ Adicione a notação @Controller antes do nome da classe

# Criando um método com mapeamento

- ▶ Crie um método nesta classe chamado index, retornando uma String:

```
public String index(){  
    return "index";  
}
```

- ▶ Coloque a notação de mapeamento abaixo sobre o método:  
`@RequestMapping("/")`

# Criando o template

- ▶ Crie um arquivo em src/resources/templates chamado index.html
- ▶ Coloque o conteúdo HTML que desejar, desde que seja um HTML válido

# Outras Notações para Mapeamentos

- ▶ @GetMapping
- ▶ @PostMapping
- ▶ @DeleteMapping
- ▶ @PutMapping

# Outras Notações para Mapeamentos

- ▶ RequestMapping também pode ser utilizado a nível de Controller

```
@Controller
```

```
@RequestMapping("/compras")
```

```
public class ComprasController {
```

```
    @GetMapping("/cadastrar") {
```

```
        public String cadastrarCompras(){...}
```