DESENVOLVIMENTO DE SOFTWARE PARA A WEB 1

Módulo 6
SpringMVC + Thymeleaf

Delano Medeiros Beder
delano@dc.ufscar.br

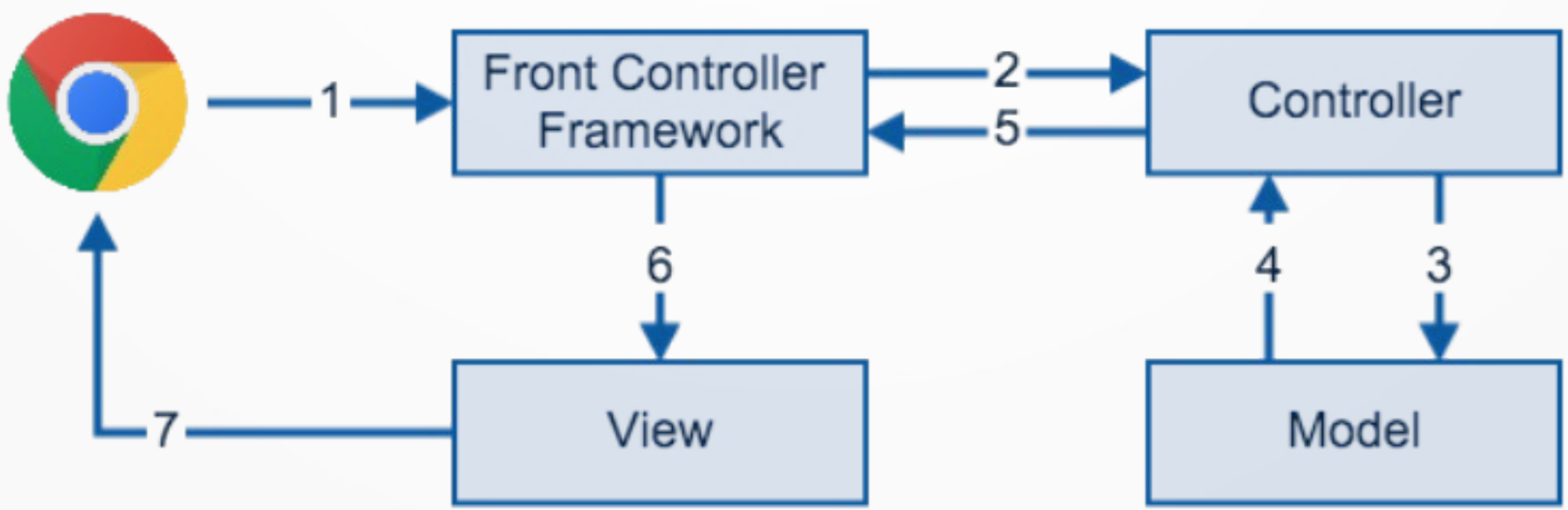
Spring

O Spring não é um framework apenas, mas um conjunto de projetos que resolvem várias situações do cotidiano de um programador, ajudando a criar aplicações Java com simplicidade e flexibilidade.

Existem muitas áreas cobertas pelo ecossistema Spring, como **Spring Data JPA** para acesso a banco de dados, **Spring Security** para prover segurança, e diversos outros projetos que vão de *cloud computing* até *big data*.

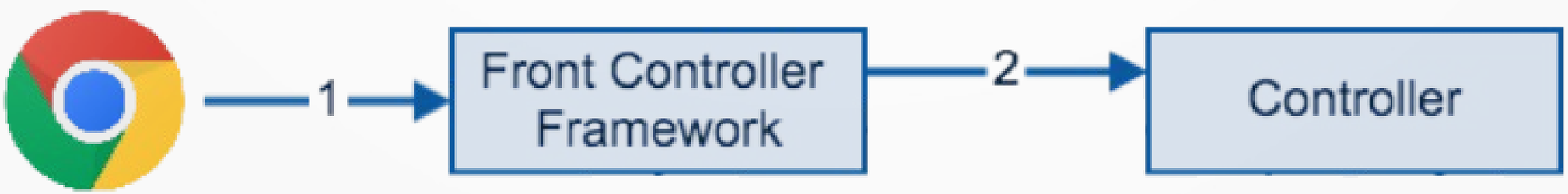
Spring MVC

- Dentre os projetos **Spring**, o **Spring MVC** é o framework que te ajuda no desenvolvimento de aplicações web robustas, flexíveis e com uma clara separação de responsabilidades nos papéis do tratamento da requisição.



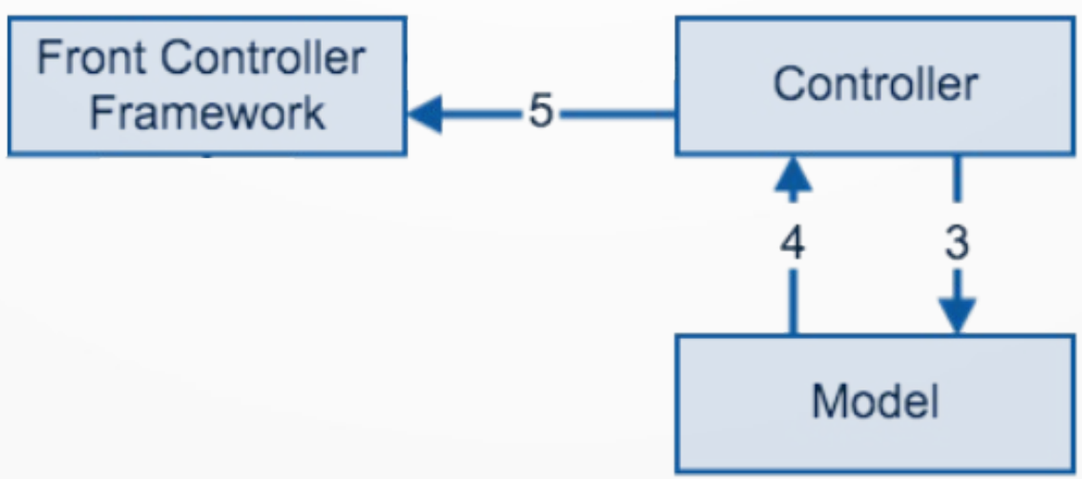
Spring MVC

1. Acessamos uma URL no navegador que envia a requisição HTTP para o servidor que roda a aplicação web desenvolvido utilizando o **Spring MVC**. Perceba que quem recebe a requisição é o controlador (**Front Controller**) do **Spring MVC**.
2. O controlador do **Spring MVC**, irá procurar qual classe é responsável por tratar essa requisição, entregando a ela os dados enviados pelo navegador. Essa classe faz o papel do ***Controlador***.



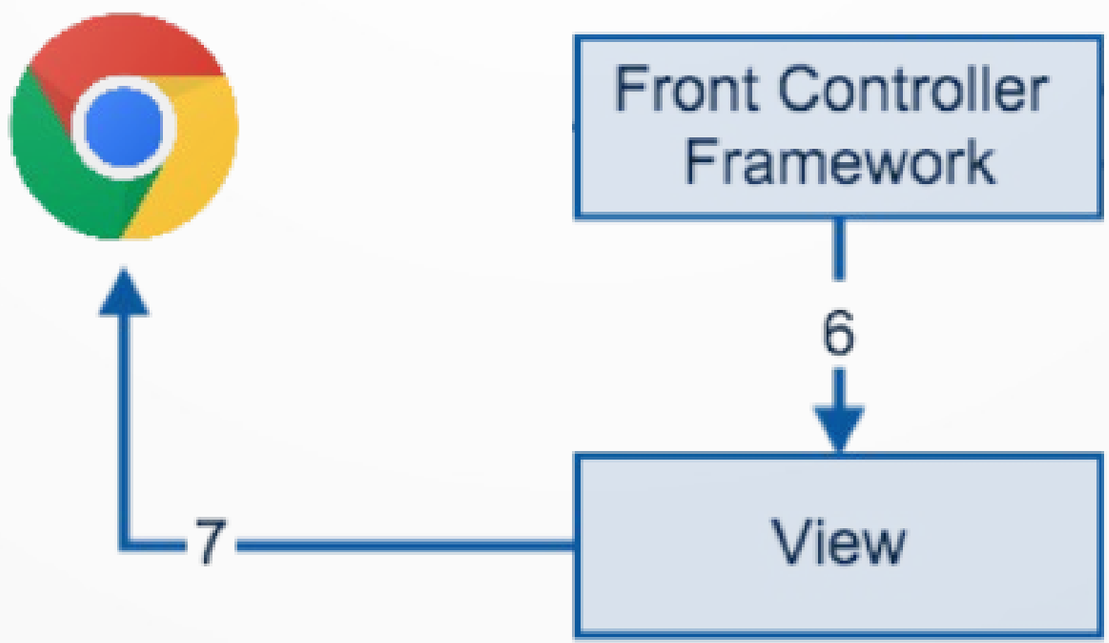
Spring MVC

3. O ***Controlador*** passa os dados para o ***Modelo***, que por sua vez executa todas as regras de negócio, como cálculos, validações e acesso ao banco de dados.
4. O resultado das operações realizadas pelo ***Modelo*** é retornado ao ***Controlador***.
5. O ***Controlador*** retorna o nome da ***Visão***, junto com os dados que a visão precisa para renderizar a página.

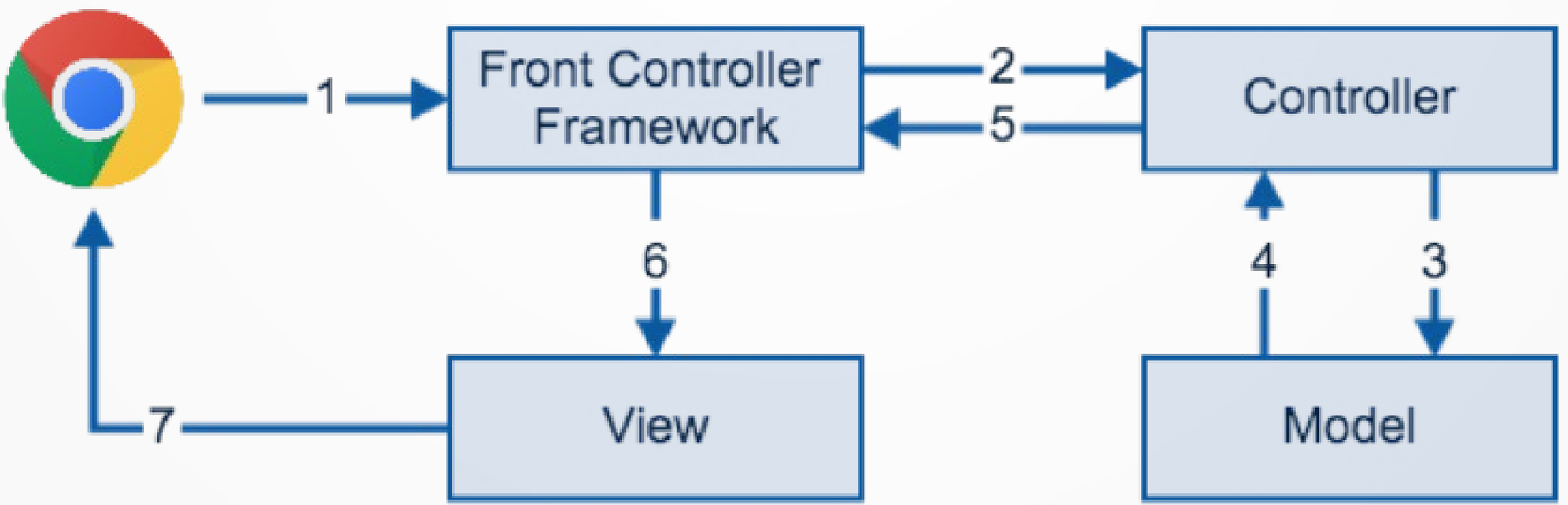


Spring MVC

6. O **Spring MVC** “encontra” a visão que processa os dados, transformando o resultado em um HTML.
7. Finalmente, o HTML é retornado ao navegador do usuário.



Spring MVC


1
Front Controller
Framework
2
5
Controller
6
4
3
7
View
Model

Controlador MVC

@Controller

Transforma a classe em um bean do tipo *controller do MVC*

```java
package br.ufscar.dc.dsw.controller;

import java.text.SimpleDateFormat;

@Controller
public class AloMundoController {

    @GetMapping("/")
    public String index(Model model) {

        SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss - dd MMMM yyyy");

        Calendar cal = Calendar.getInstance();
        model.addAttribute("date", dateFormat.format(cal.getTime()));

        return "index";
    }

}
```

@GetMapping

Requisições HTTP GET

Dados Modelo

Visão

@GetMapping shortcut para @RequestMapping(method = RequestMethod.GET)

Thymeleaf

- O Thymeleaf não é um projeto Spring, mas uma biblioteca que foi criada para facilitar a criação da camada de view com uma forte integração com o Spring, e uma boa alternativa ao JSP.

- O principal objetivo do Thymeleaf é prover uma forma elegante e bem formatada para criarmos nossas páginas. O dialeto do Thymeleaf é bem poderoso como você verá no desenvolvimento da aplicação.

Thymeleaf (Exemplo Básico)

HomeController.java

```
@RequestMapping(“/”)
public String home (Model model) {
    model.addAttribute(“firstName”, “Fulano”);
    model.addAttribute(“lastName”, “Silva”);
    return “home”;
}
```

templates/home.html

```
<span th:text=”${‘Olá’+ firstName + ‘ ‘ + lastName}”>Oi</span>
```

Com renderização

Olá Fulano Silva

Sem renderização

Oi

SpringMVC + Thymeleaf

Demonstração 1

Thymeleaf (Exemplo I18n)

templates/home.html

Oi

messages_pt.properties

home.hi = Olá

messages_en.properties

home.hi = Hi

Renderização (pt)

Olá Fulano Silva

Renderização (en)

Hi Fulano Silva

SpringMVC + Thymeleaf

Demonstração 2

SpringMVC + Thymeleaf

Demonstração 3

Thymeleaf – Scoping Variables

```
<span>
    [[${user.firstName}]] [[${user.lastName}]] <br/>
    [[${user.address.line1}]] <br/>
    [[${user.address.line2}]] <br/>
    [[${user.address.city}]], [[${user.address.state}]] [[${user.address.zipcode}]]
</span>

<span th:object=”${user.address}”>
    [[${user.firstName}]] [[${user.lastName}]] <br/>
    [[*{line1}]] <br/>
    [[*{line2}]] <br/>
    [[*{city}]], [[*{state}]] [[*{zipcode}]]
</span>
```

Thymeleaf – Scoping Variables

```
<span th:object=”${user.address}>
    [[${user.firstName}]] [[${user.lastName}]] <br/>
    [[*{line1}]] <br/>
    [[*{line2}]] <br/>
    [[*{city}]], [[*{state}]] [[*{zipcode}]]
</span>

<span th:object=”${user.address}
        th:with=”fullName=${user.firstName + ‘ ‘ + user.lastName}”>
            [[${fullName}]] <br/>
            [[*{line1}]]<br/>
            [[*{line2}]]<br/>
            [[*{city}]], [[*{state}]] [[*{zipcode}]]
</span>
```

Conditionals and Loops

```
<th:block th:each="product : ${products}">
    <div>
        <span th:if="${product.isOnSale()}" th:text="${product.salePrice}"></span>
        <span th:unless="${product.isOnSale()}" th:text="${product.retailPrice}"></span>
    </div>
</th:block>
```

```
<th:block th:each="product : ${products}">
    <div>
        <span th:text="${product.isOnSale() ? product.salePrice :
        product.retailPrice}"></span>
    </div>
</th:block>
```

FIM