

Disciplina Regular 3

Desenvolvimento Web com Java EE

Graduação em Engenharia de Software - 2020

Etapa 3 Aula 1

Arquitetura MVC

Para acompanhar pelo Moodle você
deve estudar a etapa 4

Etapa 3 Aula 1 - Competências

- **Competências** Trabalhadas Nesta Etapa
 - **Construir aplicações Java Web usando Arquitetura MVC.**
 - Compreender a importância dos Frameworks na construção de aplicações.
 - Conhecer os conceitos fundamentais do Spring Framework.

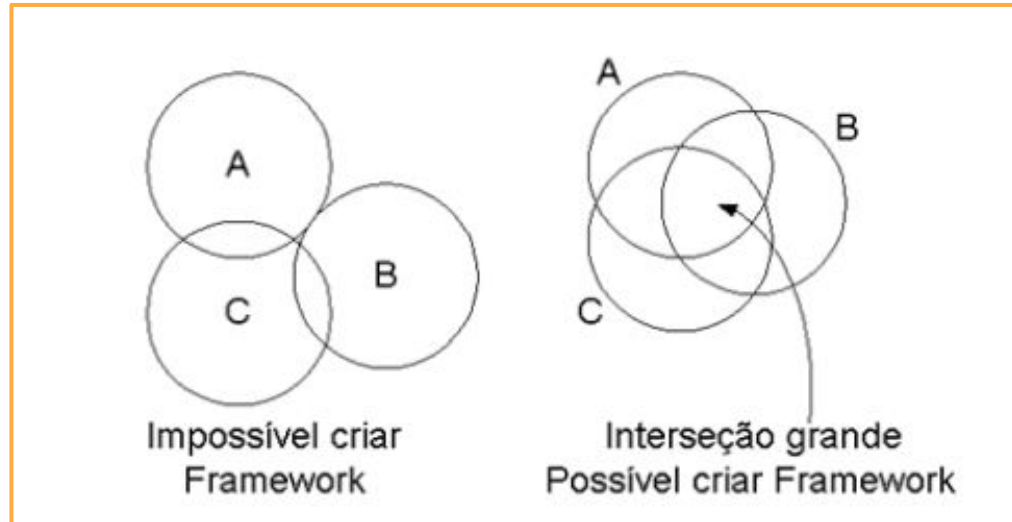
Etapa 3 Aula 1 - Checklist

- **Checklist DR3:**
 - Ter revisado os slides e **feito os exercícios** das aulas anteriores.
 - Ter lido o enunciado e tirado as dúvidas da **TP1**. Se possível ter iniciado a tarefa para 04/05/2020.
- **Checklist PB:**
 - Ter agendado a primeira reunião com o grupo para tratar do plano de trabalho do ASSESSMENT e da especificação do módulo.
 - Ter criado a conta pessoal no GitHub.
 - **TP7** será adiado para o dia 11/05/2020.

Framework

Conceitos

- Um **framework** captura a funcionalidade comum a várias aplicações.
- As aplicações devem ter algo razoavelmente importante em comum; algo que pertença a um mesmo domínio de problema.

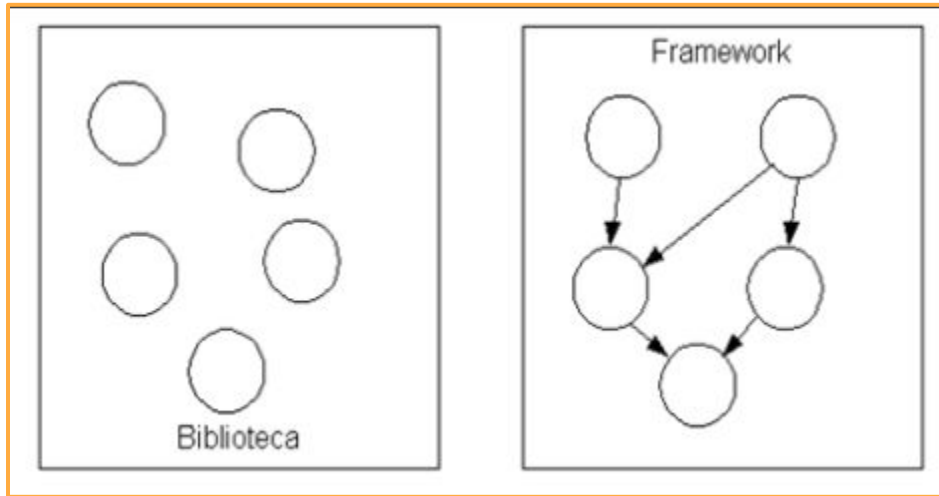


Conceitos

- Características principais de um Framework:
 - Provê uma solução para uma família de problemas semelhantes.
 - Usa um conjunto de classes e interfaces que mostra como decompor a família de problemas.
 - O conjunto de classes deve ser flexível e **extensível** para permitir a construção de várias aplicações com **pouco esforço**, especificando apenas as particularidades de cada aplicação.
- Ao receber um framework, seu trabalho consiste em prover os pedaços que são específicos para a sua aplicação.

Conceitos

- Framework X Biblioteca de Classes:
 - Em uma biblioteca de classes, cada classe é única e **independente** das outras.
 - Em um framework, as dependências / colaborações estão embutidas.



Conceitos

- Um Framework deve ser reusável.
- Deve ser **extensível** (herança → classe ou interface):
 - O Framework contém funcionalidade abstrata (sem implementação) que deve ser completada.
- Deve ser de uso seguro.
- Deve ser eficiente.
- Deve ser completo para endereçar soluções para o domínio do problema pretendido.

Spring Framework

- O Spring Framework foi construído com base no princípio de que Java EE deveria ser mais fácil de implementar.
- O que é o Spring?
 - **Integrador de Frameworks.**
 - Faz a ligação entre as diversas “camadas” (**tier**) de uma aplicação: Web, Persistência, Transações, Segurança etc.
 - Gerencia as classes de negócio.
 - Facilita a extensibilidade.

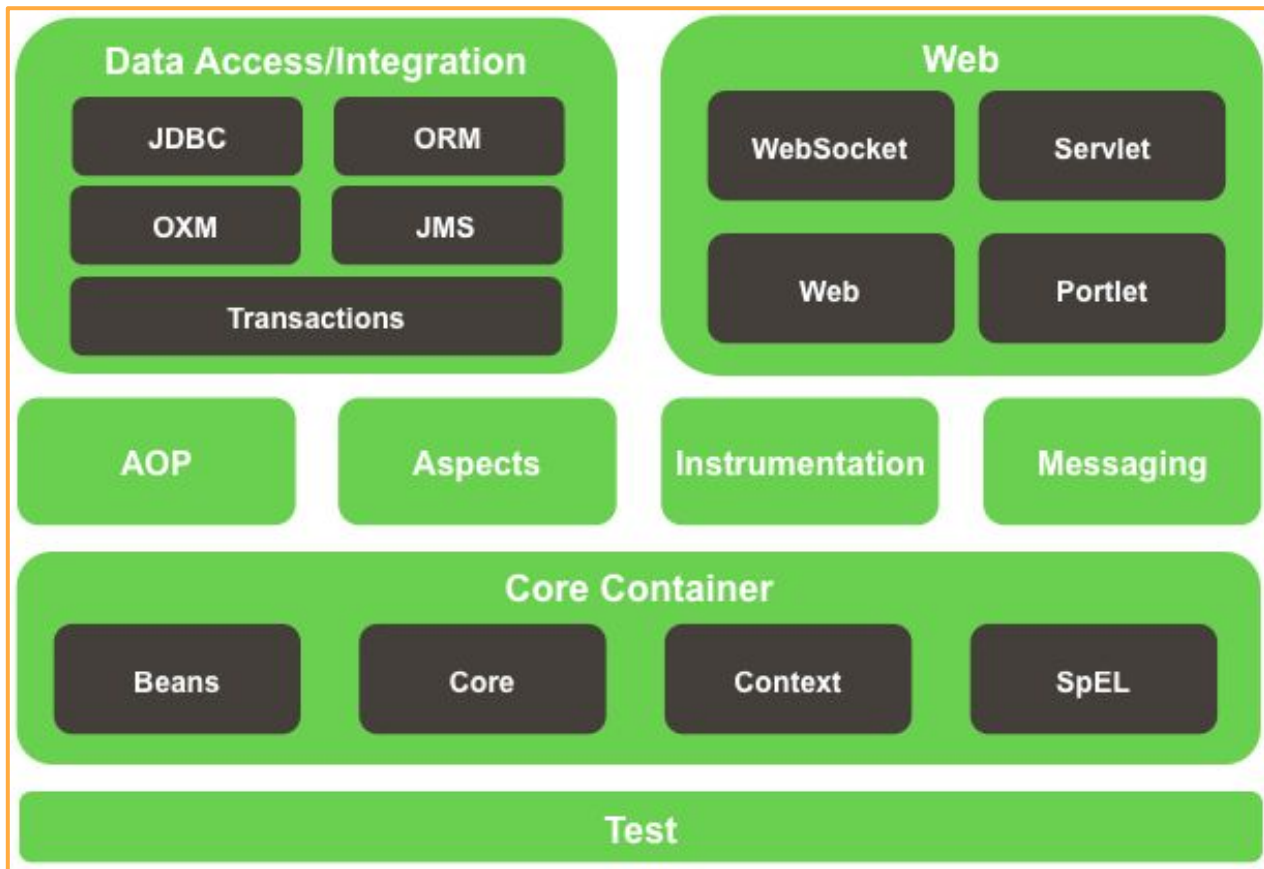
Spring Framework

- Spring é uma plataforma de desenvolvimento **voltada para o desenvolvimento de aplicações com classes Java “puras”** – POJO (Plain Old Java Object).
- Foi introduzido pela primeira vez no livro Expert One-on-One J2EE Design and Development de **Rod Johnson**.
- Um dos problemas mais comuns com o qual os arquitetos e desenvolvedores de aplicação precisam lidar é obter **desacoplamento** dos componentes – este é o foco principal do Spring.

Spring Framework

- O Spring cuida todas as camadas de uma aplicação – da Web até a camada de negócios.
- A arquitetura em camadas do Spring Framework permite decidir qual de seus componentes você deseja implementar.
 - Fornece a flexibilidade de usar o **Spring em fases**, onde é possível usar um componente do Spring, colocá-lo em funcionamento e então selecionar outro.
- Os módulos do Spring são projetados sobre o módulo principal “**Core**”, que funciona como container para criar, gerenciar e configurar beans no tempo de execução.

Spring Framework



IoC Design Pattern

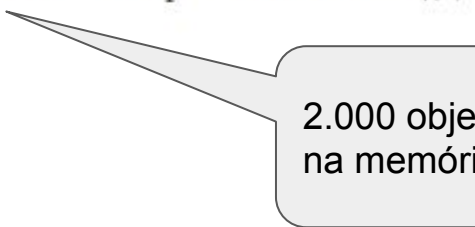
- Inversion of Control: frameworks executam diversas tarefas independentes da aplicação, ou seja, quem controla é o framework.
- IoC é um padrão que ajuda a eliminar o acoplamento entre componentes OOP.
- Há diferentes maneiras de implementar IoC, mas, em termos básicos, é obtido introduzindo-se uma interface entre X e Y para que interajam um com o outro.
- O container do Spring Framework cuida da resolução de dependência e do ciclo de vida dos objetos em runtime.

Dependency Injection

- Dependency Injection é um padrão de desenvolvimento utilizado quando é necessário manter baixo o nível de acoplamento entre diferentes módulos de um sistema.
- Nesta solução as dependências entre os módulos não são definidas programaticamente, mas sim pela configuração de uma infraestrutura de software responsável por "injetar" em cada componente suas dependências declaradas.
- Dependency Injection se relaciona com o padrão IoC mas não pode ser considerada um sinônimo deste – são conceitos complementares.

Dependency Injection


```
1 public class ClienteServico {  
2  
3     private ClienteRepositorio repositorio = new ClienteRepositorioJPA();  
4  
5     public void salvar(Cliente cliente) {  
6         this.repositorio.salvar(cliente);  
7     }  
8  
9     ...  
10 }
```



2.000 objetos na memória



```
1 public class ClienteServico {  
2  
3     @Autowired  
4     private ClienteRepositorio repositorio;  
5  
6     ...  
7 }
```



2 objetos na memória

Dependency Injection

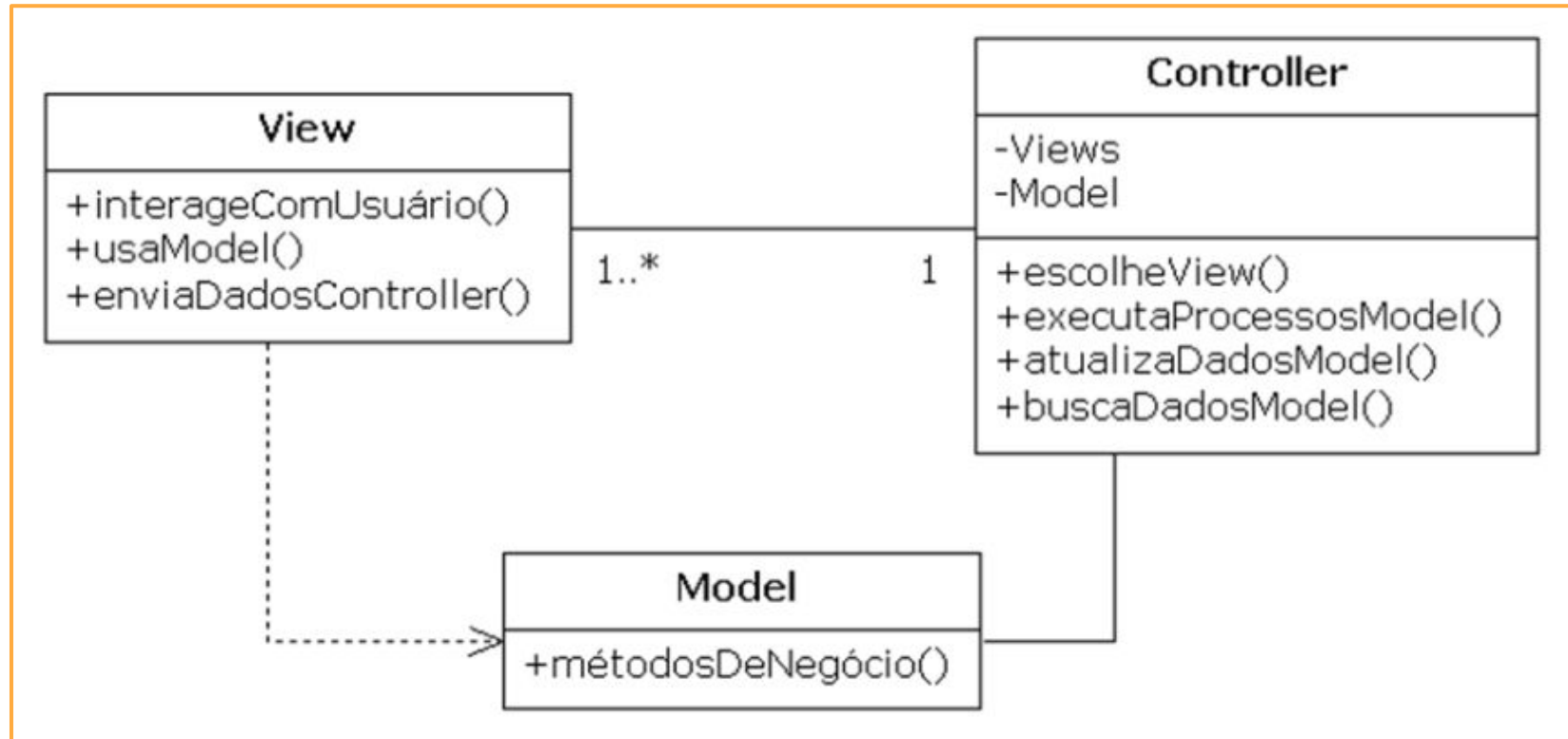
- Para que uma instância do tipo `ClienteRepository` possa ser injetada em algum dos pontos de injeção é preciso que ela se torne um bean Spring.
- Fazemos isso anotando a classe com **@Component** ou com qualquer uma de suas especializações: **@Repository**, **@Service**, **@Controller**.
- O Spring chama essas quatro anotações de estereótipos, sendo que as três últimas são como anotações “filhas” da anotação **@Component**.

Dependency Injection

- A anotação **@Component** é a mais geral e as outras são para usos mais específicos em componentes de persistência, serviço e controlador, respectivamente.
- Apesar de termos esses quatro estereótipos, eles não tem especificidades, com exceção da anotação **@Repository**, que pode adicionar um pequeno comportamento relacionado a tradução das exceções de persistência.

Spring MVC

Model-View-Controller



Model-View-Controller

- **View** → Criar as telas com os formulários a serem preenchidos pelo ator do caso de uso → JavaServer Pages.
- **Model** → Criar um Javabean com propriedades correspondentes aos campos do formulário de forma a transportá-los em um único objeto. Se necessário, criar as classes de serviços e repositórios.
- **Controller** → Criar um servlet que represente o processamento do caso de uso:
 1. Receber os dados do formulário.
 2. Validar os dados do formulário.
 3. **Executar o fluxo do caso de uso e seus desvios.**
 4. Inserir objetos na requisição ou na sessão.
 5. Redirecionar para a view, conforme o resultado do processamento.

Aqui entra
o Spring
MVC

Características

- Spring MVC é um módulo do Spring Framework.
- Spring MVC é um framework extensível para criação de aplicações web, direcionando o desenvolvimento naturalmente para a criação de componentes Model, View e Controller.
- Seu criador, Rod Johnson, dedicou vários anos de trabalho para desenvolver um framework modular, versátil e completo.

Características

- No Spring MVC, uma requisição feita pelo browser tem um ciclo de vida que passa por vários componentes do framework.
- O componente responsável por orquestrar o funcionamento do Spring MVC é o **Dispatcher Servlet**.
- Trata-se da implementação do padrão **Front Controller**, muito usado na escrita de frameworks voltados à criação de aplicações web.



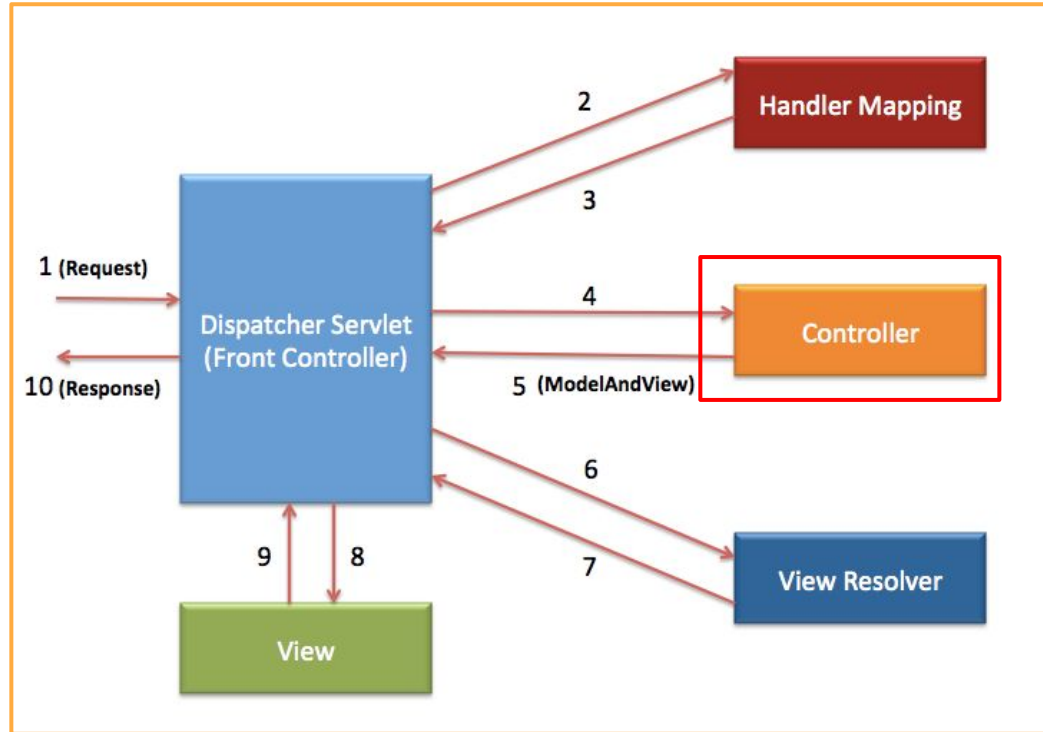
Conceito → **Front Controller**

O objetivo desse design pattern é fornecer um ponto de entrada central para todas as requisições direcionadas à nossa aplicação.

O trabalho desse padrão consiste em interpretar as requisições e decidir qual o componente responsável por seu processamento e eventual retorno para o usuário.

Características

- Toda chamada ao servidor inicia-se sob a forma de uma requisição e termina como uma resposta enviada ao cliente.



Receita para o Spring MVC - 1 de 2

- Criar o Projeto:
 - Criar o projeto web+maven normalmente: JEE 6 + JSE 8.
 - Lembrar de colocar os projetos em um diretório "raso" (com nome curto) para facilitar os backups.
- Configurar a Aplicação Web:
 - Criar a pasta WEB-INF e o deployment descriptor (web.xml).
 - Copiar e colar os arquivos de configuração do Spring.
 - applicationContext.xml
 - dispatcher-servlet.xml
 - Configurar o Dispatcher Servlet no web.xml.

Receita para o Spring MVC - 2 de 2

- Dependências de Bibliotecas do Maven:
 - "org.springframework:**spring-webmvc**"
 - "org.hibernate:**hibernate-validator**"
 - "javax.servlet:**jstl**"
 - "commons.lang"
 - etc.

Exercício

- Construir uma aplicação simples implementando o Primeiro Controller.



```
1  package br.com.infnet.web;
2
3  import org.springframework.stereotype.Controller;
4  import org.springframework.web.bind.annotation.RequestMapping;
5
6  @Controller
7  public class HelloController {
8
9      @RequestMapping("/helloSimples")
10     public String helloSimples() {
11
12         System.out.println("Passei pelo Controller do Spring MVC!");
13         return "index";
14     }
15 }
```

Exercício

- Incluir no Controller anterior um novo método que insira uma mensagem na requisição antes do retorno. Use EL para recuperar a mensagem no `index.jsp`.

```
17      @RequestMapping("/helloMensagem")
18      public ModelAndView helloMensagem() {
19
20          ModelAndView mav = new ModelAndView();
21          mav.setViewName("index");
22          mav.addObject("mensagem", "Hello Spring MVC World!");
23          return mav;
24      }
25 }
```