Spring-MVC

A Inversão de Controle

O IoC é um princípio de design que separa objetos de um programa orientado a objetos de suas dependências.

O desacoplamento é alcançado externalizando a responsabilidade da criação de objetos e da Injeção de Dependência em um componente externo, como um contêiner IoC.

No tempo de execução, uma entidade externa, como um contêiner IoC, resolve suas implementações concentradas especificadas em algum lugar e as injeta.

A Injeção de Dependência

Injeção de Dependência é uma forma específica de Inversão de Controle.

É um padrão de design mais formalizado, em que as dependências de um objeto são injetadas por um montador.

A DI é geralmente realizada em três estilos principais: injeção de construtor, injeção de propriedade (setter) ou, às vezes, injeção de interface. IoC e DI são freqüentemente usados de forma intercambiável.

O DI oferece vários benefícios, incluindo o desacoplamento efetivo de dependências, código mais limpo.

As Anotações de Componentes

Estereótipo	Descrição
@Component	Um tipo genérico para todos os componentes gerenciados pelo Spring (beans).
@Service	Meta-anotação de marcador para componentes da camada de serviço. Atualmente, Spring trata isso da mesma forma que **@Component**, sem função especial.
@Repository	Usado como DAOs na sua camada de persistência. As bibliotecas Spring Data fornecem funcionalidade adicional.
@Controller	Lida com pontos de extremidade do Web MVC para processar solicitações HTTP mapeadas para URLs específicos.
@RestController	Um controlador especializado para serviços da Web RESTful, parte do Web MVC. É uma meta-anotação que combina **@Controller** e **@ResponseBody**.

O Spring MVC

Construído sobre tecnologias da Web Java, como Servlets, JSP e JSTL, e pode ser implantado em qualquer contêiner padrão do Servlet, como o Tomcat.

Implementado com base no padrão de arquitetura Model-View-Controller (MVC), com separação clara de interesses usando anotações simples e tags XML de namespace.

Validação de entrada declarativa, ligação de dados e manipulação de exceção.

Mapeamento de URL flexível com transformação automática de solicitação e resposta em vários formatos, como JSON, XML e HTML.

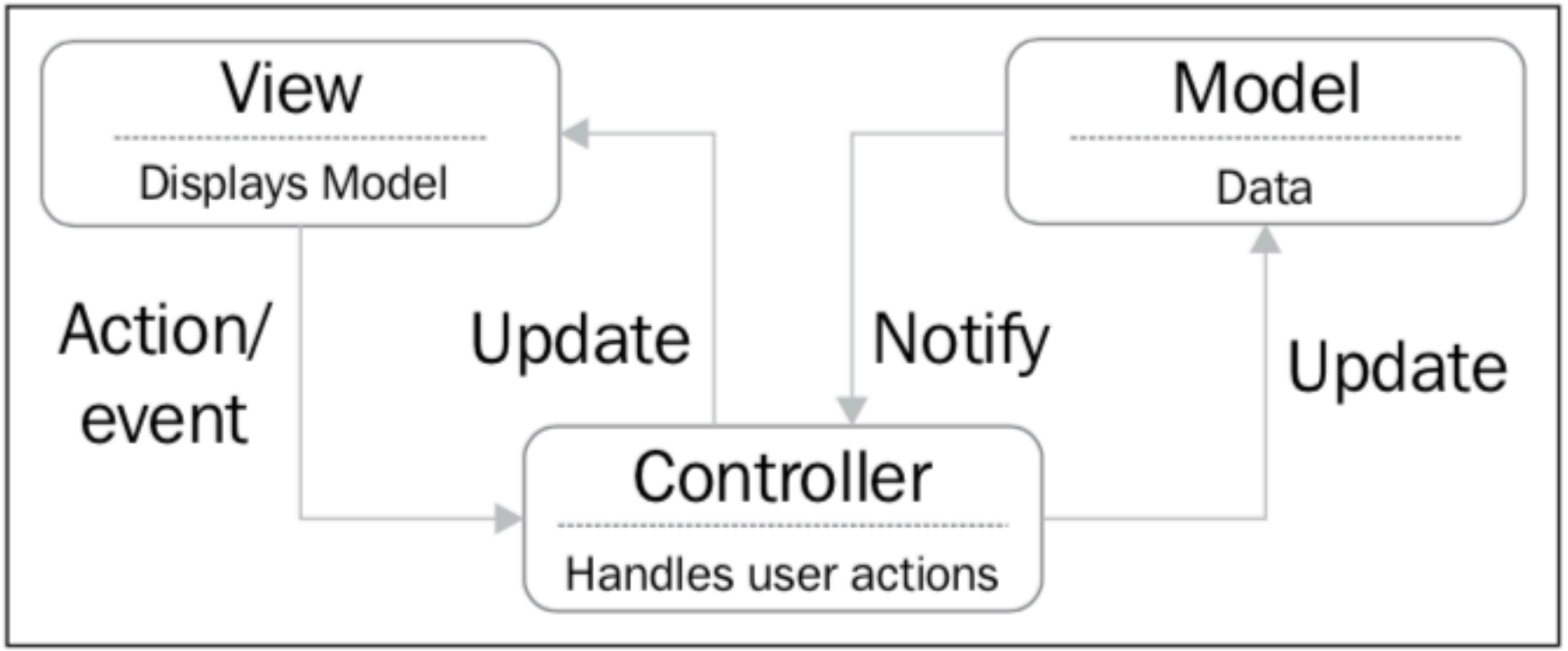
Model-View-Controller

O **Model** representa dados, o **View** exibe o modelo e o **Controller** manipula as ações do usuário.

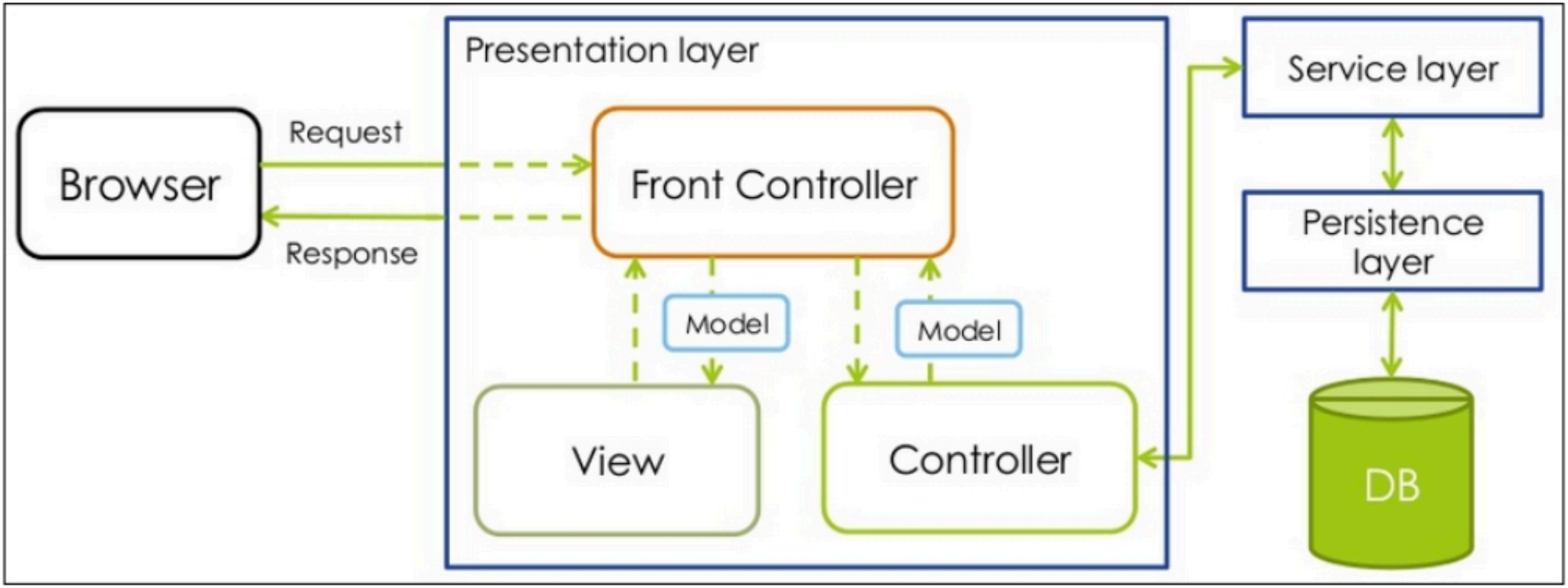
O **Model** pode ser qualquer dado, incluindo aquele armazenado em um banco de dados, podendo ser exibido em várias visualizações, dependendo de como o aplicativo é projetado.

O **Controller** atua como um intermediário entre **View** e **Model**. Ele delega as ações do usuário para manipuladores apropriados e, finalmente, redireciona para outra exibição para exibir o resultado dessa ação.

Model-View-Controller



Application's Layer



The layers of a Spring MVC application

O Papel do Controller

Os **Controllers**, com seus métodos anotados com @RequestMapping, manipulam solicitações da web.

Eles aceitam dados de entrada em vários formulários e os transformam em atributos do **Model** para serem consumidos por visualizações que são exibidas de volta ao cliente.

Eles conectam o usuário aos **beans** da camada de serviço, onde o comportamento do aplicativo é definido.

```
@Controller
public class CadastroController {
    @Autowired
    private CadastroService service;

    @RequestMapping({"/", "index.html"})
    public String inicia(Model model) {
        model.addAttribute("cliente", new Cliente());
        return "index";
    }

    @RequestMapping("/cadastra")
    public ModelAndView cadastra(@ModelAttribute("cliente") Cliente cliente) {
        service.salvar(cliente);
        return new ModelAndView("resultado", "cliente", cliente);
    }
}
```

Tratando requisições de URLs

Os mapas de anotação **@RequestMapping** solicitam URLs para uma classe **@Controller** inteira ou seus métodos manipuladores.

É anotado **@RequestMapping** no nível da classe para mapear um grupo de URLs relacionadas.

Anotamos **@RequestMapping** no nível do método para tratar ações específicas, como criar, ler, atualizar, excluir, fazer upload e download.

Anotações que tratam Requisições

Anotações	Descrição

@PathVariable	A URL pode ter qualquer número de variáveis de caminho esta é mapeada por uma anotação **@PathVariable**. Ele é colocado dentro de chaves e anotado como um argumento de método para mapeamento. Ex.:
	`@RequestMapping("/editaCliente/{clienteId}")`
	`public ModelAndView editaCliente(@PathVariable(value="clienteId") int id) {…}`

@RequestParam	Parâmetros de solicitação que são sequenciais com strings de URI podem ser mapeados com argumentos de método usando a anotação **@RequestParam**. Ex.:
	`@RequestMapping(path = "/tasks", method = RequestMethod.GET)`
	`public String list(@RequestParam(name = "status",required = false) String status, Model model) {…}`

@ModelAttribute	Um método com argumentos mapeados com **@ModelAttribute** permite associar os dados transmitidos por uma requisição em um **Model**. Ex.:
	`@RequestMapping("/removeCliente")`
	`public ModelAndView removeCliente(@ModelAttribute("lista") Cadastro cadastro, RedirectAttributes redirectAttributes) {…}`

Referência

https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html#spring-web

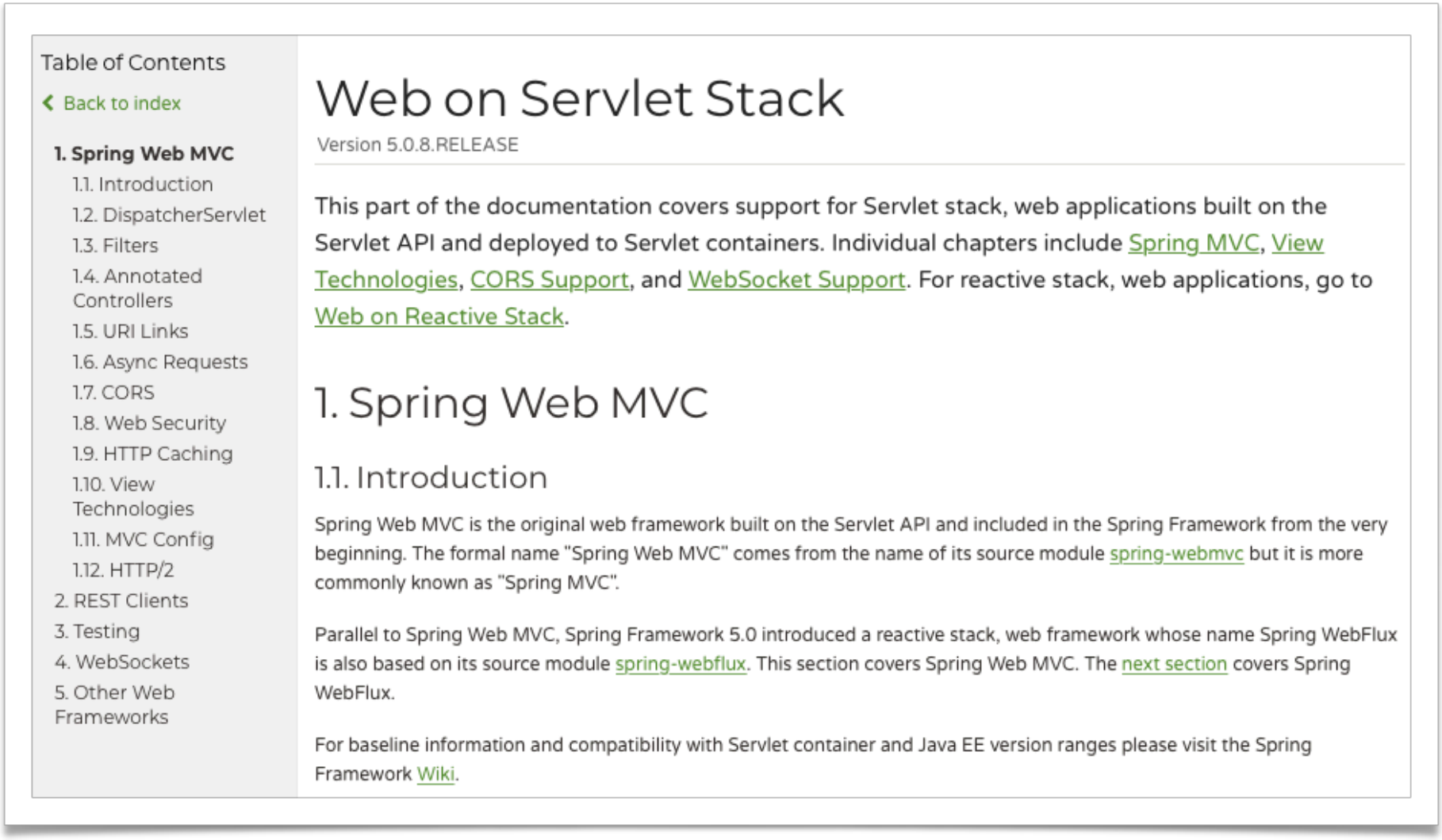


Table of Contents

Back to index

Web on Servlet Stack

Version 5.0.8.RELEASE

This part of the documentation covers support for Servlet stack, web applications built on the Servlet API and deployed to Servlet containers. Individual chapters include Spring MVC, View Technologies, CORS Support, and WebSocket Support. For reactive stack, web applications, go to Web on Reactive Stack.

1. Spring Web MVC

1.1. Introduction

Spring Web MVC is the original web framework built on the Servlet API and included in the Spring Framework from the very beginning. The formal name "Spring Web MVC" comes from the name of its source module spring-webmvc but it is more commonly known as "Spring MVC".

Parallel to Spring Web MVC, Spring Framework 5.0 introduced a reactive stack, web framework whose name Spring WebFlux is also based on its source module spring-webflux. This section covers Spring Web MVC. The next section covers Spring WebFlux.

For baseline information and compatibility with Servlet container and Java EE version ranges please visit the Spring Framework Wiki.