Spring web mvc

Trương Phước Lộc

Nội dung

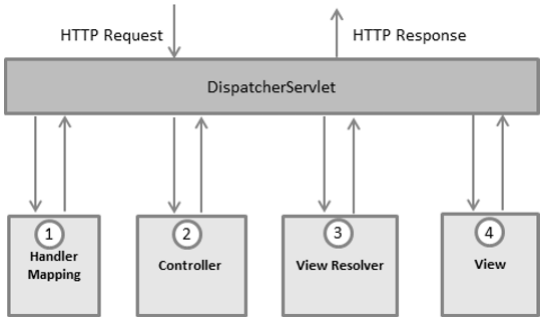
1. Spring MVC
2. DispatcherServlet
3. Controller
4. View

1. Spring Web MVC

- Là web framework được xây dựng trên API của Servlet và được tích hợp trong Spring Framework.
- Thường được gọi là Spring MVC.
- Cung cấp kiến trúc MVC (Model-View-Controller).

2. DispatcherServlet

- Spring MVC được thiết kế xung quanh DispatcherServlet xử lý tất cả các Requests và Responses



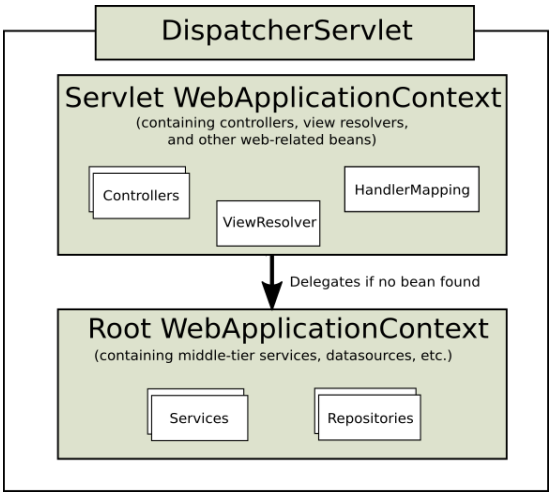
2. DispatcherServlet

- Khi nhận HTTP Request, DispatcherServlet gọi HandlerMapping để gọi Controller phù hợp.
- Controller nhận Request và gọi phương thức phù hợp (GET, POST, …). Phương thức này sẽ dùng thông tin (model), xử lý nghiệp vụ và trả kết quả (View, model) cho DispatcherServlet.
- DispatcherServlet dùng ViewResolver để hiển thị View cho Request.
- DispatcherServlet truyền thông tin (model) tới View để hiển thị.

2. DispatcherServlet

```
<web-app>

  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>

  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/app-context.xml</param-value>
  </context-param>

  <servlet>
    <servlet-name>app</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value></param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>app</servlet-name>
    <url-pattern>/app/*</url-pattern>
  </servlet-mapping>

</web-app>
```

2. WebApplicationContext



WebApplicationContext

- HandlerMapping
- HandlerAdapter
- HandlerExceptionResolver
- ViewResolver
- LocaleResolver & LocaleContextResolver
- ThemeResolver
- MultipartResolver
- FlashMapManager

WebApplicationContext => Locale => Theme => Multipart => Controller => View

3. Controller

- Controller/RestController: **/hello** ⇔ **HelloController**
- Model: **Model**
- View: **"index"**

```
@Controller
public class HelloController {
  @GetMapping("/hello")
  public String handle(Model model) {
    model.addAttribute("message", "Hello World!");
    return "index";
  }
}
```


3. Controller

- Sử dụng component scanning trong tập tin cấu. hình

```
@Configuration @ComponentScan("org.example.web")
public class WebConfig {
  // ...
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="…">
  <context:component-scan base-package="org.example.web"/>
  <!-- ... -->
</beans>
```

3.1 Mapping Requests With @RequestMapping

- **@RequestMapping** sử dụng để ánh xạ các Requests tới phương thức của các Controllers: **URL**, **HTTP method**, **request parameters**, **headers**, **media types**.
- @GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping

3.1 Mapping Requests With @RequestMapping

```
@RestController
@RequestMapping("/persons")
class PersonController {
        @GetMapping("/{id}")
        public Person getPerson(@PathVariable Long id) {
                // ...
        }
        @PostMapping
        @ResponseStatus(HttpStatus.CREATED)
        public void add(@RequestBody Person person) {
                // ...
        }
}
```

3.2.1 HTTP HEAD and OPTIONS

- @GetMapping
- @RequestMapping(method=HttpMethod.GET)

- GET,HEAD,POST,PUT,PATCH,DELETE,OPTIONS: …

3.2.2 URI Path Patterns

- Sử dụng ký tự đại diện
  - ?: 1 ký tự
  - *: 0 hoặc n ký tự
  - **: 0 hoặc n đoạn

3.2.3 URI Path Patterns

- Sử dụng **@PathVariable**
  - Lớp hoặc phương thức

```
@Controller
@RequestMapping("/owners/{ownerId}")
public class OwnerController {
    @GetMapping("/pets/{petId}")
    public Pet findPet(@PathVariable Long ownerId,
            @PathVariable Long petId){
        // ...
    }
}
```

  - Cú pháp `{varName:regex}` tham khảo Path matching

```
@GetMapping("/{name:[a-z-]+}-{version:\\d\\.\\d\\.\\d}{ext:\\.[
a-z]+}")
```


3.2.4 Matrix Variables

- Matrix Variables

```
// GET /pets/42;q=11;r=22
@GetMapping("/pets/{petId}")
public void findPet(@PathVariable String petId,
                    @MatrixVariable int q) {
    // petId == 42 // q == 11
}

// GET /owners/42;q=11/pets/21;q=22
@GetMapping("/owners/{ownerId}/pets/{petId}") public void
findPet(
 @MatrixVariable(name="q", pathVar="ownerId") int q1,
 @MatrixVariable(name="q", pathVar="petId") int q2) {
    // q1 == 11 // q2 == 22
}
```

3.2.4 Matrix Variables

```
// GET /pets/42
@GetMapping("/pets/{petId}")
public void findPet(
  @MatrixVariable(required=false, defaultValue="1") int q) {
        // q == 1
}

// GET /owners/42;q=11;r=12/pets/21;q=22;s=23
@GetMapping("/owners/{ownerId}/pets/{petId}")
public void findPet(
@MatrixVariable MultiValueMap<String, String> matrixVars,
@MatrixVariable(pathVar="petId"") MultiValueMap<String,
String> petMatrixVars) {
        // matrixVars: ["q" : [11,22], "r" : 12, "s" : 23]
        // petMatrixVars: ["q" : 22, "s" : 23]
}
```


3.2.4 Matrix Variables

- Cấu hình (mặc định là false)

```
<?xml version="1.0" encoding="UTF-8"?> <beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="…">
  <mvc:annotation-driven enable-matrix-variables="true"/>
</beans>
```

3.2.5 Media Types

- **Consumes** dựa trên Content-Type của Request
- **Produces** dựa trên Accept của Request

```
@PostMapping(path = "/pets", consumes =
                                    "application/json")
public void addPet(@RequestBody Pet pet) {
        // ...
}

@GetMapping(path = "/pets/{petId}", produces =
                    "application/json;charset=UTF-8")
@ResponseBody
public Pet getPet(@PathVariable String petId) {
        // ...
}
```


3.2.6 Request Parameters and Header Values

```
@GetMapping(path = "/pets/{petId}", params =
                    "myParam=myValue")
public void findPet(@PathVariable String petId) {
        // ...
}

@GetMapping(path = "/pets", headers =
                    "myHeader=myValue")
public void findPet(@PathVariable String petId) {
        // ...
}
```

3.2.7 Phương thức @RequestMapping

- Arguments
  - WebRequest, NativeWebRequest
  - javax.servlet.ServletRequest, javax.servlet.ServletResponse
  - javax.servlet.http.HttpSession
  - java.util.Locale
  - …
- Return values
  - @ResponseBody
  - View
  - ModelAndView
  - …

3.2.7 Phương thức @RequestMapping

- @RequestParam

```
@GetMapping
public String setupForm(@RequestParam("petId") int petId, ModelMap model) {
    Pet pet = this.clinic.loadPet(petId);
    model.addAttribute("pet", pet);
    return "petForm";
}
```

- @RequestBody

```
@PutMapping("/something")
public void handle(@RequestBody String body, Writer writer) throws IOException {
    writer.write(body);
}
```

3.2.7 Phương thức @RequestMapping

- @ResponseBody

```
@GetMapping("/something")
@ResponseBody
public String helloWorld() {
    return "Hello World";
}
```


3.3 @RestController

- Sử dụng để tạo REST api: JSON, Xml, MediaType
- Là sự kết hợp của @ResponseBody và @Controller
- …

3.4 Testing Controllers

- Sử dụng module spring-test

4. View

- Spring xử lý View thông qua 2 interface: **ViewResolver** và **View**
- Tất cả Controller => View thông qua tên (String, View, ModelAndView)

4. View

```
<bean id="viewResolver"
class="org.springframework.web.servlet.view.UrlBasedViewResolver"
>
  <property name="viewClass"
        value="org.springframework.web.servlet.view.JstlView"/>
  <property name="prefix" value="/WEB-INF/jsp/"/>
  <property name="suffix" value=".jsp"/>
</bean>

<bean id="viewResolver"
class="org.springframework.web.servlet.view.ResourceBundleViewRe
solver">
  <property name="basename" value="views"/>
  <property name="defaultParentView" value="parentView"/>
</bean>
```


4. View

- Chaining ViewResolvers
- RedirectView
- ContentNegotiatingViewResolver

Tham khảo thêm:
https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html#mvc-viewresolver

Chaining ViewResolvers

- Spring hỗ trợ nhiều ViewResolver
- View sẽ được xử lý tuần tự
- Thứ tự???
- Không liên kết được???

Chaining ViewResolvers

```
<bean id="jspViewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewRe
solver">
  <property name="viewClass"
value="org.springframework.web.servlet.view.JstlView"/>
  <property name="prefix" value="/WEB-INF/jsp/"/>
  <property name="suffix" value=".jsp"/>
</bean>
<bean id="excelViewResolver".
class="org.springframework.web.servlet.view.XmlViewResolver">
  <property name="order" value="1"/>
  <property name="location" value="/WEB-INF/views.xml"/>
</bean>

<!-- in views.xml -->
<beans>
<bean name="report"
        class="org.springframework.example.ReportExcelView"/>
</beans>
```

Hỏi đáp

Tài liệu tham khảo

- Tài liệu giảng dạy – ThS. Nguyễn Hoàng Anh, ĐH. Khoa học tự nhiên
- https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html
- https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm