互联网应用开发技术

Web Application Development

第11课
WEB后端框架-SPRING MVC

Episode Eleven

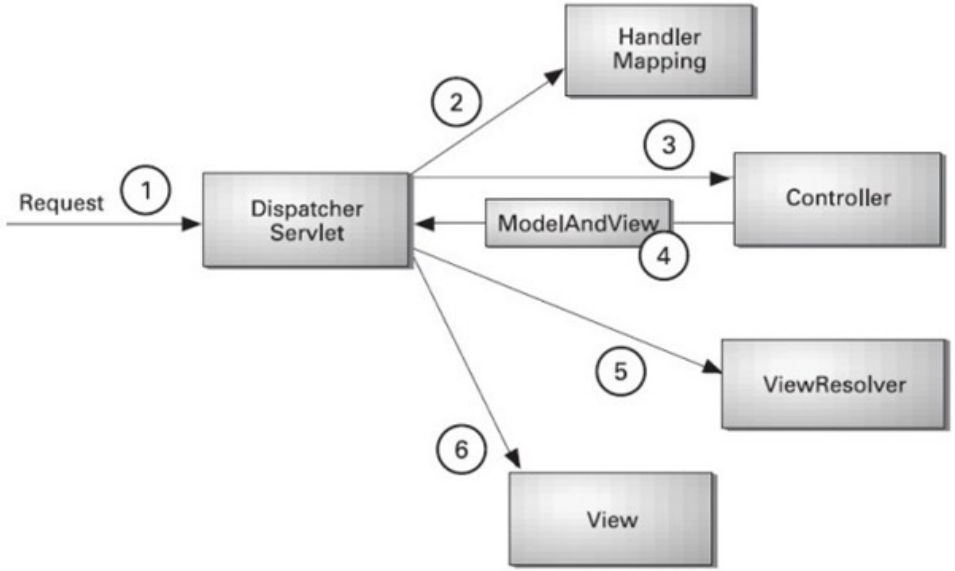
Spring MVC

陈昊鹏

chen-hp@sjtu.edu.cn

Spring MVC

- The Spring Web model-view-controller (MVC) framework is designed around a *DispatcherServlet* that handles all the HTTP requests and responses.



DD Example: Create a web controller

- HelloController.java

```
package hello.controller;

import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.Controller;

public class HelloController implements Controller {

   public ModelAndView handleRequest(javax.servlet.http.HttpServletRequest
            httpServletRequest, javax.servlet.http.HttpServletResponse
            httpServletResponse) throws Exception {
      ModelAndView mav = new ModelAndView("index.jsp");
      mav.addObject("message", "Hello Spring MVC");
      return mav;
   }
}
```

DD Example: Homepage index.jsp

REliable, INtelligent & Scalable Systems

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
 <head>
  <title>$Title$</title>
 </head>
 <body>
  <h1>${message}</h1>
 </body>
</html>
```

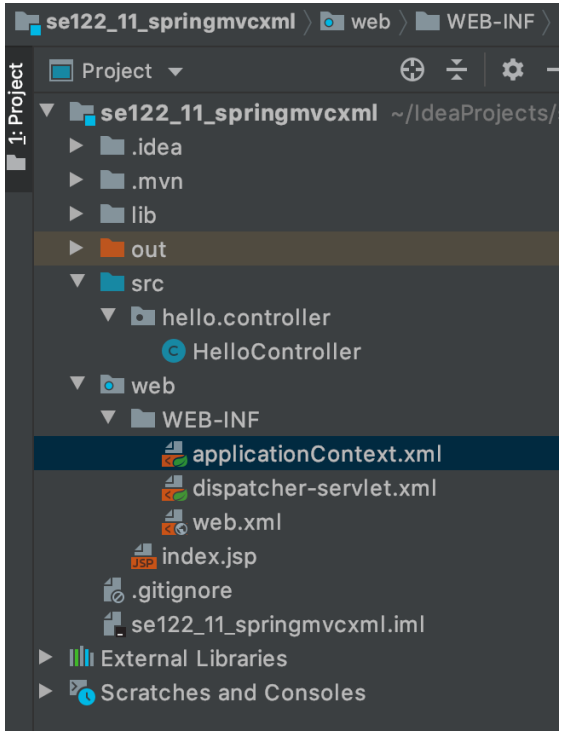
DD Example: web.xml


```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/applicationContext.xml</param-value>
  </context-param>
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
  <servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
</web-app>
```

DD Example: dispatcher-servlet.xml


```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">
  <bean id="simpleUrlHandlerMapping"
      class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
      <props>
        <prop key="/hello">helloController</prop>
      </props>
    </property>
  </bean>
  <bean id="helloController" class="hello.controller.HelloController"></bean>
</beans>
```

Deployment Descriptor Example



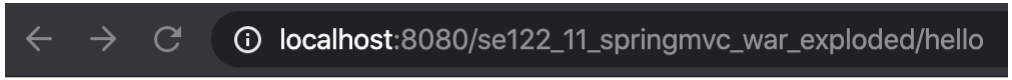
Hello, World!

Another view: hello.jsp


```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
 <head>
  <title>$Title$</title>
 </head>
 <body>
  <h1 style="color:red" >${message}</h1>
 </body>
</html>
```

- HelloController.java

```
ModelAndView mav = new ModelAndView("hello.jsp");
```



Hello Spring MVC

Annotation Example: Application

- MvcsampleApplication.java

```
package org.reins.mvcsample;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MvcsampleApplication {

   public static void main(String[] args) {
      SpringApplication.run(MvcsampleApplication.class, args);
   }

}
```

Annotation Example: Controller

- GreetingController.java

```
package org.reins.mvcsample;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class GreetingController {

  @GetMapping("/greeting")
  public String greeting(@RequestParam(name="name",
                         required=false, defaultValue="World") String name, Model model) {
    model.addAttribute("name", name);
    return "greeting";
  }

}
```

Annotation Example: View

- index.html

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Getting Started: Serving Web Content</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
  <p>Get your greeting <a href="/greeting">here</a></p>
</body>
</html>
```

Annotation Example: View

- greeting.html

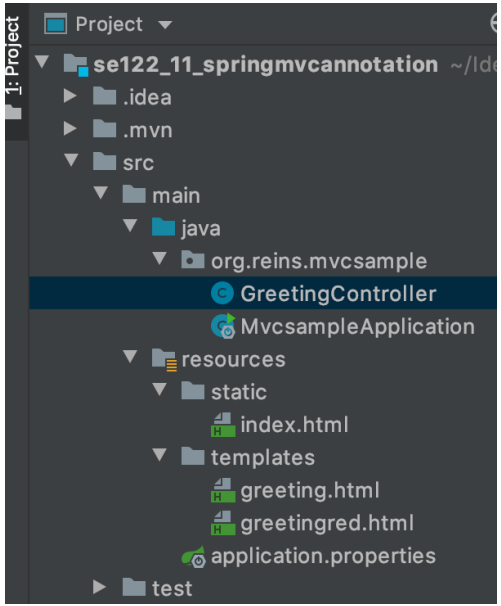
```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Getting Started: Serving Web Content</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
  <p th:text="'Hello, ' + ${name} + '!'" />
</body>
</html>
```

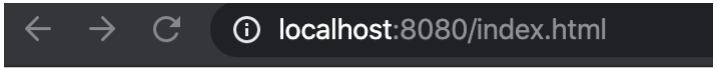
Thymeleaf

- **Thymeleaf** is a modern server-side Java template engine for both web and standalone environments.

```
<table>
  <thead>
    <tr>
      <th th:text="#{msgs.headers.name}">Name</th>
      <th th:text="#{msgs.headers.price}">Price</th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="prod: ${allProducts}">
      <td th:text="${prod.name}">Oranges</td>
      <td th:text="${#numbers.formatDecimal(prod.price, 1, 2)}">0.99</td>
    </tr>
  </tbody>
</table>
```

Annotation Example: View





Get your greeting here

Hello, World!

Another View: greetingread.html


```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Getting Started: Serving Web Content</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
  <p style="color:red" th:text="'Hello, ' + ${name} + '!'" />
</body>
</html>
```

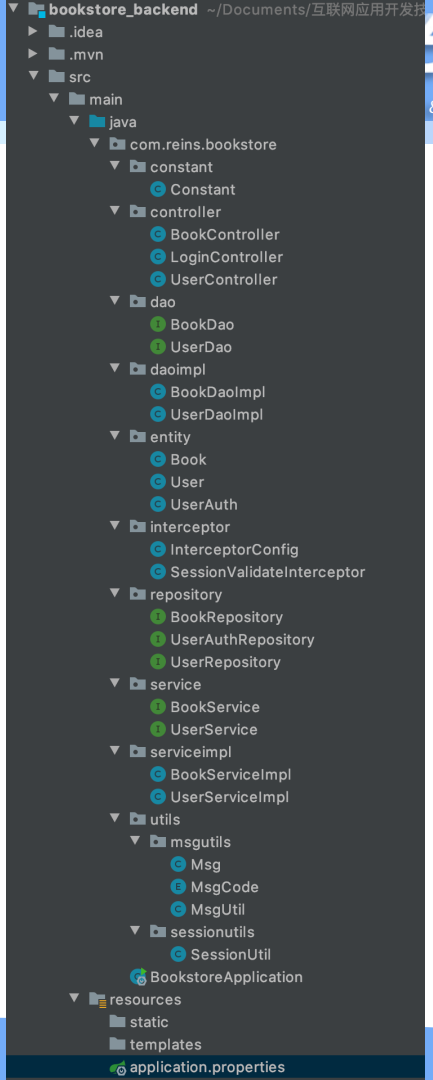
Hello, World!

- GreetingController.java

```
public String greeting(@RequestParam(name="name", required=false,
                defaultValue="World") String name, Model model) {
  model.addAttribute("name", name);
  return "greetingred";
}
```

Architecture

- Layered Architecture
  - Separation of Interface and Implementation
  - Entity – Auto mapped from database schema
  - Repository – Extended from existing lib class
  - Dao – Your own access control logic
  - Service – Business logic
  - Controller – Dispatch requests
  - IoC/DI – Independent of implementation



Annotated Controllers

- Spring MVC
  - provides an annotation-based programming model where @Controller and @RestController components use annotations to express request mappings, request input, exception handling, and more

```
@Controller
public class HelloController {

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

Annotated Controllers

- Declaration
  - To enable auto-detection of such @Controller beans, you can add component scanning to your Java configuration

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

```xml
<?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=" http://www.springframework.org/schema/beans
         https://www.springframework.org/schema/beans/spring-beans.xsd
         http://www.springframework.org/schema/context
         https://www.springframework.org/schema/context/spring-context.xsd">
  <context:component-scan base-package="org.example.web"/>
  <!-- ... -->
</beans>
```

Annotated Controllers

- Request Mapping
  - You can use the @RequestMapping annotation to map requests to controllers methods.
  - There are also HTTP method specific shortcut variants of @RequestMapping:
    - @GetMapping
    - @PostMapping
    - @PutMapping
    - @DeleteMapping
    - @PatchMapping

```
@RestController
@RequestMapping("/persons")
class PersonController {

  @GetMapping("/{id}")
  public Person getPerson(@PathVariable Long id) {
    // ...
  }

  @PostMapping
  @ResponseStatus(HttpStatus.CREATED)
  public void add(@RequestBody Person person) {
    // ...
  }
}
```

Annotated Controllers

- Request Mapping
  - URI patterns
    - ? matches one character
    - * matches zero or more characters within a path segment
    - ** match zero or more path segments

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

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

Annotated Controllers

- @RequestParam
  - You can use the @RequestParam annotation to bind Servlet request parameters (that is, query parameters or form data) to a method argument in a controller.

```
@Controller
@RequestMapping("/pets")
public class EditPetForm {
  // ...
  @GetMapping
  public String setupForm(@RequestParam("petId") int petId, Model model) {
    Pet pet = this.clinic.loadPet(petId);
    model.addAttribute("pet", pet);
    return "petForm";
  }
  // ...
}
```

CORS Filter

- The @CrossOrigin annotation enables cross-origin requests on annotated controller methods,

```
@RestController
@RequestMapping("/account")
public class AccountController {

  @CrossOrigin
  @GetMapping("/{id}")
  public Account retrieve(@PathVariable Long id) {
    // ...
  }

  @DeleteMapping("/{id}")
  public void remove(@PathVariable Long id) {
    // ...
  }
}
```

CORS Filter

- The @CrossOrigin annotation enables cross-origin requests on annotated controller methods,

```
@CrossOrigin(origins = "https://domain2.com", maxAge = 3600)
@RestController
@RequestMapping("/account")
public class AccountController {

  @GetMapping("/{id}")
  public Account retrieve(@PathVariable Long id) {
    // ...
  }

  @DeleteMapping("/{id}")
  public void remove(@PathVariable Long id) {
    // ...
  }
}
```

CORS Filter

- To enable CORS in the MVC Java config, you can use the CorsRegistry callback

```
@Configuration
@EnableWebMvc
public class WebConfig implements WebMvcConfigurer {
  @Override
  public void addCorsMappings(CorsRegistry registry) {
     registry.addMapping("/api/**")
        .allowedOrigins("https://domain2.com")
        .allowedMethods("PUT", "DELETE")
        .allowedHeaders("header1", "header2", "header3")
        .exposedHeaders("header1", "header2")
        .allowCredentials(true).maxAge(3600);
     // Add more mappings...
  }
}
```

CORS Filter

- You can apply CORS support through the built-in CorsFilter.

```
CorsConfiguration config = new CorsConfiguration();

// Possibly...
// config.applyPermitDefaultValues()
config.setAllowCredentials(true);
config.addAllowedOrigin("https://domain1.com");
config.addAllowedHeader("*");
config.addAllowedMethod("*");

UrlBasedCorsConfigurationSource source = new
                                    UrlBasedCorsConfigurationSource();
source.registerCorsConfiguration("/**", config);
CorsFilter filter = new CorsFilter(source);
```

Stereotype Annotations

- Spring provides stereotype annotations:
  - @Component, @Repository, @Service, and @Controller.
  - @Component is a generic stereotype for any Spring-managed component.
  - @Repository, @Service, and @Controller are specializations of @Component for more specific use cases (in the persistence, service, and presentation layers, respectively).
  - Therefore, you can annotate your component classes with @Component
  - But, by annotating them with @Repository, @Service, or @Controller instead, your classes are more properly suited for processing by tools or associating with aspects.
  - If you are choosing between using @Component or @Service for your service layer, @Service is clearly the better choice.

References

- Serving Web Content with Spring MVC
  - https://spring.io/guides/gs/serving-web-content/
- Spring MVC项目中直接使用idea创建spring mvc项目报错
  - https://blog.csdn.net/weixin_41060905/article/details/86911172
- Spring MVC【入门】就这一篇！
  - https://www.jianshu.com/p/91a2d0a1e45a
- Spring MVC Tutorial
  - https://www.javatpoint.com/spring-mvc-tutorial
- Thymeleaf Tutorial: Using Thymeleaf
  - https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html
- 注解@controller@service@component@repository区别
  - https://blog.csdn.net/u011305680/article/details/51701371
- Spring典型注解@Controller、@Service、@Component、@Repository
  - https://www.jianshu.com/p/5b8da1e8718b

- *Web开发技术*
- *Web Application Development*

Thank You!