第2-1课：Spring Boot 对基础 Web 开发的支持（上）

> 《精通 Spring Boot 42 讲》共分五大部分，这是第二部分内容，主要讲解了 Web 开发相关的技术点，共 10 课。首先让大家快速认识 Spring Boot 对基础 Web 开发所做的优化；接下来讲解了如何在 Spring Boot 中使用前端技术 JSP、Thymeleaf，重点介绍了 Thymeleaf 的各种使用场景；后面几课介绍了如何使用 Spring Boot 来构建 RESTful 服务、RESTful APIs，利用 WebSocket 双向通信的特性创建聊天室。

自从 B/S 架构（Browser/Server，浏览器/服务器模式）被发明以来，因为其具有跨平台、易移植、方便使用等特点，迅速地成为了技术架构的首选，前端 Web 技术迅速发展起来。人们利用前端 Web 技术构建各种应用场景，如电子商务平台、在线聊天室、后台管理系统等。页面技术也从最初的 JSP 演化为现在的模板引擎；信息交互由以前的 XML 发展到现在更流行的 JSON；Spring Filter、IoC、Aop 等概念的发展更加方便人们构建 Web 系统。

Spring Boot 对 Web 开发的支持很全面，包括开发、测试和部署阶段都做了支持。spring-boot-starter-web 是 Spring Boot 对 Web 开发提供支持的组件，主要包括 RESTful，参数校验、使用 Tomcat 作为内嵌容器等功能，接下来给大家一一介绍。

JSON 的支持

JSON (JavaScript Object Notation) 是一种轻量级的数据交换格式，易于阅读和编写，同时也易于机器解析和生成。JSON 采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯（包括 C、C++、C#、Java、JavaScript、Perl、Python 等），这些特性使 JSON 成为理想的数据交换语言。

早期人们习惯使用 XML 进行信息交互，后来 JSON 的使用更加简单，到了现在信息交互大部分都以 JSON 为主。早期在 Spring 体系中使用 JSON 还比较复杂，需要配置多项 XML 和注解，现在在 Spring Boot 体系中，对 JSON 支持简单而又完善，在 Web 层使用仅仅只需要一个注解即可完成。接下来使用案例说明。

新建一个 spring-boot-web 项目，在 pom.xml 中添加 Web 依赖。

```
<dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

启动类：

```
@SpringBootApplication
public class WebApplication {

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

在项目根路径下新建 model 包，在包下新建一个实体类 User，User 信息如下：

```
public class User {
    private String name;
    private int age;
    private String pass;
    //setter、getter省略
}
```

在项目中新建 Web 包，并在 Web 包下新建一个类 WebController，在类中创建一个方法返回 User，如下：

```
@RestController
public class WebController {
    @RequestMapping(name="/getUser", method= RequestMethod.POST)
    public User getUser() {
        User user=new User();
        user.setName("小明");
        user.setAge(12);
        user.setPass("123456");
        return user;
    }
}
```

- @RestController 注解相当于 @ResponseBody + @Controller 合在一起的作用，如果 Web 层的类上使用了 @RestController 注解，就代表这个类中所有的方法都会以 JSON 的形式返回结果，也相当于 JSON 的一种快捷使用方式；
- @RequestMapping(name="/getUser", method= RequestMethod.POST)，以 /getUser 的方式去请求，method= RequestMethod.POST 是指只可以使用 Post 的方式去请求，如果使用 Get 的方式去请求的话，则会报 405 不允许访问的错误。

在 test 包下新建 WebControllerTest 测试类，对 getUser() 方法进行测试。

```
@SpringBootTest
public class WebControllerTest {
    //省略部分代码
    @Test
    public void getUser() throws Exception {
        String responseString = mockMvc.perform(MockMvcRequestBuilders.post("/getU
ser"))
                .andReturn().getResponse().getContentAsString();
        System.out.println("result : "+responseString);
    }
}
```

这里的测试代码和上面的稍有区别，“.andReturn().getResponse().getContentAsString();”的意思是获取请求的返回信息，并将返回信息转换为字符串，最后将请求的响应结果打印出来。

执行完 Test，返回结果如下：

```
result : {"name":"小明","age":12,"pass":"123456"}
```

说明 Spring Boot 自动将 User 对象转成了 JSON 进行返回。那么如果返回的是一个 list 呢，在 WebController 添加方法 getUsers()，代码如下：

```
@RequestMapping("/getUsers")
public List<User> getUsers() {
    List<User> users=new ArrayList<User>();
    User user1=new User();
    user1.setName("neo");
    user1.setAge(30);
    user1.setPass("neo123");
    users.add(user1);
    User user2=new User();
    user2.setName("小明");
    user2.setAge(12);
    user2.setPass("123456");
    users.add(user2);
   return users;
}
```

添加测试方法：

```
@Test
public void getUsers() throws Exception {
    String responseString = mockMvc.perform(MockMvcRequestBuilders.get("/getUsers"
))
            .andReturn().getResponse().getContentAsString();
    System.out.println("result : "+responseString);
}
```

执行测试方法，返回内容如下：

```
result : [{"name":"neo","age":30,"pass":"neo123"},{"name":"小明","age":12,"pass":"1
23456"}]
```

说明不管是对象还是集合或者对象嵌套，Spring Boot 均可以将其转换为 JSON 字符串，这种方式特别适合系统提供接口时使用。

请求传参

请求传参是 Web 开发最基础的内容，前端浏览器和后端服务器正是依赖交互过程中的参数，来判断用户是否登录、用户正在执行时哪些动作，因此参数的传递和接收是一个 Web 系统最基础的功能。Spring Boot 内置了很多种参数接收方式，提供一些注解来帮助限制请求的类型、接收不同格式的参数等，接下来我们举例介绍。

使用 Spring Boot 可以轻松的对请求做一些限制，比如为了安全我们只允许 POST 请求的访问，只需要在方法上添加一个配置既可：

```
@RequestMapping(name="/getUser", method= RequestMethod.POST)
public User getUser() {
    ...
}
```

这时候再以 GET 请求去访问，就会返回如下结果：

```
Request method 'GET' not supported
```

如果要求其请求只支持 GET 请求，method 设置为：method= RequestMethod.GET；如果不进行设置默认两种方式的请求都支持。

Spring Web 层支持多种方法传参，在上一课中我们传入一个属性 name，直接使用对象接收也可以支持。

```
@RequestMapping(name="/getUser", method= RequestMethod.POST)
public String getUser(User user) {
    ...
}
```

这样的写法，只要是 User 的属性都会被自动填充到 User 对象中；还有另外一种传参的方式，使用 URL 进行传参，这种形式的传参地址栏会更加美观一些。

```
@RequestMapping(value="get/{name}", method=RequestMethod.GET)
public String get(@PathVariable String name) {
    return name;
}
```

在浏览器中输入网址：http://localhost:8080/get/neo，返回：neo，说明 name 值已经成功传入。

下一课我们继续讲解“数据校验”、“自定义 Filter”、“配置文件”等内容。