



Spring Boot 快速入门

2015年08月22日 21:59:18 茶爸爸 阅读数：287400 标签：spring boot 更多

今天给大家介绍一下Spring Boot MVC，让我们学习一下如何利用Spring Boot快速的搭建一个简单的web应用。

环境准备

- 一个称手的文本编辑器（例如Vim、Emacs、Sublime Text）或者IDE（Eclipse、Idea IntelliJ）
- Java环境（JDK 1.7或以上版本）
- Maven 3.0+（Eclipse和Idea IntelliJ内置，如果使用IDE并且不使用命令行工具可以不安装）

一个最简单的Web应用

使用Spring Boot框架可以大大加速Web应用的开发过程，首先在Maven项目依赖中引入spring-boot-starter-web：

pom.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5
6     <groupId>com.tianmaying</groupId>
7     <artifactId>spring-web-demo</artifactId>
8     <version>0.0.1-SNAPSHOT</version>
9     <packaging>jar</packaging>
10
11     <name>spring-web-demo</name>
12     <description>Demo project for Spring WebMvc</description>
13
14     <parent>
15         <groupId>org.springframework.boot</groupId>
16         <artifactId>spring-boot-starter-parent</artifactId>
17         <version>1.2.5.RELEASE</version>
18         <relativePath/>
19     </parent>
20
21     <properties>
22         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
23         <java.version>1.8</java.version>
24     </properties>
25
26     <dependencies>
27         <dependency>
28             <groupId>org.springframework.boot</groupId>
29             <artifactId>spring-boot-starter-web</artifactId>
30         </dependency>
31     </dependencies>
32
33     <build>
34         <plugins>
35             <plugin>
36                 <groupId>org.springframework.boot</groupId>
37                 <artifactId>spring-boot-maven-plugin</artifactId>
38             </plugin>
39         </plugins>
40     </build>
41
42
43 </project>
```

接下来创建src/main/java/Application.java:

```
1 import org.springframework.boot.SpringApplication;
2 import org.springframework.boot.autoconfigure.SpringBootApplication;
3 import org.springframework.web.bind.annotation.RequestMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @SpringBootApplication
7 @RestController
8 public class Application {
9
10     @RequestMapping("/")
11     public String greeting() {
12         return "Hello World!";
13     }
14
15     public static void main(String[] args) {
16         SpringApplication.run(Application.class, args);
17     }
18 }
```

运行应用：`mvn spring-boot:run`或在IDE中运行`main()`方法，在浏览器中访问`http://localhost:8080`，`Hello World!`就出现在了页面中。只用了区区十几行Java代码，一个Hello World应用就可以正确运行了，那么这段代码究竟做了什么呢？我们从程序的入口`SpringApplication.run(Application.class, args)`开始分析：

1. `SpringApplication`是Spring Boot框架中描述Spring应用的类，它的`run()`方法会创建一个Spring应用上下文（Application Context）。另一方面它会扫描当前应用类路径上的依赖，例如本例中发现`spring-webmvc`（由`spring-boot-starter-web`传递引入）在类路径中，那么Spring Boot会判断这是一个Web应用，并启动一个内嵌的Servlet容器（默认是Tomcat）用于处理HTTP请求。
2. Spring WebMvc框架会将Servlet容器里收到的HTTP请求根据路径分发给对应的`@Controller`类进行处理，`@RestController`是一类特殊的`@Controller`，它的返回值直接作为HTTP Response的Body部分返回给浏览器。
3. `@RequestMapping`注解表明该方法处理那些URL对应的HTTP请求，也就是我们常说的URL路由（routing），请求的分发工作是有Spring完成的。例如上面的代码中`http://localhost:8080/`根路径就被路由至`greeting()`方法进行处理。如果访问`http://localhost:8080/hello`，则会出现`404 Not Found`错误，因为我们并没有编写任何方法来处理`/hello`请求。

使用@Controller实现URL路由

现代Web应用往往包括很多页面，不同的页面也对应着不同的URL。对于不同的URL，通常需要不同的方法进行处理并返回不同的内容。

匹配多个URL

```
1 @RestController
2 public class Application {
3
4     @RequestMapping("/")
5     public String index() {
6         return "Index Page";
7     }
8
9     @RequestMapping("/hello")
10    public String hello() {
11        return "Hello World!";
12    }
13 }
```

`@RequestMapping`可以注解`@Controller`类：

```
1 @RestController
2 @RequestMapping("/classPath")
3 public class Application {
4     @RequestMapping("/methodPath")
5     public String method() {
6         return "mapping url is /classPath/methodPath";
7     }
8 }
```

`method`方法匹配的URL是`/classPath/methodPath`。

可以定义多个`@Controller`将不同URL的处理方法分散在不同的类中

URL中的变量——PathVariable

在Web应用中URL通常不是一成不变的，例如微博两个不同用户的个人主页对应两个不同的URL：`http://weibo.com/user1`，`http://weibo.com/user2`。我们不可能对于每一个用户都编写一个被`@RequestMapping`注解的方法来处理其请求，Spring MVC提供了一套机制来处理这种情况：

```
1 @RequestMapping("/users/{username}")
2 public String userProfile(@PathVariable("username") String username) {
3     return String.format("user %s", username);
4 }
5
6 @RequestMapping("/posts/{id}")
7 public String post(@PathVariable("id") int id) {
8     return String.format("post %d", id);
9 }
```

在上述例子中，URL中的变量可以用`{variableName}`来表示，同时在方法的参数中加上`@PathVariable("variableName")`，那么当请求被转发给该方法处理时，对应的URL中的变量会被自动赋值给被`@PathVariable`注解的参数（能够自动根据参数类型赋值，例如上例中的`int`）。

支持HTTP方法

对于HTTP请求除了其URL，还需要注意它的方法（Method）。例如我们在浏览器中访问一个页面通常是GET方法，而表单的提交一般是POST方法。`@Controller`中的方法同样需要对其进行区分：

```
1 @RequestMapping(value = "/login", method = RequestMethod.GET)
2 public String loginGet() {
3     return "Login Page";
4 }
5
6 @RequestMapping(value = "/login", method = RequestMethod.POST)
7 public String loginPost() {
8     return "Login Post Request";
9 }
```

模板渲染

在之前所有的`@RequestMapping`注解的方法中，返回值字符串都被直接传送到浏览器端并显示给用户。但是为了能够呈现更加丰富、美观的页面，我们需要将HTML代码返回给浏览器，浏览器再进行页面的渲染、显示。

一种很直观的方法是在处理请求的方法中，直接返回HTML代码，但是这样做的问题在于——一个复杂的页面HTML代码往往也非常复杂，并且嵌入在Java代码中十分不利于维护。更好的做法是将页面的HTML代码写在模板文件中，渲染后再返回给用户。为了能够进行模板渲染，需要将`@RestController`改成`@Controller`：

```
1 import org.springframework.ui.Model;
2
3 @Controller
4 public class HelloController {
5
6     @RequestMapping("/hello/{name}")
7     public String hello(@PathVariable("name") String name, Model model) {
8         model.addAttribute("name", name);
9         return "hello"
10     }
11 }
```

在上述例子中，返回值`"hello"`并非直接将字符串返回给浏览器，而是寻找名字为`hello`的模板进行渲染，我们使用Thymeleaf模板引擎进行模板渲染，需要引入依赖：

接下来需要在默认模板文件夹`src/main/resources/templates/`目录下添加一个模板文件`hello.html`：

复制

处理静态文件

Python爬虫全栈教学，零基础教你成编程大神

想对作者说点什么


[objc]

- 感谢作者的无私分享，分享目前主流的技术干货教程：SpringBoot+SpringCloud(2 0 1 7 最新微服务系列)、Docker容器、Hadoop Spark(大数据)、RocketMq、dubbo、redis分布式、数据库性能调优、Nginx入门实战高级视频教程、SSM框架等等。
- <https://itstorage.github.io/java/goods.html>


(3天前 #47楼)


 yongjiu_smile: 很不错, 值得学习! (10个月前 #45楼)


 **CSDN | 你的笑忘书：** 为什么你的Markdown写出来的代码 `` 代码 ``，还有`，渲染出来是这样的，我的代码字体是微软雅黑。好不协调。 (10个月前 #43楼)

 **遇见_青葱**: 什么东西都不需要改, 就可以运行, 谢谢楼主 (10个月前 #42楼)

 lmfxrj : Check your ViewResolver setup! (Hint: This may be the result of an unspecified view, due to default view name generation.) 如何解决 是不是找不到模板的位置呀
(10个月前 #40楼) [查看回复\(1\)](#)

 jameskaron : 谢谢分享 (11个月前 #38楼)

 leeroy白：非常感谢!学到了不少东西 (11个月前 #37楼)

 mr_dzhao : 不错的文章 (1年前 #36楼)

 一缕清风o：Spring boot教程：<https://blog.yoodb.com/springboot/tutorial> 有问题请留言，及时解答。 (1年前 #35楼)

 **heng7758258** : Spring Boot教程大全等www.it448.com视频教程 (1年前 #34楼)