链滴

SpringBoot 构建复杂的 REST ful 风格 API

作者：Javen

原文链接：https://ld246.com/article/1497247011376

来源网站：链滴

首先，回顾并详细说明一下在快速入门中使用的@Controller、@RestController、@RequestMapping注解。如果您对Spring MVC不熟悉并且还没有尝试过快速入门案例，建议先看一下快速入门的内。

@Controller：修饰class，用来创建处理http请求的对象

@RestController：Spring4之后加入的注解，原来在@Controller中返回json需要@ResponseBod来配合，如果直接用@RestController替代@Controller就不需要再配置@ResponseBody，默认返回son格式。

@RequestMapping：配置url映射

下面我们尝试使用Spring MVC来实现一组对User对象操作的RESTful API，配合注释详细说明在Sprig MVC中如何映射HTTP请求、如何传参、如何编写单元测试。

RESTful API具体设计如下：

User实体定义：

```
public class User {

    private Long id;
    private String name;
    private Integer age;

    // 省略setter和getter

}
```

实现对User对象的操作接口

```
@RestController
@RequestMapping(value="/users")     // 通过这里配置使下面的映射都在/users下
public class UserController {

    // 创建线程安全的Map
    static Map<Long, User> users = Collections.synchronizedMap(new HashMap<Long, Use
>());

    @RequestMapping(value="/", method=RequestMethod.GET)
    public List<User> getUserList() {
        // 处理"/users/"的GET请求，用来获取用户列表
        // 还可以通过@RequestParam从页面中传递参数来进行查询条件或者翻页信息的传递
        List<User> r = new ArrayList<User>(users.values());
        return r;
    }

    @RequestMapping(value="/", method=RequestMethod.POST)
    public String postUser(@ModelAttribute User user) {
        // 处理"/users/"的POST请求，用来创建User
        // 除了@ModelAttribute绑定参数之外，还可以通过@RequestParam从页面中传递参数
        users.put(user.getId(), user);
        return "success";
```

```
    }

    @RequestMapping(value="/{id}", method=RequestMethod.GET)
    public User getUser(@PathVariable Long id) {
        // 处理"/users/{id}"的GET请求，用来获取url中id值的User信息
        // url中的id可通过@PathVariable绑定到函数的参数中
        return users.get(id);
    }

    @RequestMapping(value="/{id}", method=RequestMethod.PUT)
    public String putUser(@PathVariable Long id, @ModelAttribute User user) {
        // 处理"/users/{id}"的PUT请求，用来更新User信息
        User u = users.get(id);
        u.setName(user.getName());
        u.setAge(user.getAge());
        users.put(id, u);
        return "success";
    }

    @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
    public String deleteUser(@PathVariable Long id) {
        // 处理"/users/{id}"的DELETE请求，用来删除User
        users.remove(id);
        return "success";
    }

}
```

下面针对该Controller编写测试用例验证正确性，具体如下。当然也可以通过浏览器插件等进行请求交验证。

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(classes = MockServletContext.class)
@WebAppConfiguration
public class ApplicationTests {

    private MockMvc mvc;

    @Before
    public void setUp() throws Exception {
        mvc = MockMvcBuilders.standaloneSetup(new UserController()).build();
    }

    @Test
    public void testUserController() throws Exception {
        // 测试UserController
        RequestBuilder request = null;

        // 1、get查一下user列表，应该为空
        request = get("/users/");
        mvc.perform(request)
                .andExpect(status().isOk())
                .andExpect(content().string(equalTo("[]")));
```

```
        // 2、post提交一个user
        request = post("/users/")
                .param("id", "1")
                .param("name", "测试大师")
                .param("age", "20");
        mvc.perform(request)
                .andExpect(content().string(equalTo("success")));

        // 3、get获取user列表，应该有刚才插入的数据
        request = get("/users/");
        mvc.perform(request)
                .andExpect(status().isOk())
                .andExpect(content().string(equalTo("[{\"id\":1,\"name\":\"测试大师\",\"age\":20}]")
);

        // 4、put修改id为1的user
        request = put("/users/1")
                .param("name", "测试终极大师")
                .param("age", "30");
        mvc.perform(request)
                .andExpect(content().string(equalTo("success")));

        // 5、get一个id为1的user
        request = get("/users/1");
        mvc.perform(request)
                .andExpect(content().string(equalTo("{\"id\":1,\"name\":\"测试终极大师\",\"age\":3
}")));

        // 6、del删除id为1的user
        request = delete("/users/1");
        mvc.perform(request)
                .andExpect(content().string(equalTo("success")));

        // 7、get查一下user列表，应该为空
        request = get("/users/");
        mvc.perform(request)
                .andExpect(status().isOk())
                .andExpect(content().string(equalTo("[]")));

    }

}
```

至此，我们通过引入web模块（没有做其他的任何配置），就可以轻松利用Spring MVC的功能，以常简洁的代码完成了对User对象的RESTful API的创建以及单元测试的编写。其中同时介绍了Spring VC中最为常用的几个核心注解：@Controller,@RestController,RequestMapping以及一些参数绑的注解：@PathVariable,@ModelAttribute,@RequestParam等。