星朝

昵称： 星朝
园龄： 3年11个月
粉丝： 1246
关注： 2
+加关注

< 2021年2月 >

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	1	2	3	4	5	6
7	8	9	10	11	12	13

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签
java经验集锦(3676)
通用实践(3524)
SQL(786)
Mysql(715)
DB(698)
JDK(576)
SpringBoot(549)
高并发(537)
性能优化(488)
Spring(485)

SpringBoot基础之MockMvc单元测试

SpringBoot创建的Maven项目中，会默认添加spring-boot-starter-test依赖。在《5分钟快速上手SpringBoot》中编写的单元测试使用了MockMvc。本篇文章就围绕MockMvc在SpringBoot中的使用进行讲解。

什么是Mock

在面向对象的程序设计中，模拟对象（英语：mock object）是以可控的方式模拟真实对象行为的假对象。在编程过程中，通常通过模拟一些输入数据，来验证程序是否达到预期结果。

为什么使用Mock对象

使用模拟对象，可以模拟复杂的、真实的对象行为。如果在单元测试中无法使用真实对象，可采用模拟对象进行替代。

在以下情况可以采用模拟对象来替代真实对象：

- 真实对象的行为是不确定的（例如，当前的时间或温度）；
- 真实对象很难搭建起来；
- 真实对象的行为很难触发（例如，网络错误）；
- 真实对象速度很慢（例如，一个完整的数据库，在测试之前可能需要初始化）；
- 真实的对象是用户界面，或包括用户界面在内；
- 真实的对象使用了回调机制；
- 真实对象可能还不存在；
- 真实对象可能包含不能用作测试（而不是为实际工作）的信息和方法。

使用Mockito一般分三个步骤：1、模拟测试类所需的外部依赖；2、执行测试代码；3、判断执行结果是否达到预期；

MockMvc

MockMvc是由spring-test包提供，实现了对Http请求的模拟，能够直接使用网络的形式，转换到Controller的调用，使得测试速度快、不依赖网络环境。同时提供了一套验证的工具，结果的验证十分方便。

接口MockMvcBuilder，提供一个唯一的build方法，用来构造MockMvc。主要有两个实现：StandaloneMockMvcBuilder和DefaultMockMvcBuilder，分别对应两种测试方式，即独立安装和集成Web环境测试（并不会集成真正的web环境，而是通过相应的Mock API进行模拟测试，无须启动服务器）。MockMvcBuilders提供了对应的创建方法standaloneSetup方法和webAppContextSetup方法，在使用时直接调用即可。

SpringBoot中使用

第一步：jar包引入。创建SpringBoot项目中默认引入的spring-boot-starter-test间接引入了spring-test，因此无需再额外引入jar包。

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

第二步：创建HelloWorldController类，并提供hello方法作为待测试的业务接口。

```
@RestController
public class HelloWorldController {
```

更多

随笔分类

J2EE(40)
J2se(97)
JVM(26)
Markdown(2)
Maven(5)
Office办公(2)
OpenCMS(4)
Python(25)
Security(25)
Spring(15)
SpringMVC(67)
SQL(614)
Struts2(1)
WebServer(4)
Web前端(73)
更多

随笔档案

2020年4月(1)
2020年1月(43)
2019年12月(175)
2019年11月(131)
2019年10月(95)
2019年9月(226)
2019年8月(209)
2019年7月(271)
2019年6月(395)
2019年5月(411)
2019年4月(113)
2019年3月(21)
2019年2月(84)

```
@RequestMapping
public String hello(String name){
        return "Hello " + name + "!";
}


}
```

第三步：编写测试类。实例化MockMvc有两种形式，一种是使用StandaloneMockMvcBuilder，另外一种是使用DefaultMockMvcBuilder。测试类及初始化MockMvc初始化：

```
//SpringBoot1.4版本之前用的是SpringJUnit4ClassRunner.class
@RunWith(SpringRunner.class)
//SpringBoot1.4版本之前用的是@SpringApplicationConfiguration(classes = Application.class
@SpringBootTest
//测试环境使用，用来表示测试环境使用的ApplicationContext将是WebApplicationContext类型的
@WebAppConfiguration
public class HelloWorldTest {

    private MockMvc mockMvc;

    @Autowired
    private WebApplicationContext webApplicationContext;

    @Before
    public void setup() {
        // 实例化方式一
        mockMvc = MockMvcBuilders.standaloneSetup(new HelloWorldController()).build
        // 实例化方式二


//              mockMvc = MockMvcBuilders.webAppContextSetup(webApplicationContext).b

}
```

单元测试方法：

```
@Test
public void testHello() throws Exception {

    /*
     * 1、mockMvc.perform执行一个请求。
     * 2、MockMvcRequestBuilders.get("XXX")构造一个请求。
     * 3、ResultActions.param添加请求传值
     * 4、ResultActions.accept(MediaType.TEXT_HTML_VALUE))设置返回类型
     * 5、ResultActions.andExpect添加执行完成后的断言。
     * 6、ResultActions.andDo添加一个结果处理器，表示要对结果做点什么事情
     *   比如此处使用MockMvcResultHandlers.print()输出整个响应结果信息。
     * 7、ResultActions.andReturn表示执行完成后返回相应的结果。
     */
    mockMvc.perform(MockMvcRequestBuilders
                    .get("/hello")
                    // 设置返回值类型为utf-8，否则默认为ISO-8859-1
                    .accept(MediaType.APPLICATION_JSON_UTF8_VALUE)
                    .param("name", "Tom"))
                    .andExpect(MockMvcResultMatchers.status().isOk())
                    .andExpect(MockMvcResultMatchers.content().string("Hello Tom!"))
                    .andDo(MockMvcResultHandlers.print());


}
```

测试结果打印：

```
FlashMap:
       Attributes = null

MockHttpServletResponse:

Status = 200
```

2019年1月(141)
2018年12月(127)
更多

文章分类
JVM(4)
Markdown(4)
SQL(5)
开源项目(4)
人工智能(9)
算法(2)

最新评论
1. Re:动态表单的数据库结构设计
您好，请问有表设计可以参考一下吗？
--曾家琦
2. Re:Java泛型详解：<T>和Class<T>的使用。泛型类，泛型方法的详细使用实例
学习了
--伊文Statistics
3. Re:Java8 Date与LocalDate互转
中午睡个午觉
--君君的喵爸
4. Re:ANSI转UTF-8中文无乱码解决方案
成功解决
--qsq陌上花开
5. Re:Java子线程中的异常处理（通用）
讲的不错
--花溪的小石头

阅读排行榜
1. js判断对象是否为空对象的几种方法(282216)
2. JSON字符串转换为Map(165045)

```
Error message = null

Headers = [Content-Type:"application/json;charset=UTF-8", Content-Length:"10"]

Content type = application/json;charset=UTF-8

Body = Hello Tom!

Forwarded URL = null

Redirected URL = null

Cookies = []

2019-04-02 21:34:27.954  INFO 6937 --- [       Thread-2] o.s.s.concurrent.ThreadPoolT
```

整个过程如下：
1、准备测试环境
2、通过MockMvc执行请求
3、添加验证断言
4、添加结果处理器
5、得到MvcResult进行自定义断言/进行下一步的异步请求
6、卸载测试环境

注意事项：如果使用DefaultMockMvcBuilder进行MockMvc实例化时需在SpringBoot启动类上添加组件扫描的package的指定，否则会出现404。如：

```
@ComponentScan(basePackages = "com.secbro2")
```

相关API

RequestBuilder提供了一个方法buildRequest(ServletContext servletContext)用于构建MockHttpServletRequest；其主要有两个子类MockHttpServletRequestBuilder和MockMultipartHttpServletRequestBuilder（如文件上传使用）。

MockMvcRequestBuilders提供get、post等多种方法用来实例化RequestBuilder。

ResultActions，MockMvc.perform(RequestBuilder requestBuilder)的返回值，提供三种能力：andExpect，添加断言判断结果是否达到预期；andDo，添加结果处理器，比如示例中的打印；andReturn，返回验证成功后的MvcResult，用于自定义验证/下一步的异步处理。

一些常用的测试

1.测试普通控制器

```
mockMvc.perform(get("/user/{id}", 1)) //执行请求
            .andExpect(model().attributeExists("user")) //验证存储模型数据
            .andExpect(view().name("user/view")) //验证viewName
            .andExpect(forwardedUrl("/WEB-INF/jsp/user/view.jsp"))//验证视图渲染时forwar
            .andExpect(status().isOk())//验证状态码
            .andDo(print()); //输出MvcResult到控制台
```

2.得到MvcResult自定义验证

```
MvcResult result = mockMvc.perform(get("/user/{id}", 1))//执行请求
        .andReturn(); //返回MvcResult
Assert.assertNotNull(result.getModelAndView().getModel().get("user")); //自定义断言
```

3.验证请求参数绑定到模型数据及Flash属性

```
mockMvc.perform(post("/user").param("name", "zhang")) //执行传递参数的POST请求(也可以post
            .andExpect(handler().handlerType(UserController.class)) //验证执行的控制器类
            .andExpect(handler().methodName("create")) //验证执行的控制器方法名
            .andExpect(model().hasNoErrors()) //验证页面没有错误
            .andExpect(flash().attributeExists("success")) //验证存在flash属性
            .andExpect(view().name("redirect:/user")); //验证视图
```

4.文件上传

```
byte[] bytes = new byte[] {1, 2};
mockMvc.perform(fileUpload("/user/{id}/icon", 1L).file("icon", bytes)) //执行文件上传
```

3. Tor Browser（洋葱浏览器）——一款使你匿名上网的浏览器(148835)

4. Unsupported major.minor version 52.0解决办法(142862)

5. Shiro权限管理框架详解(141963)

评论排行榜

1. Java泛型详解：<T>和Class<T>的使用。泛型类，泛型方法的详细使用实例(11)

2. Shiro权限管理框架详解(9)

3. Mybatis Update操作返回值问题(6)

4. 树的前序遍历、中序遍历、后序遍历详解(5)

5. SpringBoot项目优化和Jvm调优(楼主亲测，真实有效)(5)

推荐排行榜

1. Shiro权限管理框架详解(25)

2. Java泛型详解：<T>和Class<T>的使用。泛型类，泛型方法的详细使用实例(14)

3. js判断对象是否为空对象的几种方法(13)

4. Spring中@Async用法总结(11)

5. 树的前序遍历、中序遍历、后序遍历详解(9)

```
        .andExpect(model().attribute("icon", bytes)) //验证属性相等性
        .andExpect(view().name("success")); //验证视图
```

5.JSON请求/响应验证

```
String requestBody = "{\"id\":1, \"name\":\"zhang\"}";
    mockMvc.perform(post("/user")
            .contentType(MediaType.APPLICATION_JSON).content(requestBody)
            .accept(MediaType.APPLICATION_JSON)) //执行请求
            .andExpect(content().contentType(MediaType.APPLICATION_JSON)) //验证响应co
            .andExpect(jsonPath("$.id").value(1)); //使用Json path验证JSON 请参考http:/
```

```
String errorBody = "{id:1, name:zhang}";
MvcResult result = mockMvc.perform(post("/user")
        .contentType(MediaType.APPLICATION_JSON).content(errorBody)
        .accept(MediaType.APPLICATION_JSON)) //执行请求
        .andExpect(status().isBadRequest()) //400错误请求
        .andReturn();

Assert.assertTrue(HttpMessageNotReadableException.class.isAssignableFrom(result.getI
```

6.异步测试

```
//Callable
    MvcResult result = mockMvc.perform(get("/user/async1?id=1&name=zhang")) //执行请求
            .andExpect(request().asyncStarted())
            .andExpect(request().asyncResult(CoreMatchers.instanceOf(User.class))) //
            .andReturn();
```

```
mockMvc.perform(asyncDispatch(result))
        .andExpect(status().isOk())
        .andExpect(content().contentType(MediaType.APPLICATION_JSON))
        .andExpect(jsonPath("$.id").value(1));
```

7.全局配置

```
mockMvc = webAppContextSetup(wac)
            .defaultRequest(get("/user/1").requestAttr("default", true)) //默认请求 如果
            .alwaysDo(print())  //默认每次执行请求后都做的动作
            .alwaysExpect(request().attribute("default", true)) //默认每次执行后进行验证的
            .build();
```

```
mockMvc.perform(get("/user/1"))
        .andExpect(model().attributeExists("user"));
```

```
    </div>
```

原文地址：https://blog.csdn.net/wo541075754/article/details/88983708

分类: 工作总结, 开发经验, 运维

标签: java经验集锦, 通用实践, Junit, SpringBoot, 单元测试, Hamcrest, Mock&Mockito

好文要顶 关注我 收藏该文

星朝
关注 - 2
粉丝 - 1246
+加关注

0 推荐 0 反对

« 上一篇： MockMvc详解
» 下一篇： Spring、Spring Boot和TestNG测试指南 - 使用Spring Boot Testing工具

posted @ 2019-05-30 17:32 星朝 阅读(652) 评论(0) 编辑 收藏

登录后才能发表评论，立即 登录 或 注册， 访问 网站首页

【推荐】阿里云Java训练营第一期：结合理论与实践，实现知识套现
【推荐】阿里云专家领学，2021数仓之旅免费必修课，加入学习
【推荐】阿里云寒假充电季，防疫不停学，21天打卡进阶Java工程师
【推荐】大型组态、工控、仿真、CADGIS 50万行VC++源码免费下载
【推荐】AWS携手博客园为开发者送福利，新用户立享12个月免费套餐
【推荐】第一个 NoSQL 数据库，在大规模和一致性之间找到了平衡
【推荐】Agora 教程 | 如何实现15mins自主搭建一个教育平台？

相关博文：
· 单元测试中模拟mvc测试对象MockMvc
· SpringBoot
· SpringBoot
· SpringBoot
· Springboot
» 更多推荐...

最新 IT 新闻:
· 理想汽车春节就地过年补贴曝光：每人1千现金补贴
· 三星将退出苏州面板工厂：给员工N+1补偿、年终奖照发
· 网易云音乐称酷狗抄袭：酷狗副总裁如此回应
· 菜鸟春节期间不打烊：小二发货可获得五六千元收入
· 马斯克谈脑机接口新进展：猴子已经能够用大脑操控游戏
» 更多新闻...

历史上的今天：
2019-05-30 免费tk域名+freewebhostingarea空间
2018-05-30 jquery动态为个span，input，div，等标签赋值的方法总结，js动态隐藏div
2018-05-30 页面加载即执行JQuery的三种方法
2018-05-30 开源的多行字符串工具： 在JS中整段地写HTML
2018-05-30 在JQuery中$(document.body)和这个$("body") 这两的区别在哪里？
2018-05-30 jquery追加元素的几种方法（append()、prepend()、after()、before()、insertA...
2018-05-30 jQuery中的text()、html()和val()以及innerText、innerHTML和value

Copyright © 2021 星朝
Powered by .NET 5.0 on Kubernetes