# MockMVC

通常我们在测试 Web 应用程序时，都会编写一些测试来断言应用程序的行为，常见的测试方式是通过启动应用程序并侦听连接（就像在生产中所做的那样），然后发送 HTTP 请求并断言响应。

另一种有用的方法是根本不启动服务器，而只测试其下的层。这里我们可以使用 SpringMockMvc 并通过使用@AutoConfigureMockMvc 在测试用例上注释为我们注入。它能够让 Spring 处理传入的 HTTP 请求，并将其传递给我们的控制器。这样，几乎整个堆栈都被使用了，而我们的代码将以与处理真实 HTTP 请求完全相同的方式调用，达到了无需启动服务器的效果。

在测试中，我们还可以使用@WebMvcTest 注解将测试范围缩小到仅 Web 层。从而无需启动完整的 Spring 应用程序上下文。如果在具有多个控制器的应用程序中，我们可以使用@WebMvcTest(UserController.class)来详细地指定我们测试的 web 层。

下面为 **“添加用户“** 功能做一个测试：

添加用户前的数据：

对象 | user @db_springboot (localhost - M...

开始事务 | 文本 | 筛选 | 排序 | 导入 | 导出

| user_id | user_name | user_pwd | create_date | modify_date | enable | version |
|---|---|---|---|---|---|---|
| 1 | 张三 | 123 | 2021-12-07 19:07:36 | 2021-12-22 23:17:49 | 1 | 1 |
| 2 | admin | admin | 2021-12-22 23:17:53 | 2021-12-22 23:42:17 | 1 | 1 |
| 9 | admin | 123 | 2021-12-23 00:33:53 | 2021-12-23 00:33:53 | 1 | 1 |

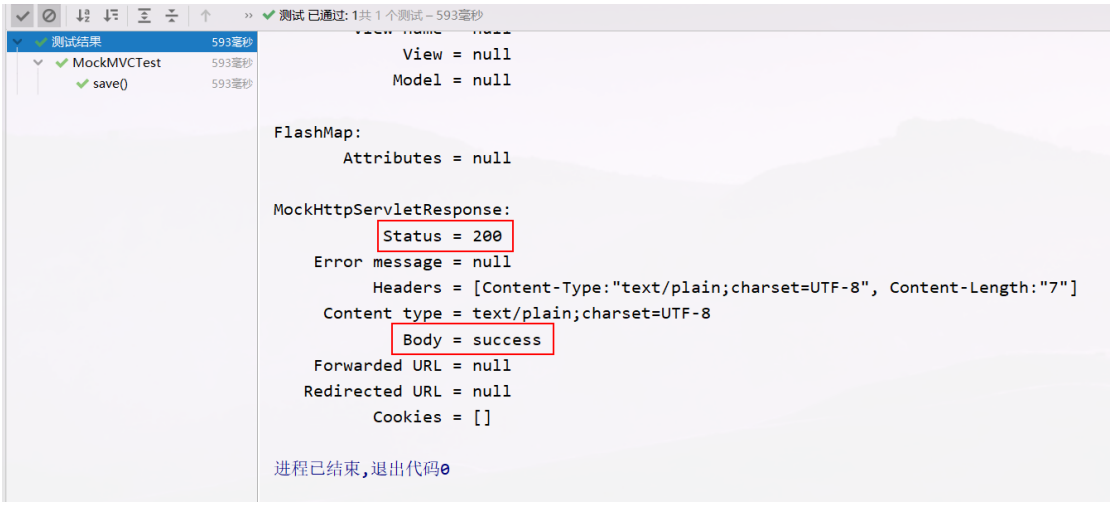
添加用户的测试代码：（UserController.java）

```
@RequestMapping("/test/save")
public String saveUser(@RequestBody User user){
    if(userService.save(user))return "success";
    else return "error";
}
```

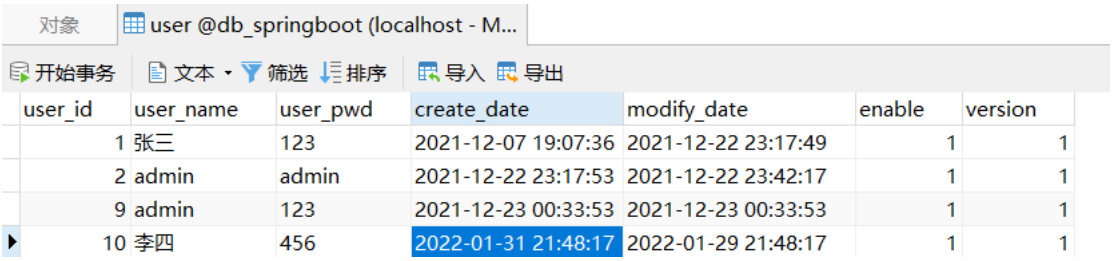
添加用户的测试代码:

```java
package com.study.webTest;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.study.springboot.controller.UserController;
import com.study.springboot.entity.User;
import com.study.springboot.service.UserService;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.http.MediaType;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.MvcResult;

import static org.mockito.ArgumentMatchers.isA;
import static org.mockito.Mockito.when;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
import static org.springframework.test.web.servlet.result.MockMvcResultHandlers.print;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

@WebMvcTest(UserController.class)//选择测试的控制器，让程序无需加载整个上下文
public class MockMVCTest {

    @MockBean
    private UserService userService;

    @Autowired
    private MockMvc mockMvc;//mock工具的依赖注入

    @Autowired
    ObjectMapper objectMapper;


    @Test
    public void save() throws Exception {
        /*
         * @author: 何翔
         * @param: []
         * @return: void
         * @date: 2021/12/31 14:42
         * @description: 使用mockMvc模拟请求测试添加用户
         */
        User user = new User();
        user.setUserName("李四");
        user.setUserPwd("456");
        when(userService.save(isA(User.class))).thenReturn(true);
       MvcResult result = mockMvc.perform(post( urlTemplate: "/springboot/user/test/save")
                .contentType(MediaType.APPLICATION_JSON)
                .content(objectMapper.writeValueAsString(user)))
                .andDo(print())
                .andExpect(status().isOk())
                .andReturn();
                System.out.println(result);
                this.mockMvc.perform(post( urlTemplate: "/springboot/user/test/save")
                                .contentType(MediaType.APPLICATION_JSON)
                                .content(objectMapper.writeValueAsString(user)))
                .andDo(print())
                .andExpect(status().isOk())
                .andExpect(jsonPath( expression: "$").value( expectedValue: "success"));

    }
}
```

测试成功:




数据查看:




| user_id | user_name | user_pwd | create_date | modify_date | enable | version |
|---|---|---|---|---|---|---|
| 1 | 张三 | 123 | 2021-12-07 19:07:36 | 2021-12-22 23:17:49 | 1 | 1 |
| 2 | admin | admin | 2021-12-22 23:17:53 | 2021-12-22 23:42:17 | 1 | 1 |
| 9 | admin | 123 | 2021-12-23 00:33:53 | 2021-12-23 00:33:53 | 1 | 1 |
| 10 | 李四 | 456 | 2022-01-31 21:48:17 | 2022-01-29 21:48:17 | 1 | 1 |

# 总结:

以往 controller 层的单元测试，通常只能使用浏览器测试，如果通过启动服务器，建立 http client 进行测试，这样会使得测试变得很麻烦，比如，启动速度慢，测试验证不方便，依赖网络环境等，这样会导致测试无法进行，为了可以对 Controller 进行测试，可以通过引入 MockMVC 进行解决。

MockMvc 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，这样可以使得测试速度快、不依赖网络环境，而且提供了一套验证的工具，这样可以使得请求的验证统一而且很方便。