



1

1



Spring_Boot_FirstExample [boot] [devtools]

- ▼ src/test/java
 - ▼ com.springboot_firstExample
 - > HelloControllerTest.java
 - ▼ domain
 - > HelloServiceTest.java

2

2

Testing the controller



```
package com.springboot_firstExample;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.model;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.view;

import org.junit.jupiter.api.Test;
import static org.mockito.Mockito.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.test.context.bean.override.mockito.MockitoBean;
import org.springframework.test.web.servlet.MockMvc;

import domain.HelloService;
```

3

3

@WebMvcTest(HelloController.class)

class HelloControllerTest {

//Injects MockMvc

@Autowired

private MockMvc mockMvc;

@MockitoBean

private HelloService mockService;

Class HelloController:

@Autowired

private HelloService helloService;

Dependency Injection



4

4

@Test

void testHelloGet() throws Exception {

```
    mockMvc.perform(get("/hello"))
        .andExpect(status().isOk())
        .andExpect(view().name("nameForm"))
        .andExpect(model().attributeExists("name"));
}
```

This test issues a **GET** request for **/hello**, asserts that the response has an **HTTP 200 (OK)** status, that the resulting **view** is named **nameForm** and that the **model** has an attribute named **name**.

5

5

```
...                                     nameForm.html
<form th:action="@{/hello}" th:object="${name}" method="post">
...
    <input type="text" th:field="*{value}" size="15"
                                   placeholder="name"/>
...

```

@Test

void testHelloPost() throws Exception {

```
    String expResult = "Hello testMock!";
    Mockito.when(mockService.sayHello("test")).thenReturn(expResult);

    mockMvc.perform(post("/hello").param("value", "test"))
        .andExpect(status().isOk())
        .andExpect(view().name("helloView"))
        .andExpect(model().attributeExists("helloMessage"))
        .andExpect(model().attribute("helloMessage", expResult));
```

}

6

6

OR

```
@Getter @Setter @NoArgsConstructor @AllArgsConstructor  
public class Name {
```

```
    private String value;
```

```
}
```

```
@PostMapping("/hello")  
public String onSubmit(Name name, Model model) {
```

@Test

```
void testHelloPost() throws Exception {  
    String expResult = "Hello testMock!";  
    Mockito.when(mockService.sayHello("test")).thenReturn(expResult);
```

```
//mockMvc.perform(post("/hello").param("value", "test"))  
mockMvc.perform(post("/hello").flashAttr("name", new Name("test")))
```

```
.andExpect(status().isOk())  
.andExpect(view().name("helloView"))  
.andExpect(model().attributeExists("helloMessage"))  
.andExpect(model().attribute("helloMessage", expResult));
```

```
}
```

7