<https://blog.okazuki.jp/entry/2015/07/14/205627>

Spring Bootは単体テスト機能もついて

ます。

[35. Testing](http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-testing.html)

Spring Bootでコントローラの単体テストをするには、まず単体テストのクラスをアノテーションでいろいろ飾ってやる必要があります。

package okazuki.validationEdu;

import org.junit.After;

import org.junit.Before;

import org.junit.Rule;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.Mockito;

import org.mockito.junit.MockitoJUnit;

import org.mockito.junit.MockitoRule;

import org.springframework.test.context.ContextConfiguration;

import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import org.springframework.test.context.web.WebAppConfiguration;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;

import org.springframework.test.web.servlet.result.MockMvcResultHandlers;

import org.springframework.test.web.servlet.result.MockMvcResultMatchers;

import org.springframework.test.web.servlet.setup.MockMvcBuilders;

@RunWith(SpringJUnit4ClassRunner.class)

@WebAppConfiguration

@ContextConfiguration(classes = App.class)

public class HomeControllerTest {

}

RunWithで実行をSpringのJUnitRunnerに切り替えて、Web系の設定をやるように指定して、最後にContextを設定します。ここでは本番用のAppクラスをそのまま指定していますが、テスト用の設定をしたクラスを指定してもOKです。

そして、コントローラのフィールドにモックを仕込みたい場合は、以下のようにMockitRuleというものをpublicなフィールドに用意します。

@Rule

public final MockitoRule rule = MockitoJUnit.rule();

そして、テスト対象のコントローラとMockとして差し込みたいクラスをInjectMocksとMockアノテーションをつけて定義します。

@InjectMocks

private HomeController controller;

@Mock

private Logic logic;

そして、MockMvcをフィールドに定義してBeforeで初期化します。

private MockMvc mockMvc;

@Before

public void before() {

this.mockMvc = MockMvcBuilders.standaloneSetup(this.controller).build();

}

あとはMockitoでモックの挙動を指定してmockMvcを使ってテストを書くだけです。

@Test

public void test() throws Exception {

Mockito.when(logic.greet()).thenReturn("こんにちは世界");

this.mockMvc.perform(MockMvcRequestBuilders.post("/").param("name", ""))

.andDo(MockMvcResultHandlers.print())

.andExpect(MockMvcResultMatchers.status().isOk())

.andExpect(MockMvcResultMatchers.model().hasErrors())

.andExpect(MockMvcResultMatchers.view().name("index"))

.andExpect(MockMvcResultMatchers.model().attribute("message", "こんにちは世界"));

}

むずいわ！

コードの全体はこんな感じです。

package okazuki.validationEdu;

import org.junit.Before;

import org.junit.Rule;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.Mockito;

import org.mockito.junit.MockitoJUnit;

import org.mockito.junit.MockitoRule;

import org.springframework.test.context.ContextConfiguration;

import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import org.springframework.test.context.web.WebAppConfiguration;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;

import org.springframework.test.web.servlet.result.MockMvcResultHandlers;

import org.springframework.test.web.servlet.result.MockMvcResultMatchers;

import org.springframework.test.web.servlet.setup.MockMvcBuilders;

@RunWith(SpringJUnit4ClassRunner.class)

@WebAppConfiguration

@ContextConfiguration(classes = App.class)

public class HomeControllerTest {

@Rule

public final MockitoRule rule = MockitoJUnit.rule();

private MockMvc mockMvc;

@InjectMocks

private HomeController controller;

@Mock

private Logic logic;

@Before

public void before() {

this.mockMvc = MockMvcBuilders.standaloneSetup(this.controller).build();

}

@Test

public void test() throws Exception {

Mockito.when(logic.greet()).thenReturn("こんにちは世界");

this.mockMvc.perform(MockMvcRequestBuilders.post("/").param("name", ""))

.andDo(MockMvcResultHandlers.print())

.andExpect(MockMvcResultMatchers.status().isOk())

.andExpect(MockMvcResultMatchers.model().hasErrors())

.andExpect(MockMvcResultMatchers.view().name("index"))

.andExpect(MockMvcResultMatchers.model().attribute("message", "こんにちは世界"));

}

}

テスト対象のコントローラはこんな感じ。

package okazuki.validationEdu;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

@Controller

public class HomeController {

@Autowired

Logic logic;

@RequestMapping

public String index(@ModelAttribute("form") Person p) {

return "index";

}

@RequestMapping(method = RequestMethod.POST)

public String indexPost(@ModelAttribute("form") @Validated(All.class) Person p, BindingResult r, Model model) {

model.addAttribute("message", this.logic.greet());

return "index";

参照：

<https://docs.spring.io/spring-boot/docs/current/reference/html/spring-boot-features.html#boot-features-testing>