LAB- Spring Mvc -Testing

In this lab you will be understanding how Test Controllers

STEP -1

In this step, you will be working on **10mvc-tests-start** project

We want to write test cases for CourseController methods.

Our CourseController has dependency on TestimonialService and CourseService.

Firstly, all the dependencies have to be mocked.

So, Write java Configuration file for Testing as shown below :

```
@Configuration
public class TestConfig {

	@Bean
	public ITestimonialService testimonialService(){
		return Mockito.mock(ITestimonialService.class);
	}
	@Bean
	public ICourseService courseService(){
		return Mockito.mock(ICourseService.class);
	}

	@Bean
	public IAboutUsService aboutUsService(){
		return Mockito.mock(IAboutUsService.class);
	}
}
```

sdfsaf

Now Open CourseControllerTests.java under com.way2learnonline.tests and observe the code .

Observe that TestimonialService, CourseService and WebApplicationContext have been autowired for you already.

Also, in @Before annotated method, same course and testimonial object are initialized.

We have to complete the two @Test methods now.

Firstly, create a variable of MockMVc in this class as below :

```
private MockMvc mockMvc;
```

Initialize mockMvc as shown below inside setUp() method :

```
mockMvc=MockMvcBuilders.webAppContextSetup(webApplicationContext).build();
```

Now, reset the services as shown below :

```
Mockito.reset(testimonialService,courseService);
```

Initialize the mocks with expectations as shown below :

```
when(courseService.get("hadoop")).thenReturn(course);

when(testimonialService.getTestimonialsByCourse("hadoop")).thenReturn(Arrays.asList(tes
timonial1,testimonial2));
```

Inside testIndividualCourse() method , create a request builder as shown below :

```
MockHttpServletRequestBuilder
requestBuilder=MockMvcRequestBuilders.get("/viewIndividualCourse.htm").param("courseId"
, "hadoop");
```

Now, perform the request as shown below :

```
ResultActions resultActions=mockMvc.perform(requestBuilder);
```

Now, you can assert that the status is 200 as shown below :

```
resultActions.andExpect(MockMvcResultMatchers.status().isOk());
```

you can assert that the view name returned by your controller is "viewIndividualCourse" as shown below :

```
resultActions.andExpect(MockMvcResultMatchers.view().name("viewIndividualCourse"));
```

You can assert that there will be a model attribute called as course as shown below :

```
resultActions.andExpect(MockMvcResultMatchers.model().attribute("course", course));
```

You can assert the size of a model attribute as shown below :

```
resultActions.andExpect(model().attribute("testimonials", hasSize(2)));
```

You can assert for properties of any object in a collection as shown below :

```
resultActions.andExpect(
                                    model().attribute(
                                                "testimonials", hasItem(

      allOf(

            hasProperty("rating",is(5)),

            hasProperty("title",is("Hadoop Trainer was Excellent")),

            hasProperty("description",is("Very Good Training"))

      )
                                                                                  )
                                                            )
```

Finally, verify that courseService is invoked only once as shown below :

```
verify(courseService,times(1)).get("hadoop");
verifyNoMoreInteractions(courseService);
```

2) Can you write the test case for `testViewIndividualCourseWithInvalidCourseId()` ?

Congratulations !! You know how to Test your Controllers