Spring MVC JUnit And Mockito Configuration

```
1   @RunWith(SpringJUnit4ClassRunner.class)
2   @ContextConfiguration(locations = {"classpath:/applicationContext.xml",
3                         "classpath:/mvc-dispatcher-servlet.xml",
4                         "classpath:/spring-security.xml"})
5   @WebAppConfiguration
6   public class SimpleTest {
7        @Autowired
8        WebApplicationContext wac;
9
10       @Autowired
11       private UserService userService;
12
13       private MockMvc mockMvc;
14
15       @Before
16       public void setup(){
17           MockitoAnnotations.initMocks(this);
18           this.mockMvc= MockMvcBuilders.webAppContextSetup(wac).build();
19       }
20
21       @Test
22       public void testtWelcomePage() throws Exception {
23           ModelAndView modelAndView = mockMvc.perform(get("/admin"))
24           .andExpect(status().isOk())
25           .andReturn()
26           .getModelAndView();
27           assertEquals("admin/index", modelAndView.getViewName());
28       }
29
30       @Test
31       public void getAllUsers() {
32           User user = userService.getUser("codesenior");
33           assertNotNull(user);
34       }
35  }
```

At line 1, We should use SpringJUnit4ClassRunner class, not MockitoJUnitRunner class, but if you want to use both of them, you should define `@RunWith(SpringJUnit4ClassRunner.class)` and add `MockitoAnnotations.initMocks(this)` in setup method annotated with @Before. Because we can't use multiple @RunWith annotation, we choosed to replace `@RunWith(MockitoJUnitRunner.class)` with `MockitoAnnotations.initMocks(this)`

At line 2 and 3, we configured Spring application. Notice that there are three xml configuration file are used. If you are developing wep application, these files most probably located in the WEB-INF directory. Add below configuration into the maven pom.xml file, then Maven will automatically add **WEB-INF** directory into the classpath:

```
<build>
    <testResources>
        <testResource>
            <directory>src/main/webapp/WEB-INF</directory>
        </testResource>
    </testResources>
</build>
```

Mockito library provides us **MockMvc** class to emulate **GET, POST**, etc. HTTP requests, so we have to initialize this class in **setup()** method as above.

At line 31, we called UserService's **getUser()** method.

Now, lets look at Maven dependencies:

```
<properties>
    <spring.test>4.1.4.RELEASE</spring.test>
    <junit.version>4.12</junit.version>
    <hamcrest.version>1.3</hamcrest.version>
    <mockito.all>2.0.2-beta</mockito.all>
```

```
</properties>

<dependencies>
<dependency>
     <groupId>org.springframework</groupId>
     <artifactId>spring-test</artifactId>
     <version>${spring.test}</version>
     <scope>test</scope>
</dependency>
<dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>${junit.version}</version>
     <scope>test</scope>
</dependency>
<dependency>
     <groupId>org.hamcrest</groupId>
     <artifactId>hamcrest-library</artifactId>
     <version>${hamcrest.version}</version>
     <scope>test</scope>
</dependency>
<dependency>
     <groupId>org.mockito</groupId>
     <artifactId>mockito-all</artifactId>
     <version>${mockito.all}</version>
     <scope>test</scope>
</dependency>
</dependencies>
```

If you want to use Mockito standalone configuration, you can look at this article: http://www.codesenior.com/en/tutorial/Spring-MVC-JUnit-Mock-Standalone-Configuration