

Forum:[Hack Forums](#)

Thread: [Spring Boot Unit Testing](#)

SIM-1947090 - November 26, 2017, 2:13 pm

Hey,

I am pretty much new to this JUnit testing thing :/ I tried a whole lot different methods, and none seemed to work.

Specially reading this docs:

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

I either suck at reading docs, or this one is not really complete or, I am just too stupid. So help me out :/

Im trying to test my services for now, and then try some other cool features in this area. But my tests just do not pass :O

This is the service Im trying to test: <https://pastebin.com/C4VWMQFj>

This is the service test: <https://pastebin.com/H0LFq4Pe>

This is Spring Boot application:

Code @SpringBootApplication

```
public class Spotted {
```

```
    public static void main(String[] args) {
        SpringApplication.run(Spotted.class, args);
    }
}
```

This is config:

Code @Configuration

```
public class SpottedConfig extends WebMvcConfigurerAdapter {
```

```
    @Autowired
    private AuthInterceptor authInterceptor;
```

```
    @Override
    public void addInterceptors(InterceptorRegistry registry){
```

```
        registry.addInterceptor(authInterceptor).addPathPatterns("/**").excludePathPatterns("/user/r/**",
"/user/l/**", "/user/cr/**", "/t/**");
    }
}
```

As you can see from the pastebin, userRepository.save(user) returns null (assertion fails obviously) for some reason.

What am I doing wrong?

Dialog - November 26, 2017, 2:50 pm

How are you handling dependency injection? You should have a separate layer for your testing environment here, and use mockito for mocking objects.

SIM-1947090 - November 26, 2017, 3:02 pm

(Nov 26, 2017 03:50 PM)Dialog Wrote: [CITE](#) How are you handling dependency injection? You should have a separate layer for your testing environment here, and use mockito for mocking objects.

Hmm, isn't this supposed to do the job, all the required connections, mocking and stuff:

- inside of UserService (userRepository is autowired)

Code @Autowired

```
private UserRepository userRepository;
and
- inside of test UserService (both are MockBean-ed) - shouldn't this make a valid connection
between two mocked objects?
```

Code @MockBean

```
private UserService userService;
```

@MockBean

```
private UserRepository userRepository;
```

Just one more thing to note:

I added this one and it's passed: `verify(userService, times(1)).register(user);`

So, how can that line pass but still return null from `userService.register(user)`

UPDATE:

Maybe I got it.

I had to manually create Mocks and Inject those. MockBean annotation doesn't seem to do that job :O (I thought it would)

Looks like this gets the job done (but I'm not sure why @MockBean didn't work well):

Code @InjectMocks

```
private UserService userService;
```

@Mock

```
private UserRepository userRepository;
```

@Mock

```
private SessionRepository sessionRepository;
```

@Mock

```
private MailService mailService;
```

@Mock

```
private TemplateEngine templateEngine;
```

@Before

```
public void setup() {
    MockitoAnnotations.initMocks(this);
}
```

ZeratulTheWizard - November 26, 2017, 4:44 pm

(Nov 26, 2017 04:02 PM)H1gh-T3ch Wrote: [\[CITE\]](#) Hmm, isn't this supposed to do the job, all the required connections, mocking and stuff:

- inside of UserService (userRepository is autowired)

Code @Autowired

```
private UserRepository userRepository;
and
- inside of test UserService (both are MockBean-ed) - shouldn't this make a valid connection
between two mocked objects?
```

Code @MockBean

```
private UserService userService;
```

@MockBean

```
private UserRepository userRepository;
```

Just one more thing to note:

I added this one and it's passed: `verify(userService, times(1)).register(user);`

So, how can that line pass but still return null from `userService.register(user)`

UPDATE:

Maybe I got it.

I had to manually create Mocks and Inject those. MockBean annotation doesn't seem to do that job :O (I thought it would)

Looks like this gets the job done (but I'm not sure why @MockBean didn't work well):

Code @InjectMocks

```
private UserService userService;
```

@Mock

```
private UserRepository userRepository;
```

@Mock

```
private SessionRepository sessionRepository;
```

@Mock

```
private MailService mailService;
```

@Mock

```
private TemplateEngine templateEngine;
```

@Before

```
public void setup() {
    MockitoAnnotations.initMocks(this);
}
```

If you're using JUnit Runner or Mockito Runner as the RunWith, that might be an issue. You might need to use Spring Runner if you're using Spring annotations.

SIM-1947090 - November 27, 2017, 12:13 am

(Nov 26, 2017 05:44 PM)ZeratulTheWizard Wrote: [\[CITE\]](#) If you're using JUnit Runner or Mockito Runner as the RunWith, that might be an issue. You might need to use Spring Runner if you're using Spring annotations.

Yes, I am using @RunWith(SpringRunner.class).

And, with the latest updates, I can confirm that `@MockBean` actually does the same job, I don't need to manually setup mocks with `@InjectMocks`, `@Mock` and `MockitoAnnotations.initMocks(this)`. But it appears that third-party library is causing some issues, which is Thymeleaf template engine. It for some reason fails to configure itself in tests, hence my tests fail big time (I probably need to manually configure it for tests or something).

So I decided to not waste my time and use `@InjectMocks`, `@Mock` and `MockitoAnnotations.initMocks(this)` instead and continue along.