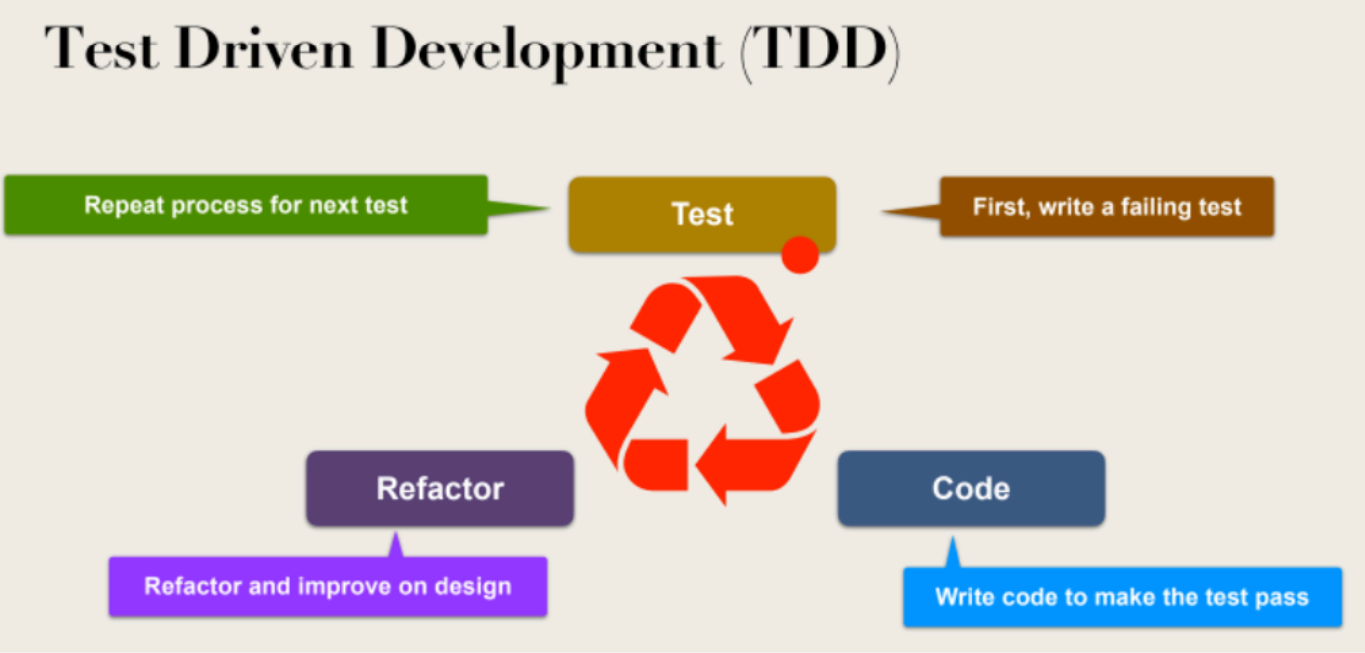
Testing

Test （super important）

Questions

1. What is the TDD?
2. What tools do you use for testing? Junit, Mockito, PowerMockito
3. What is unit-test and what is Integration Test?
4. What is automation?



OverView

General Test in industry

Environment	Test Types	Tools
Local	Peer Test Unit Test	Human Junit, Mockito, PowerMock etc
Dev	Unit Test Automation Test	Junit etc Selenium, TestNG, Cucumber etc
QA	Manual Automation Regression	Human, HP Quality Center Selenium, TestNG, Cucumber etc Automation test
Pre-Prod	Performance/Load Test	JMeter, Gatling
Prod	AB Test Alpha/Beta test (Prod-Log Retrieve)	User

看这张图时候，顺便思考开发流程和不同的环境以及数据库。

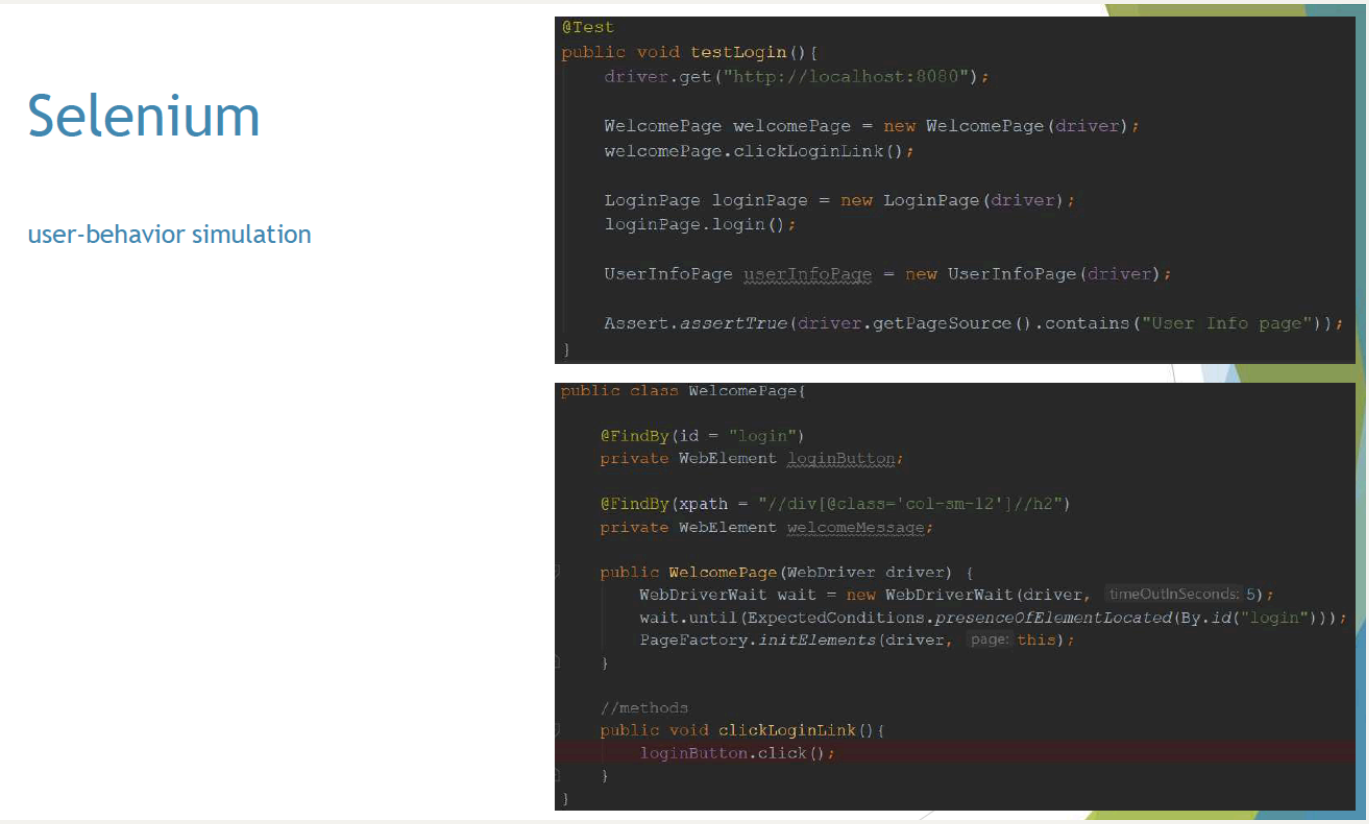
Test Suite - Automation

```
@RunWith(Suite.class)
@Suite.SuiteClasses({
        MainControllerMockTest.class,
        UserDetailsServiceImplTest.class,
        UserDetailsServiceImplMockTest.class
})
public class CompositeApplicationTests {

    @Test
    public void contextLoads() {
    }

}
```

Automation whole application from front end

- Selenium
  - WebDriver - a driver to help grab web elements
  - Page Objects - a java class to simulate a web page
- TestNG (Data-Driven Development)
- Cucumber

Selenium: User-behavior simulation



Cucumber: **Given , when and then**

BDD: Behavior-Driven Development

Unit Testing

Unit testing involves the testing of each unit or an individual component of the software application.

The purpose is to validate that each unit of the software code performs as expected.

Unit testing is done during develpment of an application by the developers

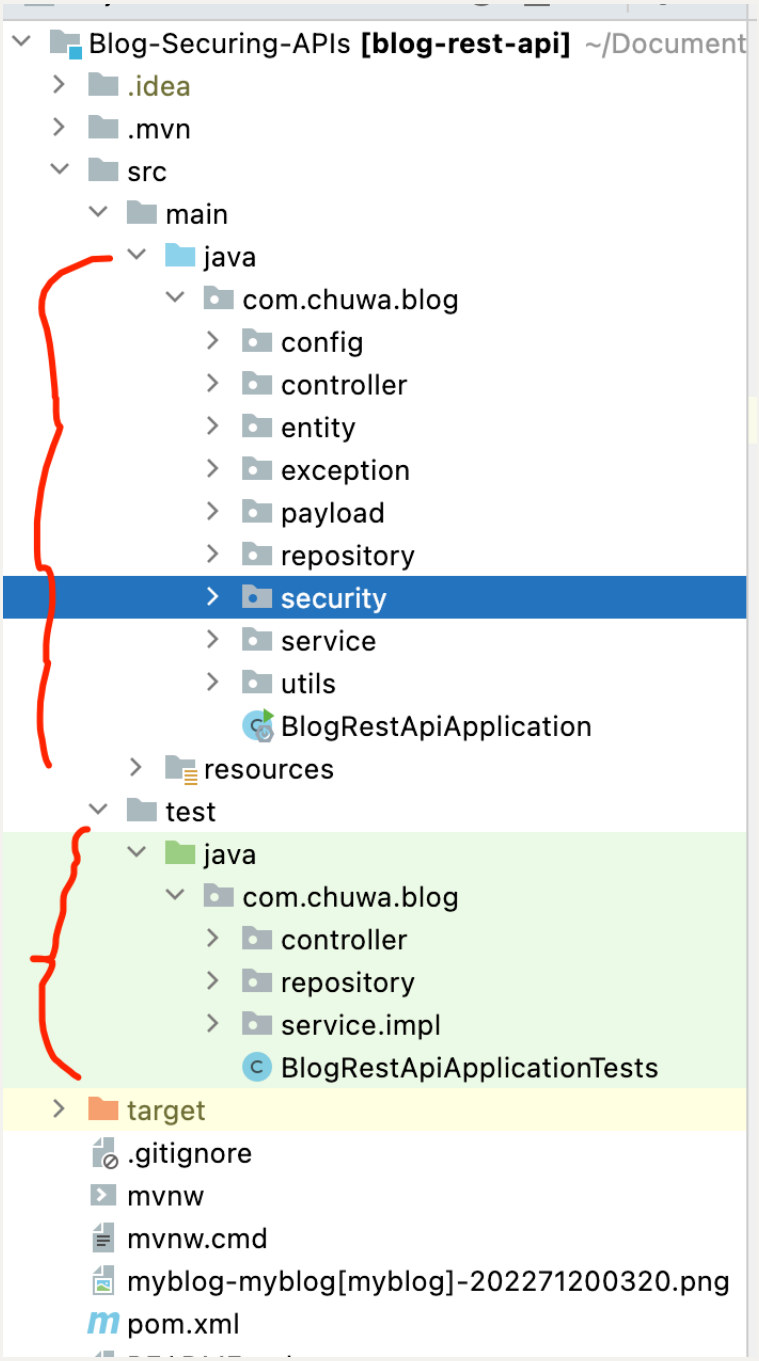
Unit may be a anindividual function, method, procedure, module and object.

Even though we have QA, usually the developer who developed the new code write unit test by themselves.

Source code structure

It's a good idea to keep the test classes **spearate** from the main source code. So, they are developed, executed, and maintained separately from the production code.

Also, it **avoids** any possibility of running test code in the production environment.



Command + Shift + T: create or find the corresponded test

Naming Conventions

Package Naming Convention

> *We should create a **similar package structure** in the src/test directory for test classes. Thus, improving the readability and maintainability of the test code. (如上图)*

好的命名能让人根据名字知道该test的意图。

进项目后，看下项目上现有的命名风格是什么。保持统一即可。

https://methodpoet.com/unit-test-method-naming-convention/

https://qualitycoding.org/unit-test-naming/
My name Convention:

- Class name: xxxxTest.java
    - UserServiceTest.java
- Method name: testMethodName.java
    - testFindUser.java
    - testFindUser_ThrowUserNotFoundException.java

Tools

1. Use Junit to assert if the result equals to expected.

```
    // 3. assert
    assertEquals(expected, actual);
    Assertions.assertNotEquals(7, actual);
```

2. **Mock** External Services using **Mockito**, **EasyMock**, or **JMockit**.

```
BDDMockito.given(employeeRepository.findByEmail(empl
oyee.getEmail()))
                .willReturn(Optional.empty());
```

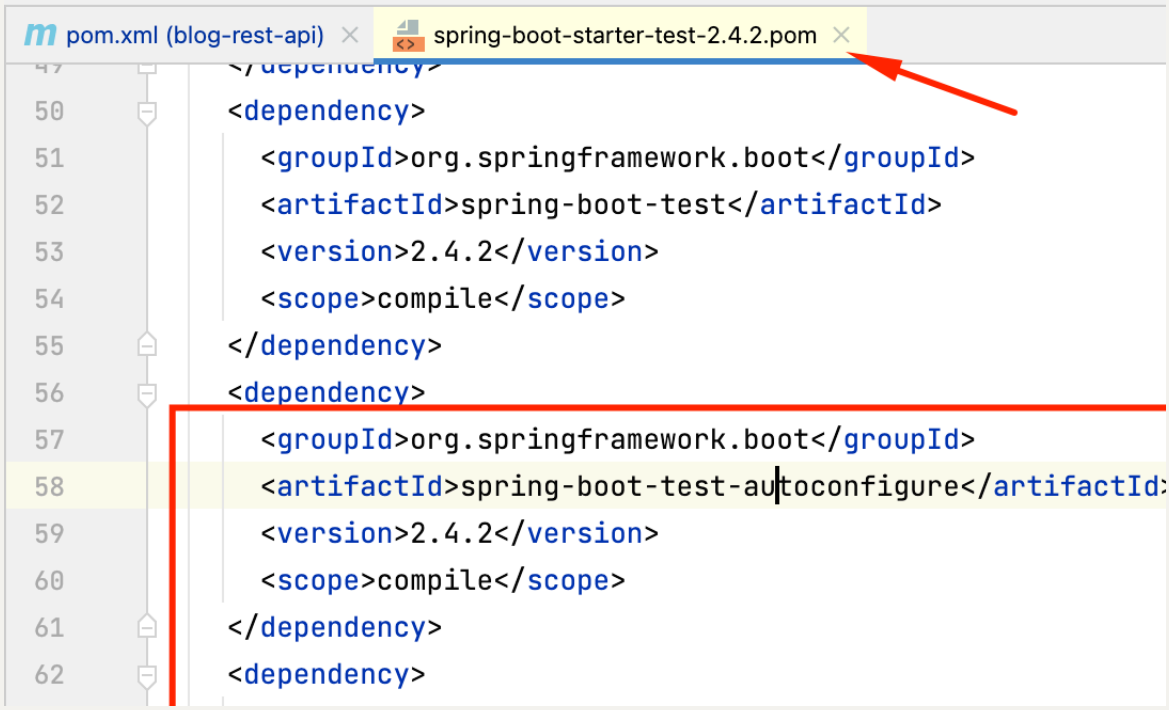
Steps of creating Unit Test

Dependency

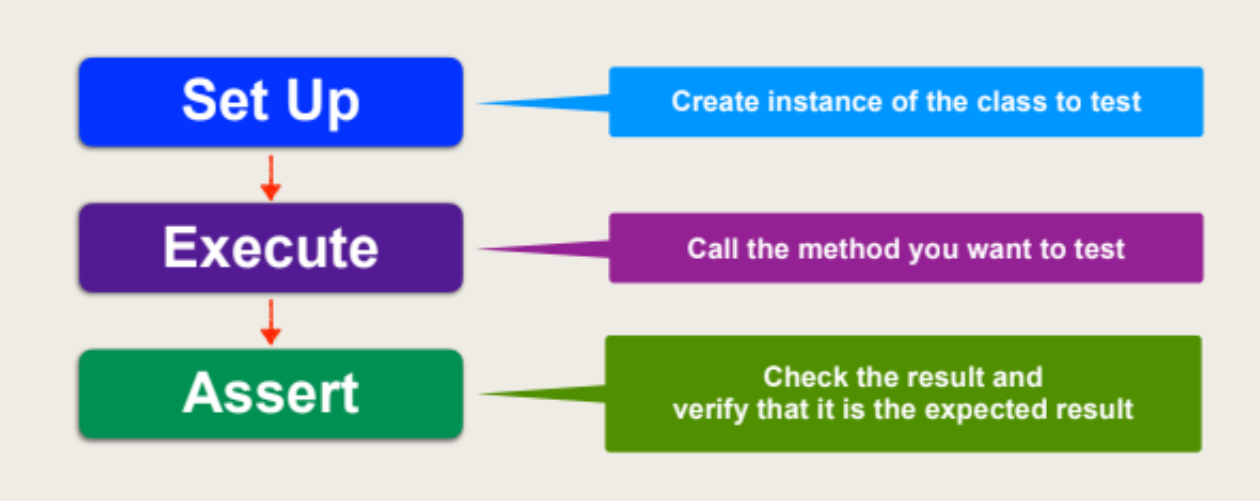
```
<dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
</dependency>
```

This starter includes:

1. Spring-specific dependencies
2. Dependencies for auto-configuration
3. Set of testing libraries - JUnit, Mockito, Hamcrest, AssertJ, JSONNassert and JsonPath



```
    <groupId>com.jayway.jsonpath</groupId>
    <artifactId>json-path</artifactId>
    <version>2.4.0</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>jakarta.xml.bind</groupId>
    <artifactId>jakarta.xml.bind-api</artifactId>
    <version>2.3.3</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.assertj</groupId>
    <artifactId>assertj-core</artifactId>
    <version>3.18.1</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.hamcrest</groupId>
    <artifactId>hamcrest</artifactId>
    <version>2.2</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.7.0</version>
    <scope>compile</scope>
  </dependency>
```




```java
public class DemoUtilsTest extends TestCase {

    @Test
    public void testEqualsAndNotEquals() {
        // 1. set up
        DemoUtils demoUtils = new DemoUtils();
        int expected = 6;

        // 2. execute
        int actual = demoUtils.add(2, 4);

        // 3. assert
        assertEquals(expected, actual);
    }
}
```

Junit

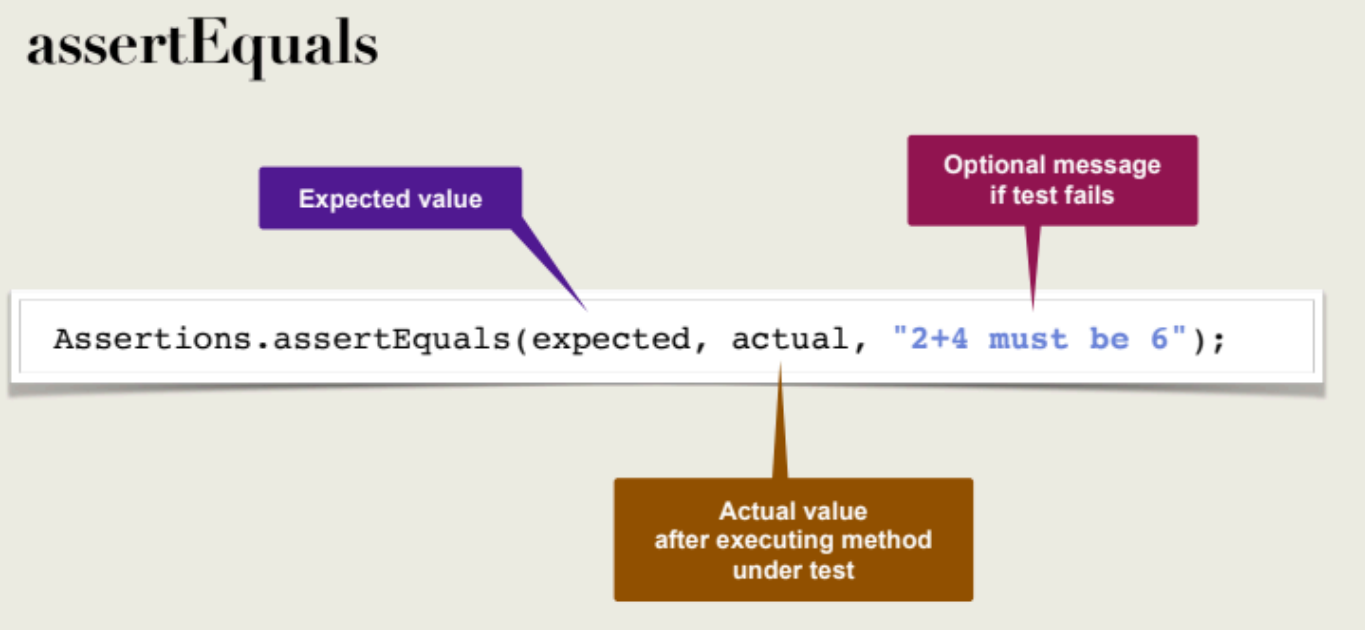
https://junit.org/junit5/docs/current/user-guide/#writing-tests

Assertions

JUnit has a collection of assertions

Defined in class: `org.junit.jupiter.api.Assertions`

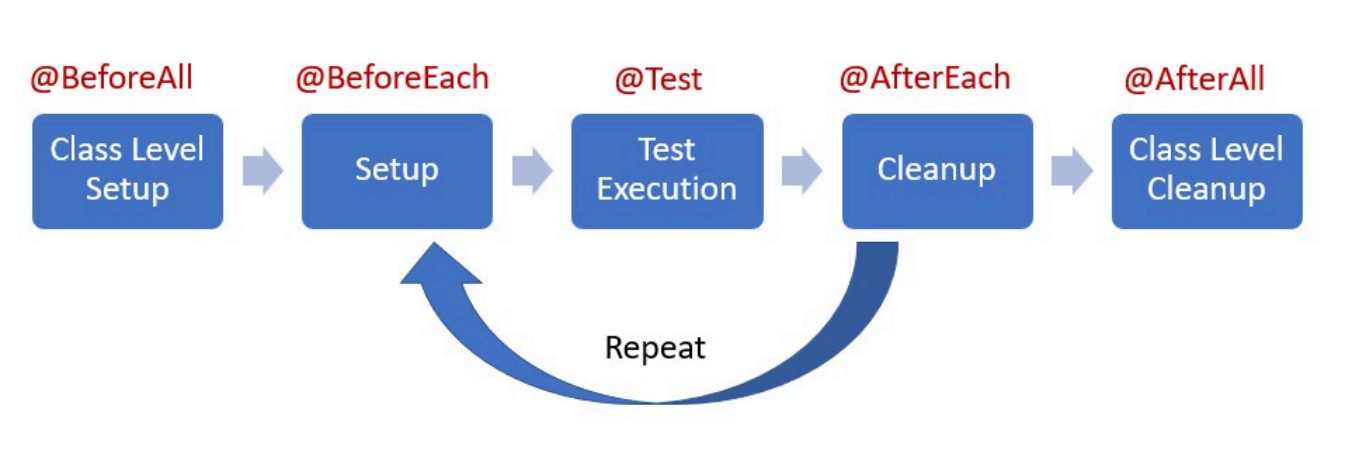
Method name	Description
`assertEquals`	Assert that the values are equal
`assertNotEquals`	Assert that the values are not equal
`assertNull`	Assert that the value is null
`assertNotNull`	Assert that the value is not null
...	...



LifeCircle

https://howtodoinjava.com/junit5/junit-5-test-lifecycle/

https://dzone.com/refcardz/junit-and-easymock#:~:text=methods%20and%20attributes.-,JUnit%20Lifecycle,teardown%2C%20all%20executed%20in%20sequence.



@BeforeAll 修饰的method是static method，因为他是class level的。
Questions

- Why do we need @BeforeAll and @BeforeEach?
- If we don't use them, what else ways we can do?

ANNOTATION	DESCRIPTION
@BeforeAll	Method is executed only once, before all test methods. *Useful for getting database connections, connecting to servers ...*
@AfterAll	Method is executed only once, before all test methods. *Useful for getting database connections, connecting to servers ...*

DAO Test

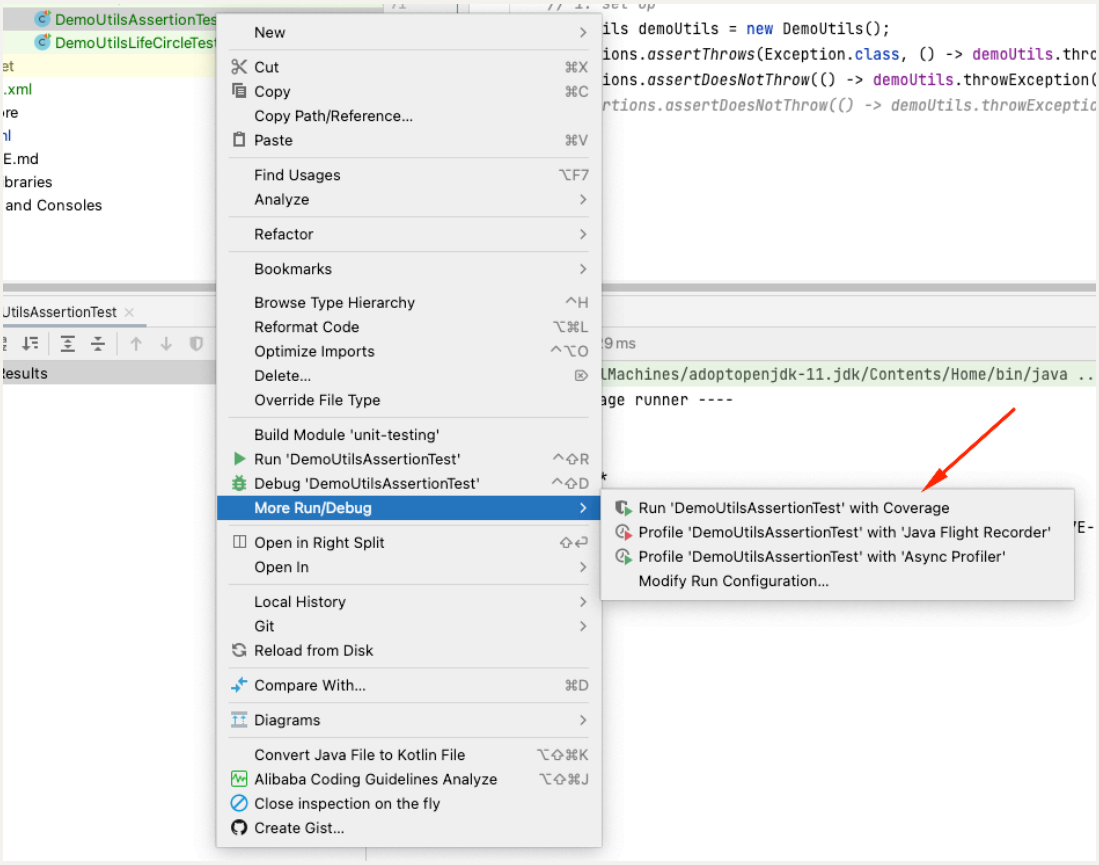
- `@DataJpaTest`
    - Spring Boot provides the @DataJpaTest annotation to test the persistence layer components that will auto configure in-memeory embedded database for testing purposes

```
1 import org.junit.jupiter.api.Test;
```

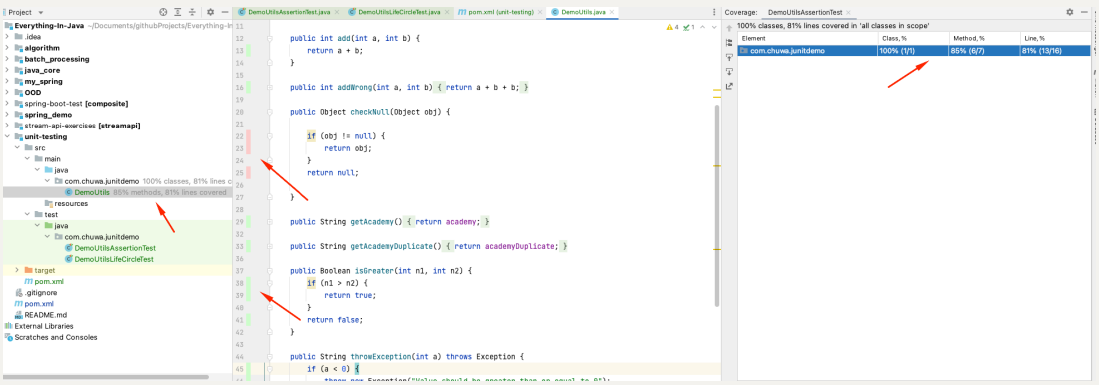
Code Coverage

IDE-Intellij

1. how to run: Run xxxTest.java with Coverage
    a.



2. Report
    a.



Maven

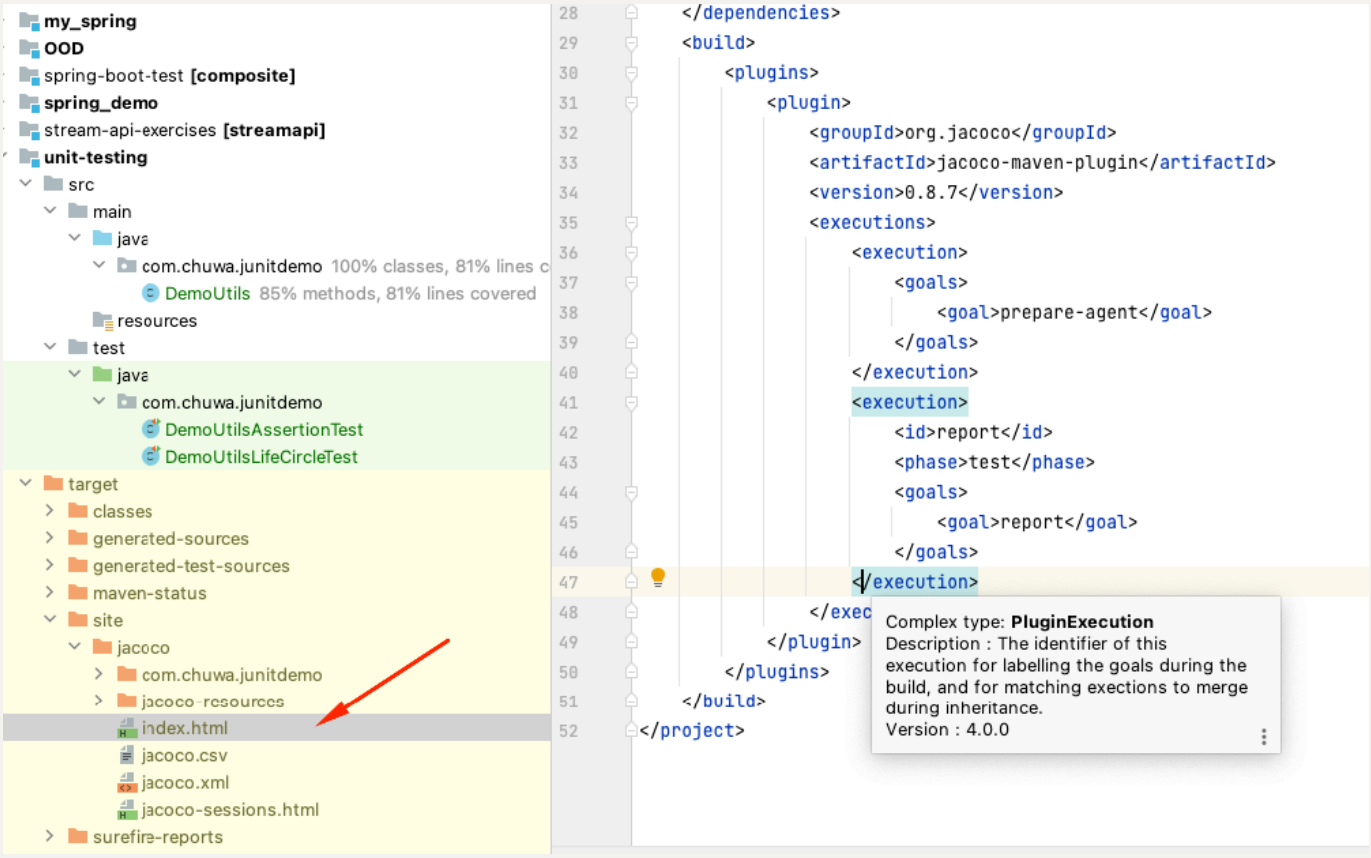
plugin in the pom.xml

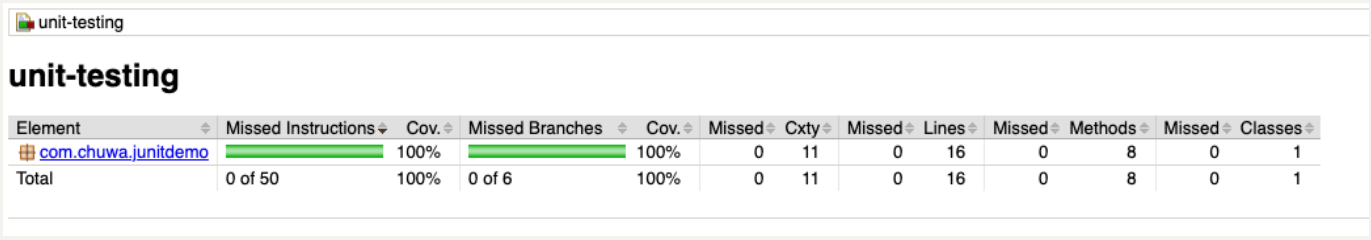
report will be produced in test phase, because we set `<phase>test</phase>`

Command line: **mvn clean test**

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.jacoco</groupId>
                <artifactId>jacoco-maven-plugin</artifactId>
                <version>0.8.7</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>prepare-agent</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>report</id>
                        <phase>test</phase>
                        <goals>
                            <goal>report</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

report is in the target/site/index.html





Mockito

Quesitons

1. what is unit test?
2. why do we need to use Mockito?

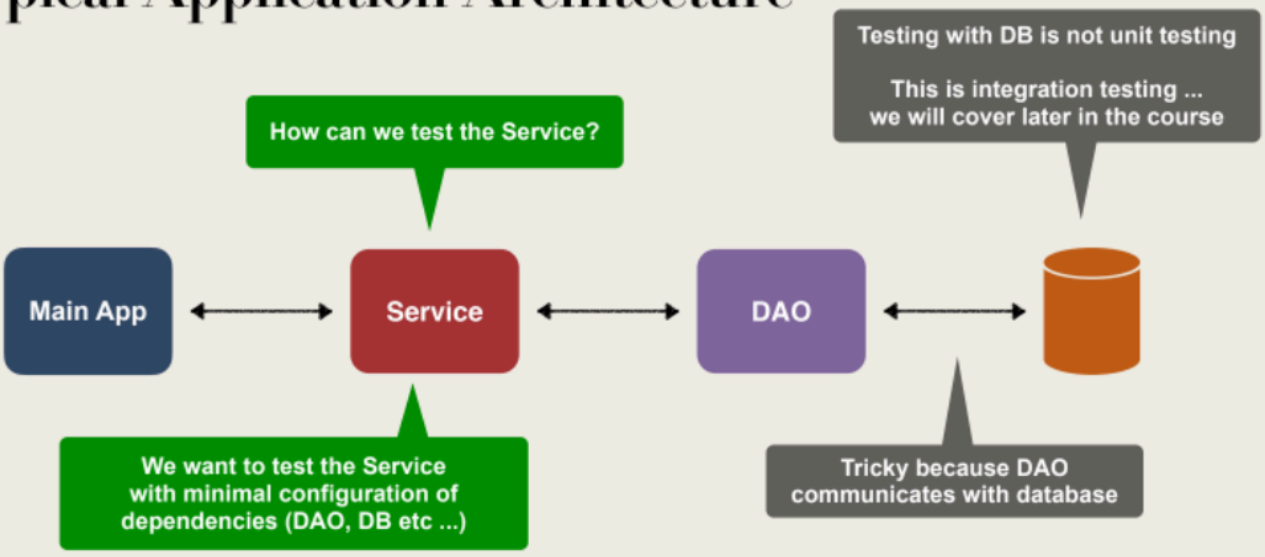
Mocking Frameworks

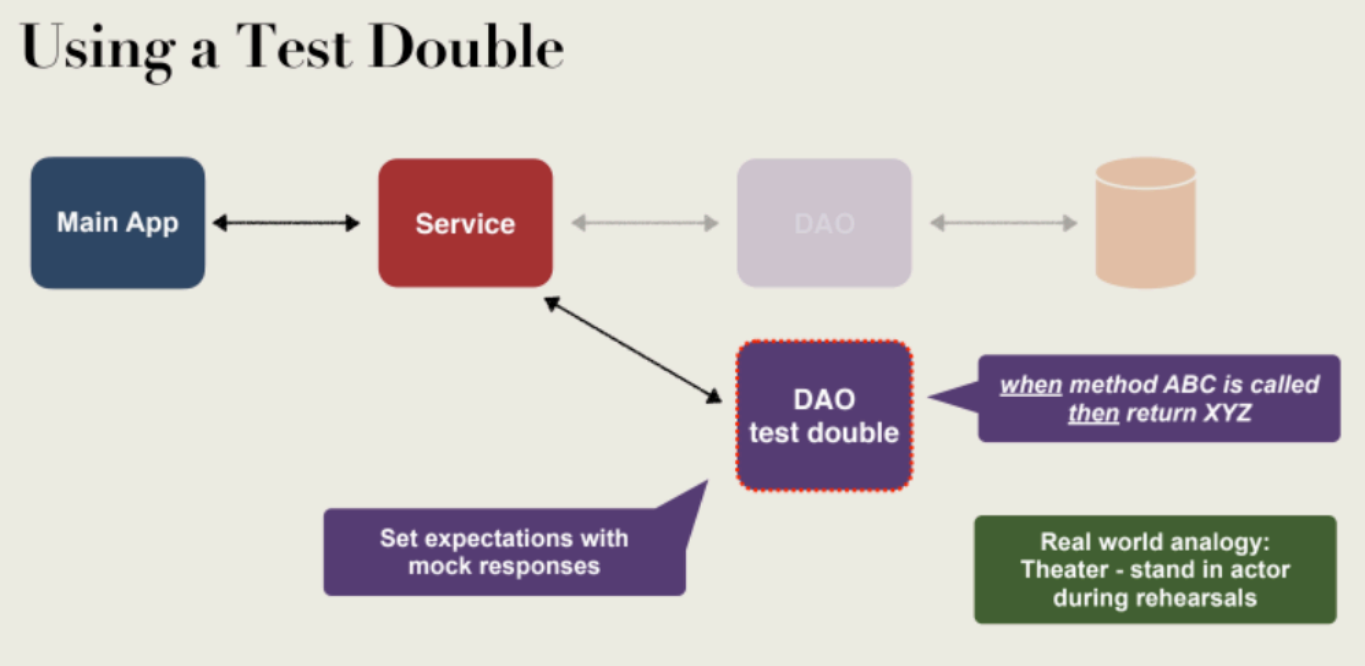
Mocking Frameworks

Name	Website
Mockito	site.mockito.org
EasyMock	www.easymock.org
JMockit	jmockit.github.io
...	

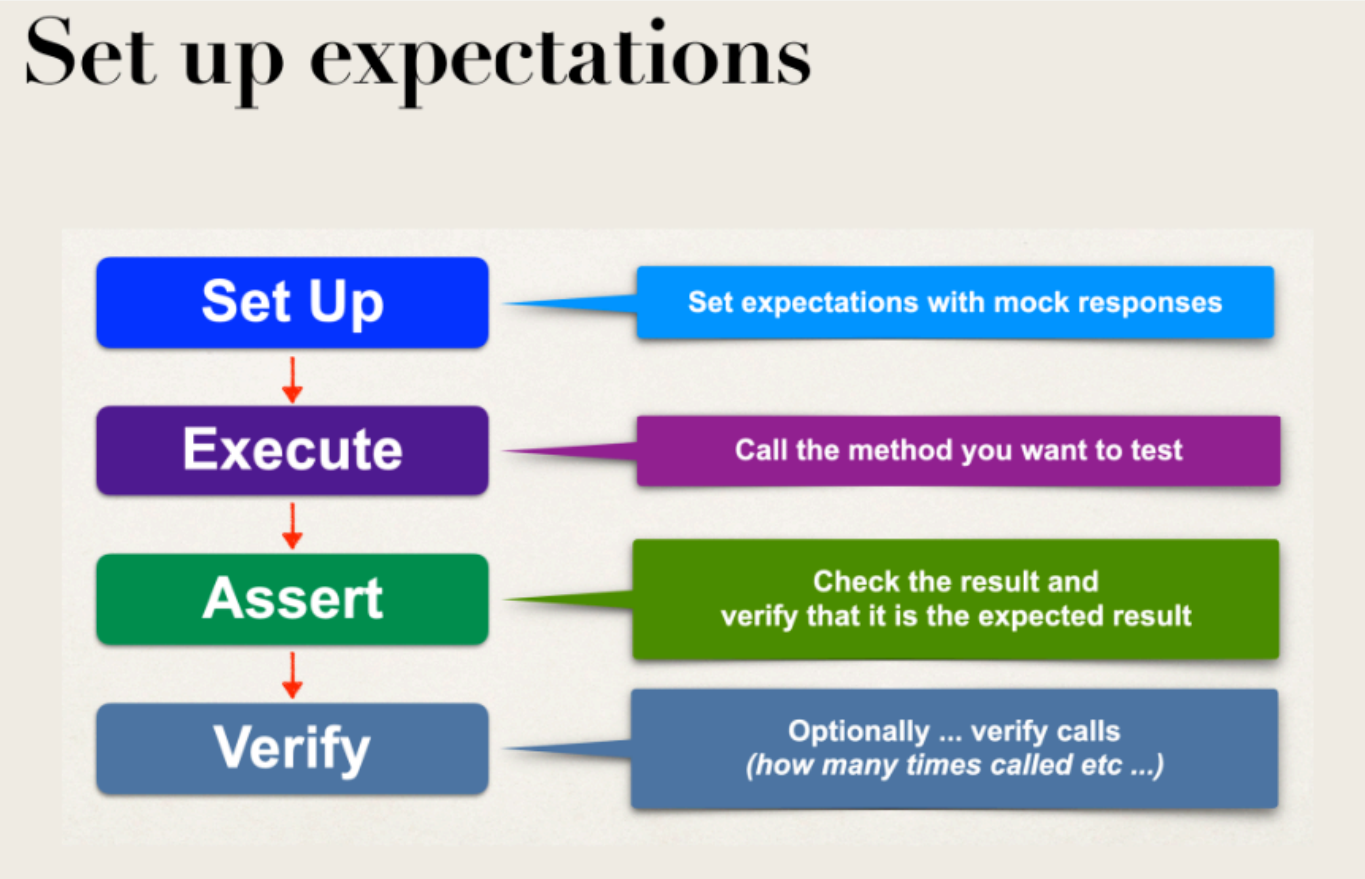
We will use Mockito since
Spring Boot has built-in support for Mockito

Typical Application Architecture





Mockito Steps



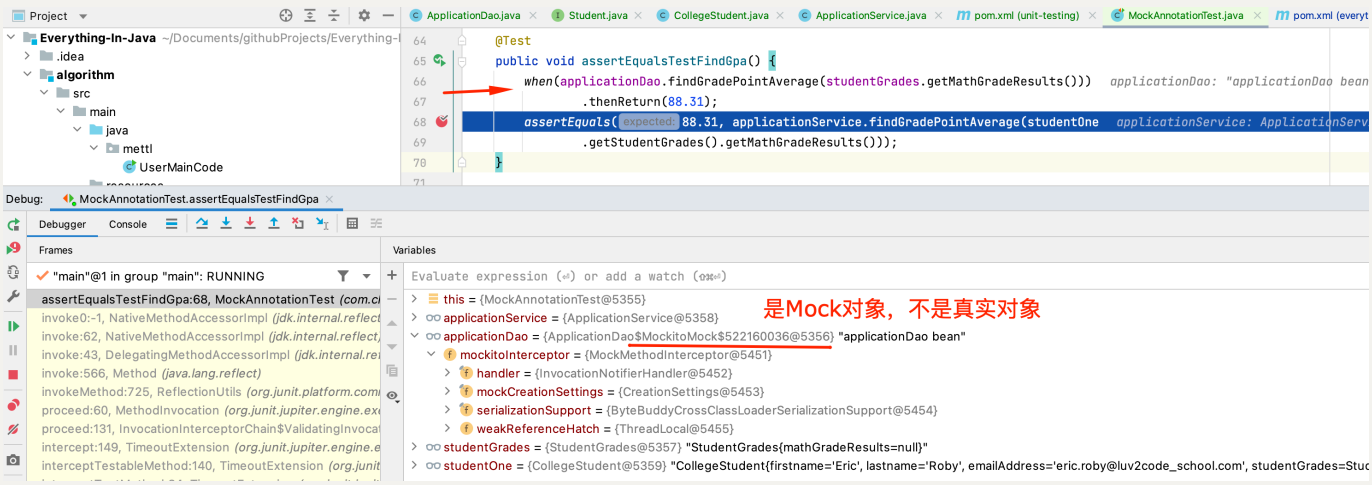
Annotations

- @SpringBootTest
- @Mock
- @Spy
- @InjectMocks
    - 被它修饰的variable service依赖一些其它方法，这些方法被@mock注释，把该类变量注入到service中
    - @InjectMocks FindUserService. this service use findUserBasicInfoDao and findUserExtraInfoDao
    - @Mock findUserBasicInfoDao, @Mock findUserExtraInfoDao
    - 这俩变量会被注入到@InjectMocks FindUserService

Mockito Methods

https://www.jianshu.com/p/7b67806b74f6

Need to debug to show we don't dive in the real DAO method.

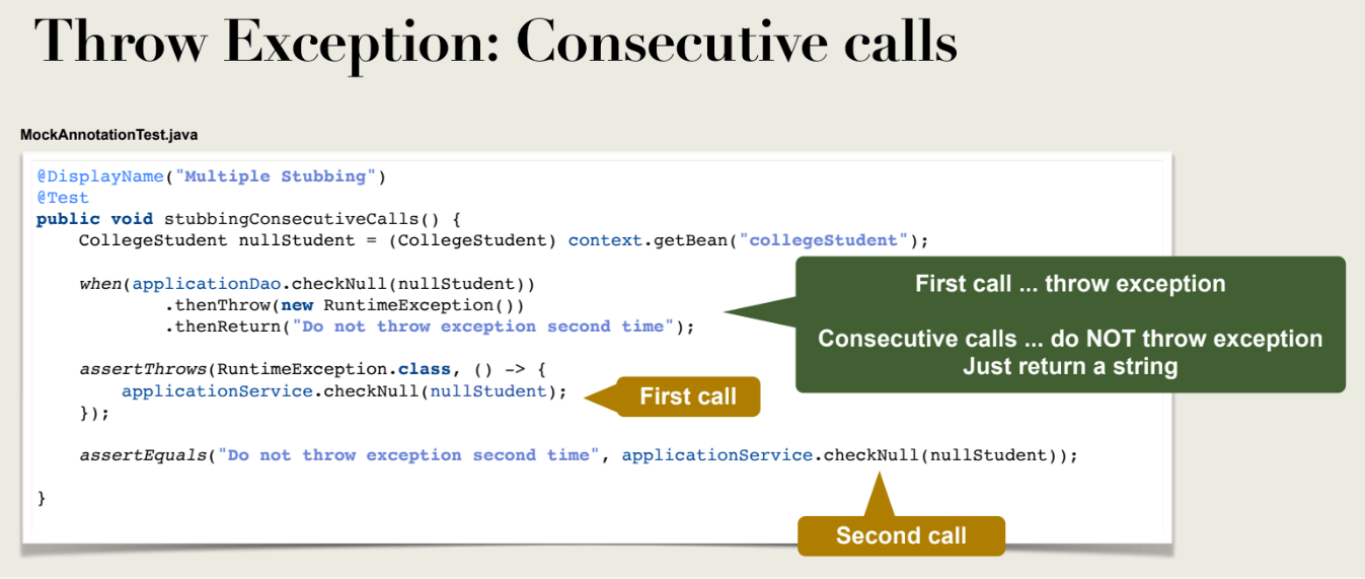


- Mockito.when(mockObject.targetMethod(args)).**thenReturn**(desiredReturnValue);
    - args could be :
    - anyString

    - anyInt
    - anyLong
    - anyObject
    - any(clazz)
    - anyCollection
    - anyCollectionOf(Clazz)
    - anyList(Map, set)
    - anyListOf(clazz)
- **thenThrow()**
    -

```
Mockito.when(postRepositoryMock.findById(ArgumentMatchers.anyLong())).thenThrow(new RuntimeException());
```

- **doNothing()**：指定void方法什么都不做
- doCallRealMethod()：指定方法调用内部的真实逻辑
- Mockito.doAnswer(desiredAnswer).when(mockObject).targetMethod(args);



PowerMockito

PowerMockito is a PowerMock's extension API to support Mockito. It provides capabilities to work with the Java **Reflection** API in a simple way to overcome the problems of Mockito, such as the lack of ability to mock **final, static or private** methods.

Repo: https://github.com/TAIsRich/spring-boot-testing

Branch: https://github.com/TAIsRich/spring-boot-testing/tree/powerMockito

```
1  @RunWith(PowerMockRunner.class)
2  @PrepareForTest({EmployeeUtil.class })
3  class CommonServiceImplTest {
4      @Test
5      void getEmailEmployees() {
6          // 已经创建好 mock 对象。
7          PowerMockito.mockStatic(EmployeeUtil.class);
8          // non-void method, just define the behavior for that
   method
9
    PowerMockito.when(EmployeeUtil.initEmployees(ArgumentMatchers.any
   Int())).thenReturn(null);
10         // void method, we need set doNothign for that class, then
   specify do nothing for which method
11         PowerMockito.doNothing().when(EmployeeUtil.class);
12         EmployeeUtil.deleteEmployees();
13
14         PowerMockito.doNothing().when(EmployeeUtil.class);
15         EmployeeUtil.sth();
16
17         CommonServiceImpl commonService = new CommonServiceImpl();
18         List<Employee> emailEmployees =
   commonService.getEmailEmployees("@gmail.com");
19         Assertions.assertThat(emailEmployees).isNull();
20 //         Whitebox.setInternalState(EmployeeUtil.class,
   "employees", BDDMockito.mock(List.class));
```

```
21
22      }
23  }
```

https://zhuanlan.zhihu.com/p/356231884

https://www.baeldung.com/intro-to-powermock

Integration Test

No Mockito, call real methods.

SonarQube(Code Quality)

https://next.sonarqube.com/sonarqube/projects

Conclusion

Unit Test -> Whitebox Test: Need to use **Mockito** to mock dependecy method behaviors. For the static/final/private methods, we need to use PowerMockito.

```
Method with return:

Mockito.When(aa.method(ArgumentMatchers.anyString()).thenReturn(new AA());

Mockito.when(bb.method()).thenThrow(Mockito.mock(CCExcetion.class));
void method: Mockito.doNothing().when(cc).method();

@RunWith(PowerMockRunner.class)
@PrepareForTest({Math.class })
PowerMockito.mockStatic(Math.class)
PowerMockito.When(Math.abs(ArgumentMatchers.anyInt())).thenReturn(2)
PowerMockito.doNothing().when(EmployeeUtil.class);
EmployeeUtil.deleteEmployees();
```

Integration Test -> Blackbox Test: run the real method. **No Mockito**

Life circle: @BeforeAll, @BeforeEach, @Test, @AfterEach, @AfterAll.

@Autowired and @Resource跟testing无关

Mockito.mock(AA.class);

@Mock, @Spy -> @InjectMocks

@Mockbean -> @MockMvc

```
        // 已经创建好 mock 对象。
        EmployeeServiceImpl mock =
Mockito.mock(EmployeeServiceImpl.class);// new
EmployeeServiceImpl();
        PowerMockito.mockStatic(EmployeeUtil.class);

        // non-void method, just define the behavior for that
method

 Mockito.when(mock.getEmployeeById(ArgumentMatchers.anyLong())).th
enReturn(null);

 PowerMockito.when(EmployeeUtil.initEmployees(ArgumentMatchers.any
Int())).thenReturn(null);
        // void method, we need set doNothign for that class, then
specify do nothing for which method
        Mockito.doNothing().when(mock).getAllEmployees();

        PowerMockito.doNothing().when(EmployeeUtil.class);
        EmployeeUtil.deleteEmployees();
```

Jira

Agile - Sprint

Sprint Planning -> Grooming -> Daily Standup -> Retro/Sprint Review

We use Jira to assign tickets in every sprint.

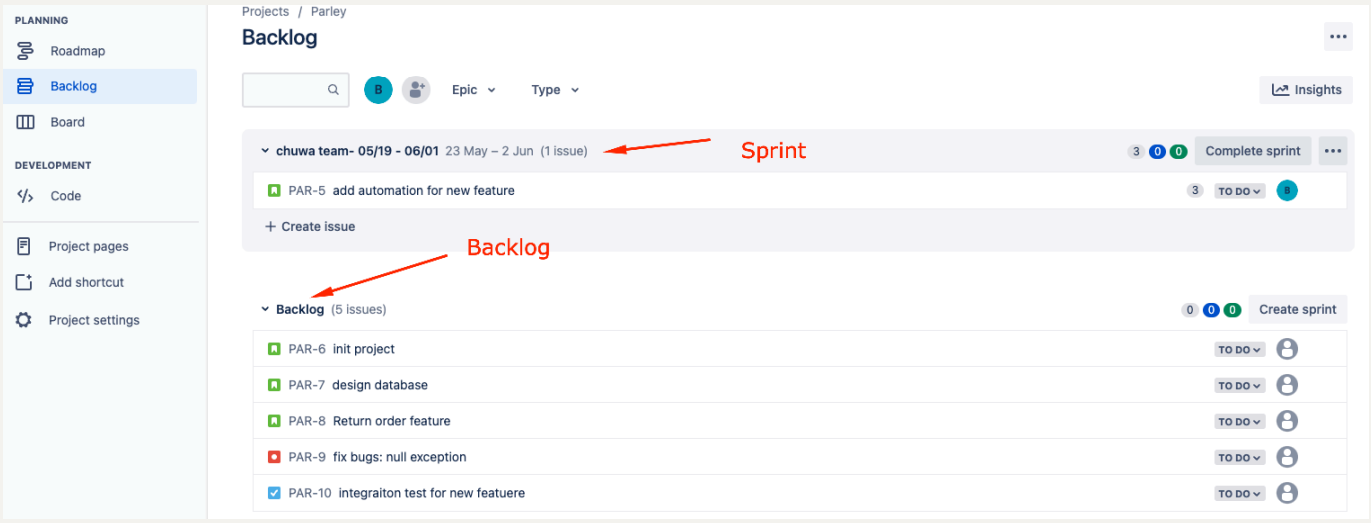
Tickets: new Story/bug fixing, points-based

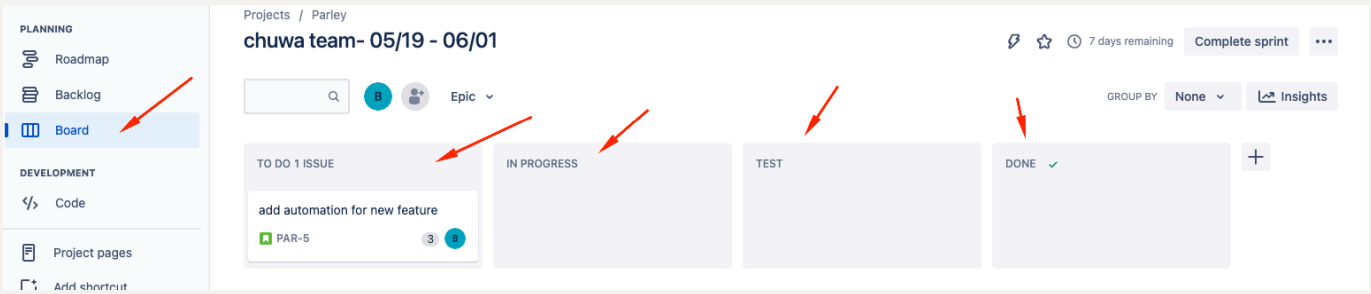
Backlog && Sprint && Board

Backlog is used to store all the stories/tasks

Sprint is usually 2 weeks. Pick up some stoires from backlog to sprint to do.

1 point usually means need 1 day to do.





该task/story/ticket做到哪个阶段，就移动到哪个阶段.
Sprint review 之前需要移动到Done

https://next.sonarqube.com/sonarqube/projects