#### Отличия от Junit

<https://www.codeflow.site/ru/article/junit-vs-testng>

#### Почитать https://csharpcoderr.com/5577/

*TestNG — это среда тестирования,*

**TestNG — это тестовый фреймворк**

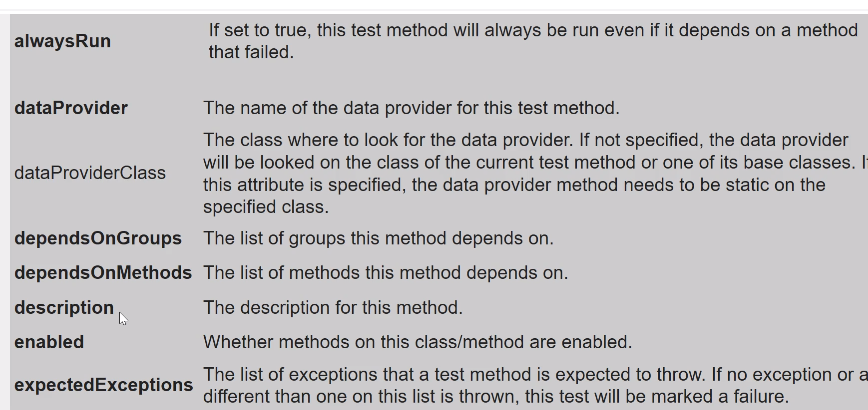
Ты можешь на видеопортале вбить в поиск log4j2 и там выдаст сразу его 7 видосов

#### Почему TestNG?

Потому что dataprovider гибкий. Потому что тест настраивается всего одним xml файлом. Потому что удобно настраивать работу с потоками.

TestNG — это среда тестирования, вдохновленная JUnit и NUnit, но представляющая некоторые новые функциональные возможности, которые делают ее более мощной и простой в использовании.

Групповое тестирование — это новая инновационная функция в TestNG, которой нет в среде JUnit.

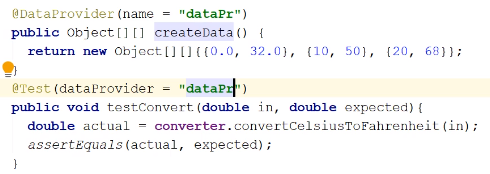
Подключить или библиотеку или зависимость в pom.xpl.  
Аннотации к тестам: (заключаются в скобки после @Test)  


Enabled - включен тест или нет? Если стадия разработки или тест изменился и его надо выключить, то false

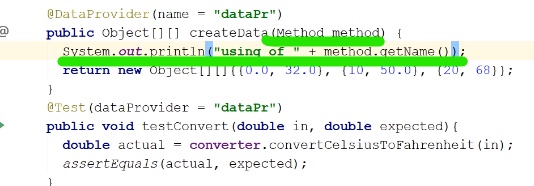
dataProvider - когда один параметризованный тест запускается на нескольких наборах данных

Может возвращать массив массивов класса обжект или итератор этих объектов.

(сразу пишется после@: DataProvider(name = “”))

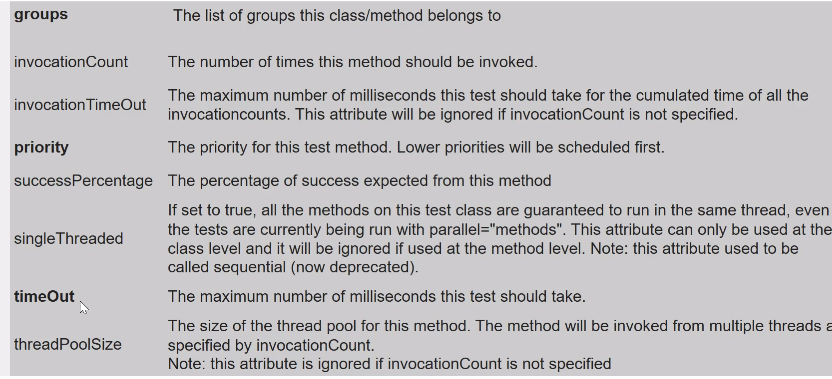


Чтобы узнать, какой метод с использовал провайдер, надо указать метод:



@Test(expectedExceptions = Exception.class) - успешное прохождение теста = генерации исключения

2 depens метода - зависимость выполнения теста от выполнения предыдущего метода или группы. Должен иметь имя, на которое будет ссылаться тест



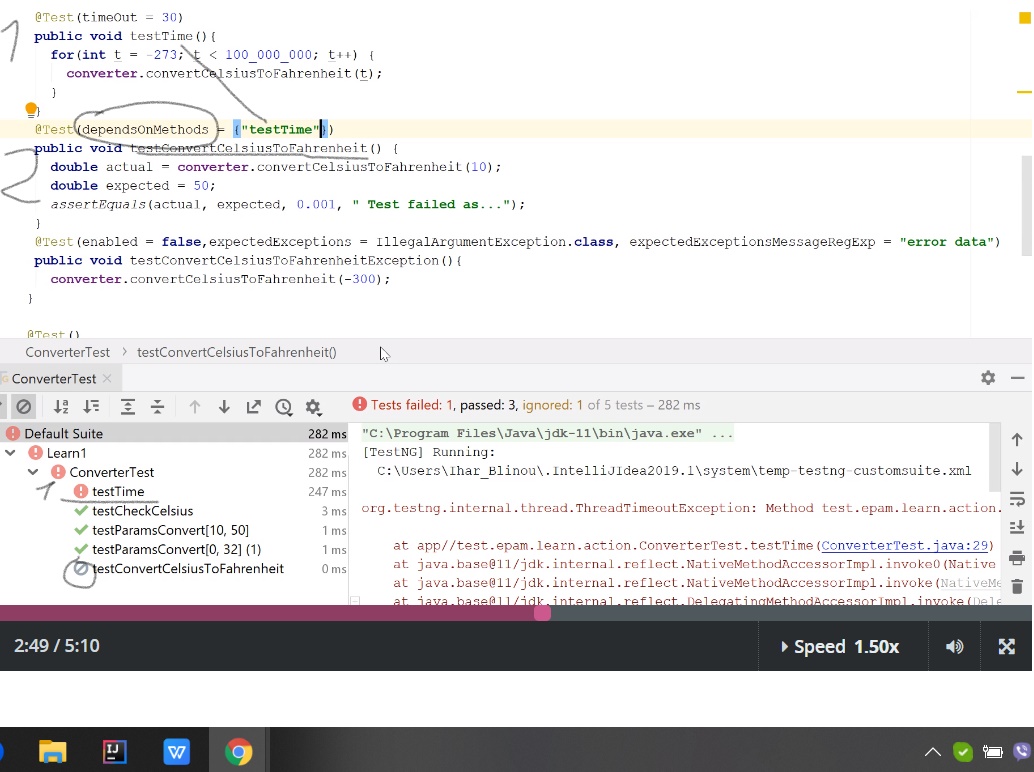
timeOut - ограничение выполнения теста по времени

В тестах только 1 assert, никаких ветвлений (if, while, switch)

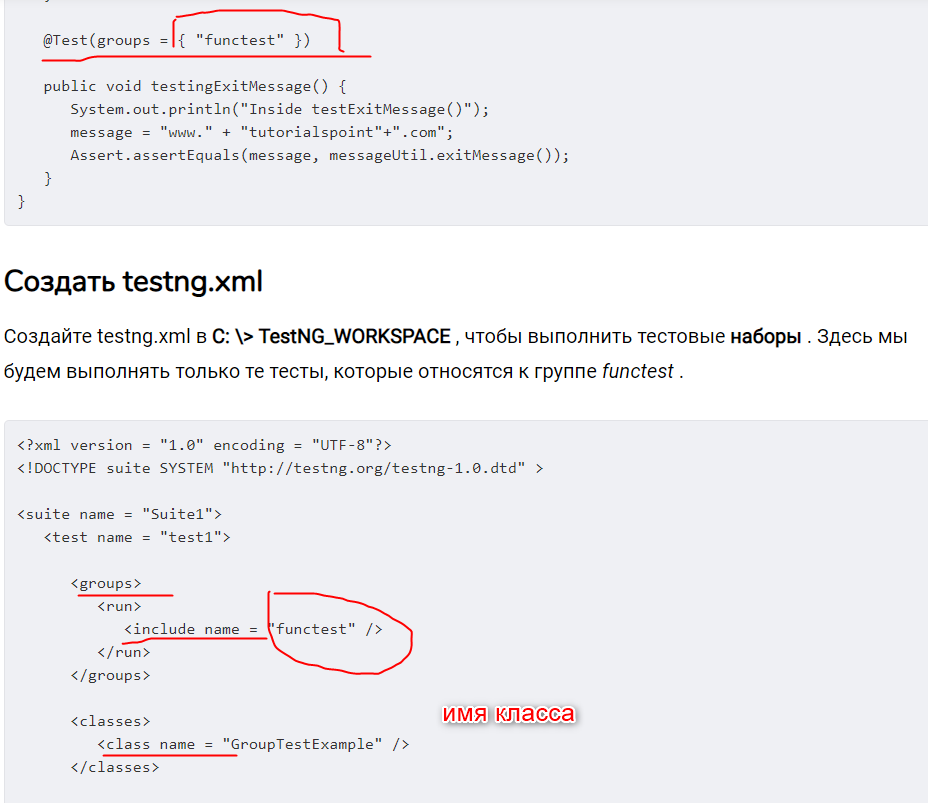
Если 1 тест не пройдет, то 2 не запускать.

Тест №1 будет запускаться первым вне зависимости от порядкового номера, далее - по алфавиту, последний тест - где есть зависимости

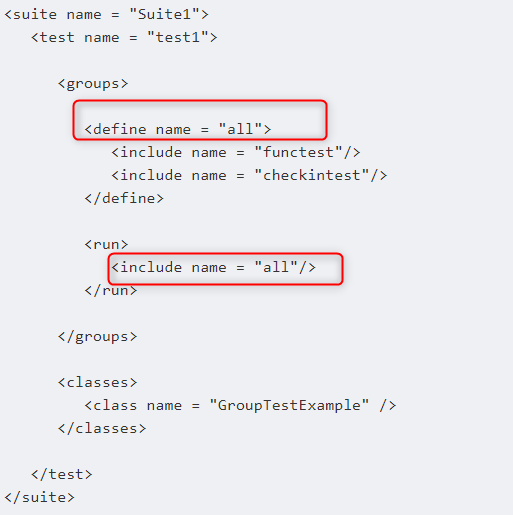
Также можно добавлять приоритет методов (priority = от 1 до 10)



Настройка групп!!



Группы могут также включать другие группы. Эти группы называются **метагруппами**



Вы можете игнорировать группу, используя тег <exclude>, как показано ниже —



<https://www.youtube.com/watch?v=H80gMTErsQQ>

<https://www.youtube.com/watch?v=7CeaD1TZZj8>

### Распараллеливание методов

Для того, чтобы понять в каком потоке выполняется тот или иной тест добавим код в каждый метод:

long threadId = Thread.currentThread().getId();

System.out.println("Thread <ИМЯ МЕТОДА>:" + threadId);

Запустив тест командой mvn test -Dparallel="methods" мы получили нужный нам результат: тесты выполнились параллельно, в разных потоках.

Второй способ, это использовать аннотацию "@DataProvider":

@DataProvider(name = "api", parallel = true)

public Object[] provide() {

return new Object[]{

"post", "get"

};

}

@Test(dataProvider = "api")

public void test(String method) {

if (method.equals("post")) {

Date date = new Date();

System.out.println("Начало " + method + " "+ new Timestamp(date.getTime()));

testPOST();

date=new Date();

System.out.println("Конец " + method + " "+ new Timestamp(date.getTime()));

return;

}

if (method.equals("get")) {

Date date = new Date();

System.out.println("Начало " + method + " " +new Timestamp(date.getTime()));

testGet();

date=new Date();

System.out.println("Конец " + method + " "+ new Timestamp(date.getTime()));

return;

}

}

В аннотации "@Test" появился параметр, который сообщает, какой DataProvider использовать.  
Также в методе test появился входной параметр типа String, куда provider поместит значение.

Fixtures - определенный тип аннотаций.

@After/BeforeClass - вызываются 1 раз перед вызовом всех методов класса. Например, создание объекта.

Прежде всего, метод beforeSuite () выполняется только один раз.

Наконец, метод afterSuite () выполняется только один раз.

Даже методы beforeTest (), beforeClass (), afterClass () и afterTest () выполняются только один раз.

Между beforeMethod () и afterMethod () выполняется каждый тестовый пример.

+- before suite/

+- before group/

+- before test/

+- before class/

+- before method/

+- test/

+- after method/ ...

+- after class/ ...

+- after test/ ...

+- after group/ ...

+- after suite/

Аннотации *@BeforeSuite*, *@AfterSuite* обозначают методы, которые исполняются единожды до/после исполнения всех тестов. Здесь удобно располагать какие-либо тяжелые настройки общие для всех тестов, например, здесь можно создать пул соединений с базой данных.

<https://coderlessons.com/tutorials/java-tekhnologii/uchitsia-testng/testng-kratkoe-rukovodstvo>

<https://riptutorial.com/ru/testng/example/21265/testng-metagroups---%D0%B3%D1%80%D1%83%D0%BF%D0%BF%D1%8B-%D0%B3%D1%80%D1%83%D0%BF%D0%BF>