**Создание набора тестов (Test Suite) с помощью JUnit**

[**https://www.baeldung.com/java-junit-test-suite**](https://www.baeldung.com/java-junit-test-suite)

1. Введение

2. Создание набора тестов

2.1. Использование аннотации @Suite

2.2. Использование аннотации @RunWith для JUnit 4

3. Включение и исключение тестов

3.1. @SelectClasses

3.2. @SelectPackages

3.3. @IncludePackages и @ExcludePackages

3.4. @IncludeClassNamePatterns и @ExcludeClassNamePatterns

3.5. @IncludeTags и @ExcludeTags

3.6. @SelectMethod

**Видео - https://www.youtube.com/watch?v=YsopPRX5gLg**

1. Введение

JUnit — одна из самых популярных платформ тестирования для приложений Java, предлагающая мощный и гибкий способ создания автоматизированных модульных тестов. Одной из его особенностей является возможность создавать наборы тестов, что позволяет нам группировать несколько тестов.

В этом уроке мы рассмотрим, как создавать наборы тестов с помощью JUnit. Сначала мы реализуем и запустим простой набор тестов. После этого мы рассмотрим некоторые конфигурации, чтобы включить или исключить некоторые тесты.

2. Создание набора тестов

Как мы знаем, набор тестов — это набор тестов, сгруппированных вместе и выполняемых как единое целое. Мы используем их для организации тестов в логические группы, например, тесты для определенного компонента или функции приложения. Мы также можем легко выполнять тесты в определенном порядке или запускать подмножество тестов на основе определенных критериев.

JUnit 5 предоставляет несколько способов создания набора тестов. Прежде чем мы начнем, нам нужно убедиться, что мы включили зависимость junit-platform-suite:

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-suite</artifactId>

<version>1.10.1</version>

<scope>test</scope>

</dependency>

Механизм JUnit Platform Suite отвечает за запуск специального набора тестов с использованием одного или нескольких механизмов тестирования в JUnit. Он также предоставляет нам дополнительные API, которые мы будем использовать для создания набора тестов.

2.1. Использование аннотации @Suite

Самый простой способ реализовать набор тестов — использовать аннотацию уровня класса @Suite, которая также является наиболее рекомендуемым решением. Эта аннотация доступна начиная с версии платформы JUnit 1.8.0.

Давайте подготовим класс JUnitTestSuite:

*@Suite*

*@SuiteDisplayName("My Test Suite")*

*public class JUnitTestSuite {*

*}*

В этом примере мы использовали аннотацию @Suite, сообщающую JUnit пометить этот класс как исполняемый в одном модуле. Кроме того, мы добавили необязательную аннотацию @SuiteDisplayName и указали собственный заголовок.

С этого момента мы можем использовать этот класс для выполнения всех тестов, настроенных в этом наборе, за один запуск, используя нашу IDE или плагин maven-surefire-plugin. Обратите внимание, что на данный момент этот пакет не включает никаких тестов.

2.2. Использование аннотации @RunWith для JUnit 4

В качестве альтернативы мы можем переписать наш набор тестов, используя аннотацию @RunWith, которая использует модель “runer” для JUnit 4:

*@RunWith(JUnitPlatform.class)*

*@SuiteDisplayName("My Test Suite")*

*public class JUnitRunWithSuite {*

*}*

Если нам не хватает класса JUnitPlafrom, нам нужно включить дополнительную зависимость:

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-runner</artifactId>

<version>1.10.1</version>

<scope>test</scope>

</dependency>

В результате этот набор тестов работает аналогично нашему предыдущему, созданному с помощью аннотации @Suite. Это решение рекомендуется только разработчикам, использующим более ранние версии JUnit. Кроме того, с версии 1.8.0 средство запуска JUnitPlatform устарело в пользу поддержки @Suite и будет удалено в будущих выпусках.

3. Включение и исключение тестов

На данный момент наш набор тестов не содержит никаких тестов. JUnit Platform Suite Engine предоставляет несколько аннотаций для включения или исключения тестов в наших наборах тестов.

Можно выделить две основные группы аннотаций: @Select и @Include/@Exclude.

Аннотации @Select указывают ресурсы, где JUnit должен искать тесты. Между тем, аннотации @Include/@Exclude указывают дополнительные условия для включения или исключения ранее найденных тестов.

Эти две аннотации не будут работать без аннотации @Select. Мы можем смешивать все аннотации, чтобы точно определить, какие тесты мы хотим запустить.

Теперь давайте взглянем на некоторые из них, настроив наш набор тестов.

3.1. @SelectClasses

Самый распространенный способ выбрать тесты для наших наборов тестов — указать тестовые классы с помощью аннотации @SelectClasses:

*@Suite*

*@SelectClasses({ClassOneUnitTest.class, ClassTwoUnitTest.class})*

*public class JUnitSelectClassesSuite {*

*}*

Набор тестов теперь выполняет все отмеченные @Test методы из обоих классов.

3.2. @SelectPackages

Вместо указания списка классов мы можем использовать @SelectPackages для предоставления пакетов для тестового сканирования:

*@Suite*

*@SelectPackages({"com.baeldung.testsuite", "com.baeldung.testsuitetwo"})*

*public class JUnitSelectPackagesSuite {*

*}*

Примечательно, что эта аннотация также выполняет все классы из подпакетов.

3.3. @IncludePackages и @ExcludePackages

На данный момент мы знаем, как включить все классы из пакетов. Чтобы дополнительно указать, включать или исключать пакеты, мы можем использовать аннотации @IncludePackages и @ExcludePackages соответственно:

*@Suite*

*@SelectPackages({"com.baeldung.testsuite")*

*@IncludePackages("com.baeldung.testsuite.subpackage")*

*public class JUnitTestIncludePackagesSuite {*

*}*

Приведенная выше конфигурация выполняет только все тесты, найденные в пакетах com.baeldung.testsuite.subpackage, игнорируя другие результаты.

Давайте посмотрим, как мы можем исключить один пакет:

*@Suite*

*@SelectPackages("com.baeldung.testsuite")*

*@ExcludePackages("com.baeldung.testsuite.subpackage")*

*public class JUnitExcludePackagesSuite {*

*}*

Теперь JUnit выполняет все тесты, найденные в пакете com.baeldung.testsuite и его подпакетах, игнорируя только классы, найденные в com.baeldung.testsuite.subpackage.

3.4. @IncludeClassNamePatterns и @ExcludeClassNamePatterns

Если мы не хотим указывать правила включения с помощью пакетов, мы можем использовать аннотации @IncludeClassNamePatterns и @ExcludeClassNamePatterns и реализовать проверку регулярных выражений для имен классов:

*@Suite*

*@SelectPackages("com.baeldung.testsuite")*

*@IncludeClassNamePatterns("com.baeldung.testsuite.Class.\*UnitTest")*

*@ExcludeClassNamePatterns("com.baeldung.testsuite.ClassTwoUnitTest")*

*public class JUnitClassNamePatternsSuite {*

*}*

Этот пример включает все тесты, найденные в пакете com.baeldung.testsuite, без его подпакетов. Имя класса должно соответствовать шаблону регулярного выражения Class.\*UnitTest, например ClassOneUnitTest и ClassThreeUnitTest.

Кроме того, мы также строго исключили имя ClassTwoUnitTest, которое будет соответствовать первому условию. Как мы знаем, в Java полное имя класса включает также его пакет. Это также следует учитывать при определении шаблонов.

3.5. @IncludeTags и @ExcludeTags

Как мы знаем, в JUnit мы можем отмечать классы и методы с помощью аннотации @Tag. Это простой способ отфильтровать наши тесты с помощью простых значений. Мы можем использовать тот же механизм при определении наших тестовых случаев, используя @IncludeTags и @ExcludeTags для запуска тестов с указанным @Tag:

*@Suite*

*@SelectPackages("com.baeldung.testsuite")*

*@IncludeTags("slow")*

*public class JUnitTestIncludeTagsSuite {*

*}*

Этот набор тестов будет сканировать пакет com.baeldung.testsuite и все подпакеты, выполняя только тесты, помеченные тегом @Tag («slow»).

Давайте теперь изменим конфигурацию:

*@Suite*

*@SelectPackages("com.baeldung.testsuite")*

*@ExcludeTags("slow")*

*public class JUnitTestExcludeTagsSuite {*

*}*

В этом примере JUnit запускает все тесты без аннотации @Tag(“slow”), включая тесты без тегов.

Стоит отметить, что в JUnit мы также можем пометить один метод внутри тестового класса. **Как и механизм описанный в следующем пункте,** **использование @IncludeTags и @ExcludeTags также позволяет нам включать отдельные методы из классов,** в отличие от предыдущих аннотаций.

3.6. @SelectMethod

Иногда нам нужно фильтровать методы тестирования по имени в наших наборах тестов. Мы можем использовать аннотацию @SelectMethod для решения этой проблемы.

Давайте посмотрим, как мы можем выбрать один метод:

*@Suite*

*@SuiteDisplayName("My Test Suite")*

*@SelectMethod(type = ClassOneUnitTest.class, name = "whenFalse\_thenFalse")* ***или***

*@SelectMethod("com.baeldung.testsuite.subpackage.ClassTwoUnitTest#whenFalse\_thenFalse")*

*public class JUnitSelectMethodsSuite {*

*// runs ClassOneUnitTest and ClassTwoUnitTest*

*}*

Мы можем выбрать класс и метод, используя атрибуты type и name. Атрибут type указывает класс, содержащий методы тестирования. Атрибут name указывает имя тестового метода, который будет выполняться внутри класса.

Кроме того, мы можем указать полное имя тестового класса и имя тестового метода с символом «#» в качестве разделителя.

В приведенном выше примере наш набор тестов включает метод WhenFalse\_thenFalse() из классов ClassOneUnitTest и ClassTwoUnitTest, поэтому при запуске набора будет выполняться только выбранный метод из этих классов.

4. Вывод

JUnit предоставляет простой способ создания автоматизированных тестов, включая возможность создания наборов тестов. Мы можем использовать наборы тестов для организации наших тестов в логические группы, запуска набора тестов в определенном порядке или запуска подмножества тестов на основе определенных критериев.