



**11 лекция**  
**java for web**  
**Junit**

# Что такое JUnit ?

JUnit — один из самых распространенных framework для тестирования программного обеспечения на языке Java.

Вся его задача состоит в том, чтобы удобно запустить некий класс, который будет состоять из функций предназначенных для тестирования Вашего приложения.

В нем все предназначено только для одной цели — удобно писать и запускать тесты, которые Вы придумали сами.

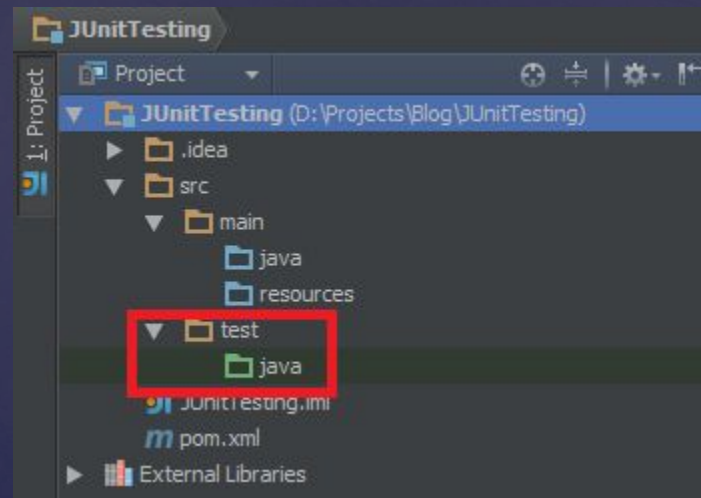
# Использование JUnit

Для начало нам нужно подключить зависимость JUnit в pom.xml не забываем что для удобства мы используем Maven.

```
<dependency>  
  <groupId>junit</groupId>  
  <artifactId>junit</artifactId>  
  <version>4.11</version>  
  <scope>test</scope>  
</dependency>
```

# Использование JUnit

Обязательно проверьте что папка, которая лежит в test/java должна быть зеленым цветом это будет обозначать то что в данной папке лежат тестовые классы и при сборке проекта они не будут собираться в проект.



# Использование JUnit

Допустим у нас есть класс, в котором есть метод, которые выполняет какие то действия, например суммирует какие то числа, это и будет наша логика, которую нужно протестировать.

```
public class Calculate {  
    public int calA(int a, int b){  
        return a+b;  
    }  
}
```

Unit тест с технической стороны — это класс который лежит в тестовом ресурсе и который предназначен только для тестирование логики, а не для использования в production коде.

# Пример JUnit теста

```
public class CalculateTest {  
    @Test  
    public void testCalA() throws Exception {  
        Calculate calculate = new Calculate();  
        int n = calculate.calA(2, 2);  
  
        assertEquals(4, n);  
    }  
}
```

В JUnit предполагается, что все тестируемые методы могут быть выполнены в произвольном порядке. Поэтому тесты не должны зависеть от других тестов.

Для того чтобы указать что данный метод есть тестовым его нужно про аннотировать **@Test** после чего данный метод можно будет запускать в отдельном потоке для проведения тестирования.

# Доступные аннотации JUnit

Аннотация	Описание
<b>@Test</b> public void method()	Аннотация <b>@Test</b> определяет что метод method() является тестовым.
<b>@Before</b> public void method()	Аннотация <b>@Before</b> указывает на то, что метод будет выполняться перед каждым тестируемым методом <b>@Test</b> .
<b>@After</b> public void method()	Аннотация <b>@After</b> указывает на то что метод будет выполняться после каждого тестируемого метода <b>@Test</b>
<b>@BeforeClass</b> public static void method()	Аннотация <b>@BeforeClass</b> указывает на то, что метод будет выполняться в начале всех тестов, а точнее в момент запуска тестов(перед всеми тестами <b>@Test</b> ).
<b>@AfterClass</b> public static void method()	Аннотация <b>@AfterClass</b> указывает на то, что метод будет выполняться после всех тестов.
<b>@Ignore</b>	Аннотация <b>@Ignore</b> говорит, что метод будет проигнорирован в момент проведения тестирования.
<b>@Test (expected = Exception.class)</b>	<b>(expected = Exception.class)</b> — указывает на то, что в данном тестовом методе вы преднамеренно ожидается Exception.
<b>@Test (timeout=100)</b>	<b>(timeout=100)</b> — указывает, что тестируемый метод не должен занимать больше чем 100 миллисекунд.

# Проверяемые методы (основные)

Метод	Описание
<code>fail(String)</code>	Указывает на то что бы тестовый метод завалился при этом выводя текстовое сообщение.
<code>assertTrue([message], boolean condition)</code>	Проверяет, что логическое условие истинно.
<code>assertEquals([String message], expected, actual)</code>	Проверяет, что два значения совпадают. <i>Примечание:</i> для массивов проверяются ссылки, а не содержание массивов.
<code>assertNull([message], object)</code>	Проверяет, что объект является пустым <code>null</code> .
<code>assertNotNull([message], object)</code>	Проверяет, что объект не является пустым <code>null</code> .
<code>assertSame([String], expected, actual)</code>	Проверяет, что обе переменные относятся к одному объекту.
<code>assertNotSame([String], expected, actual)</code>	Проверяет, что обе переменные относятся к разным объектам.



# Пример JUnit теста

```
public class CalculateTest {  
    @Test  
    public void testCalA() throws Exception {  
        Calculate calculate = new Calculate();  
        int n = calculate.calA(2, 2);  
  
        assertEquals(4, n);  
    }  
}
```

В JUnit предполагается, что все тестируемые методы могут быть выполнены в произвольном порядке. Поэтому тесты не должны зависеть от других тестов.

Для того чтобы указать что данный метод есть тестовым его нужно про аннотировать **@Test** после чего данный метод можно будет запускать в отдельном потоке для проведения тестирования.

# Запуск JUnit теста

IDE умеют находить и просто запускать тесты в проекте. Но что делать, если вы хотите запустить их вручную с помощью программного кода. Для этого воспользуемся классом JUnitCore. Добавьте следующий метод main () в наш класс с Тестами:

```
public static void main(String[] args) throws Exception {  
    JUnitCore runner = new JUnitCore();  
    Result result = runner.run(MathFuncTest.class);  
  
    System.out.println("run tests: " + result.getRunCount());  
    System.out.println("failed tests: " + result.getFailureCount());  
    System.out.println("ignored tests: " + result.getIgnoreCount());  
    System.out.println("success: " + result.wasSuccessful());  
}
```