Junit test suite-

1. Junit test suites provides us facility to run tests which are spread into multiple test classes and different packages.
2. Junit 5 provides us mainly two annotations
   1. @SelectPackages
   2. @SelectClasses

To create test suites.

1. Apart from this we may have to use other annotations for filtering test packages, classes or even test methods.
2. **JUnit 5 test suites** are written with **@Suite** annotation. Suites help us run the tests spread into multiple classes and packages.
3. We can use Include and Exclude annotations for filtering test packages, test classes or even test methods.
4. ~~@RunWith(JUnitPlatform.class)~~ has been deprecated in favor of @Suite annotation and will be removed in JUnit Platform 2.0
5. To run the suites, include junit-platform-suite-engine dependency (version 1.8 or later).

pom.xml

<dependency>

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

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

<version>1.8.1</version>

</dependency>

**2. Creating Test Suites**

**2. Creating Test Suites**

**2.1. @Suite**

Creating suites is easy. Just add the @Suite annotation of a class and start including or excluding the test classes and methods into the suite.

When we want to run the suite, simply run it as a normal JUnit test class and it will execute all the included tests in the suite.

An example of test suite in JUnit 5

**import** org.junit.platform.suite.api.IncludeTags;

**import** org.junit.platform.suite.api.SelectPackages;

**import** org.junit.platform.suite.api.Suite;

@SelectPackages({"com.howtodoinjava.junit5.examples.packageA"

,"com.howtodoinjava.junit5.examples.packageB"})

@IncludeTags("production")

@Suite

**public** **class** JUnit5TestSuiteExample {

}

**3.2. @SuiteDisplayName**

Use this annotation to give a display name for the annotated test class that is executed as a test suite on the JUnit Platform.

Display names are typically used for test reporting in IDEs and build tools and may contain spaces, special characters, and even emoji.

@IncludeTags("production")

@Suite

@SuiteDisplayName("A demo Test Suite")

**public** **class** JUnit5TestSuiteExample {

}

**3. Including and Excluding Tests**

JUnit 5 provides the following annotations to include or exclude the tests in the suites.

* *@SelectClasses*
* *@SelectPackages*
* *@IncludePackages*
* *@ExcludePackages*
* *@IncludeClassNamePatterns*
* *@ExcludeClassNamePatterns*
* *@IncludeTags*
* *@ExcludeTags*

Let’s learn about these annotations in detail.

**3.1. @SelectPackages**

@SelectPackages specifies the names of packages to select when running a test suite via @RunWith(JUnitPlatform.class).

**Specify Single Package**

Pass “packageName” as parameter to @SelectPackages annotation.

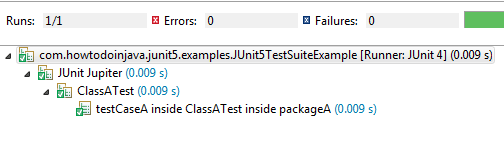
@Suite

@SelectPackages("com.howtodoinjava.junit5.examples.packageA")

**public** **class** JUnit5TestSuiteExample

{

}

@SelectPackages – Single Package Example

**Specify Multiple Packages**

Pass package names in the parameter as a string array (inside curly braces {}) to @SelectPackages annotation.

@Suite

@SelectPackages({"com.howtodoinjava.junit5.examples.packageA",

"com.howtodoinjava.junit5.examples.packageB"})

**public** **class** JUnit5TestSuiteExample

{

}

### 3.2. @SelectClasses

@SelectClasses specifies the classes to select when running a test suite via @RunWith(JUnitPlatform.class).

#### Specify Single Test Class

Pass ClassName.class as parameter to @SelectClasses annotation.

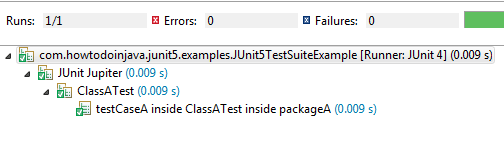
@Suite

@SelectClasses( ClassATest.**class** )

**public** **class** JUnit5TestSuiteExample

{

}

@SelectClasses – Single Class Example

#### Specify Multiple Test Classes

Pass class names in parameter as array (inside curly braces {}) to @SelectClasses annotation.

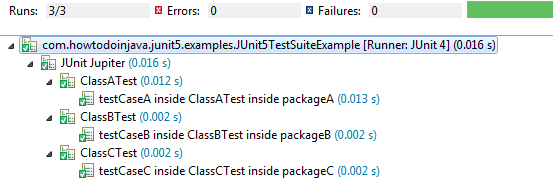
@Suite

@SelectClasses( { ClassATest.**class**, ClassBTest.**class**, ClassCTest.**class** } )

**public** **class** JUnit5TestSuiteExample

{

}

@SelectClasses – Multiple Test Classes Example

### 3.3. @IncludePackages and @ExcludePackages

As we learn that @SelectPackages causes all its sub-packages as well to be scanned for test classes.

If you want to exclude any specific package or include any package then you may use [@IncludePackages](https://junit.org/junit5/docs/current/api/org.junit.platform.suite.api/org/junit/platform/suite/api/IncludePackages.html) and [@ExcludePackages](https://junit.org/junit5/docs/current/api/org.junit.platform.suite.api/org/junit/platform/suite/api/ExcludePackages.html) annotations.

#### @IncludePackages Example

@Suite

@SelectPackages("com.howtodoinjava.junit5.examples")

@IncludePackages("com.howtodoinjava.junit5.examples.packageC")

**public** **class** JUnit5TestSuiteExample

{

}

This will add tests from test classes in com.howtodoinjava.junit5.examples.packageC only i.e. ClassCTest.

#### @ExcludePackages Example

@Suite

@SelectPackages("com.howtodoinjava.junit5.examples")

@ExcludePackages("com.howtodoinjava.junit5.examples.packageC")

**public** **class** JUnit5TestSuiteExample

{

}

This will add tests from test classes in com.howtodoinjava.junit5.examples but exclude all test classes from sub-package com.howtodoinjava.junit5.examples.packageC i.e. ClassATest and ClassBTest.

### 3.4. @IncludeClassNamePatterns and @ExcludeClassNamePatterns

Many times it is not feasible to include all packages or test class names in select annotations. In that case, you may give a broader package scope and apply filtering on which test classes to be included or excluded from the suite.

To specify test class names patterns to exclude or include, you can use [@IncludeClassNamePatterns](https://junit.org/junit5/docs/current/api/org.junit.platform.suite.api/org/junit/platform/suite/api/IncludeClassNamePatterns.html) and [@ExcludeClassNamePatterns](https://junit.org/junit5/docs/current/api/org.junit.platform.suite.api/org/junit/platform/suite/api/ExcludeClassNamePatterns.html) annotations.

#### @IncludeClassNamePatterns Example

Include all test classes with names ending with ATest or ATests.

@Suite

@SelectPackages("com.howtodoinjava.junit5.examples")

@IncludeClassNamePatterns({"^.\*ATests?$"})

**public** **class** JUnit5TestSuiteExample

{

}

#### @ExcludeClassNamePatterns Example

Exclude all test classes with names ending with ATest or ATests.

@Suite

@SelectPackages("com.howtodoinjava.junit5.examples")

@ExcludeClassNamePatterns({"^.\*ATests?$"})

**public** **class** JUnit5TestSuiteExample

{

}

You may apply more than one pattern in above annotations. In case of multiple patterns, they are combined using OR semantics.

It means that if fully qualified name of a class **matches against at least one of the patterns**, the class will be included/excluded from the test suite.

### 3.5. @IncludeTags and @ExcludeTags

In enterprise applications, you may have tagged test cases that you want to run in specific environments e.g. development or production. You can include or exclude tests based on these tags as well, from a test suite.

@Tag(“dev”)

#### @IncludeTags Example

This test suite will run all tests tagged with production inside package com.howtodoinjava.junit5.examples (and its sub-packages).

@Suite

@SelectPackages("com.howtodoinjava.junit5.examples")

@IncludeTags("production")

**public** **class** JUnit5TestSuiteExample

{

}

#### @ExcludeTags Example

This test suite will exclude all tests tagged with development inside package com.howtodoinjava.junit5.examples (and it’s sub-packages).

@Suite

@SelectPackages("com.howtodoinjava.junit5.examples")

@ExcludeTags("development")

**public** **class** JUnit5TestSuiteExample

{

}

Clearly, there are multiple ways to **create test suites in JUnit 5** and it has strong support for **filtering tests to/from test suites**.

Timeout 🡪

## @Timeout Annotation

* **JUnit 5 uses the declarative way** to define the timeout behavior of a given test **using the *@Timeout* annotation**. It is unlike [JUnit 4 @Test timeout attribute](https://howtodoinjava.com/junit/how-to-force-timeout-in-junit/).
* A **timeout configured test should fail if its execution time exceeds a given duration**. The execution of the test proceeds in the main thread. If the timeout is exceeded, the main thread is interrupted from another thread.
* The **default time unit for the timeout duration is in *seconds***. We can configure other time units as well.
* @Timeout Syntax
* @BeforeEach
* @Timeout(2)
* **void** setUp() {
* *// fails if execution time exceeds 2 seconds*
* }
* @Test
* @Timeout(3)
* **void** someTest() {
* TimeUnit.SECONDS.sleep(5);
* }

**2. Global Timeouts**

* To configure the timeout for all the tests and nested tests, we can **apply the @Timeout annotation at the class level**.
* @Timeout annotations declared at the class level are **not applied to lifecycle methods**.
* We **can override the timeout on a certain test** by applying the @Timeout annotation on that test.
* When applied with [*@RepeatedTest*](https://howtodoinjava.com/junit5/repeated-test-annotation-example/) or [*@ParameterizedTest*](https://howtodoinjava.com/junit5/parameterized-tests/), the timeout is applied on each invocation of the test.

In the given example, testMethodOne will FAIL and testMethodTwo will PASS.

@Timeout(3)

**public** **class** TimeoutTests {

@Test

**void** testMethodOne() **throws** InterruptedException {

TimeUnit.SECONDS.sleep(5);

}

@Test

@Timeout(5)

**void** testMethodTwo() **throws** InterruptedException {

TimeUnit.SECONDS.sleep(4);

}

}

String getValue() **throws** InterruptedException {

TimeUnit.SECONDS.sleep(2);

**return** "";

}

**Parameterized Test**

**Parameterized Test:** Parameterized test makes it possible to run a test multiple times with different arguments. They are simply declare just like regular @Test methods but have to use @ParameterizedTest annotation.

* + 1. We must have to at least declare one source that will provide the arguments for each invocation and then consume arguments in tests.
    2. Example-

@ParameterizedTest

@ValueSource(strings={“madam”,”radar”,”dad”})

Void palindromeTest(String input\_string){

assertThat

}