@Before

Before hooks run before the first step of each scenario.

Whatever happens in a Before hook is invisible to people who only read the features. You should consider using a [background](https://cucumber.io/docs/gherkin/reference#background) as a more explicit alternative, especially if the setup should be readable by non-technical people. Only use a Before hook for low-level logic such as starting a browser or deleting data from a database.

You can specify an explicit order for hooks if you need to.

Annotated method style:

@Before(order = 10)

public void doSomething(){

// Do something before each scenario

}

**After**

After hooks run after the last step of each scenario, even when the step result is failed, undefined, pending, or skipped.

Annotated method style:

@After

public void doSomethingAfter(Scenario scenario){

// Do something after after scenario

}

### Around

Cucumber-JVM does not support Around hooks.

## Step hooks

Step hooks invoked before and after a step. The hooks have ‘invoke around’ semantics. Meaning that if a BeforeStep hook is executed the AfterStep hooks will also be executed regardless of the result of the step. If a step did not pass, the following step and its hooks will be skipped.

### BeforeStep

@BeforeStep

public void doSomethingBeforeStep(Scenario scenario){

}

### AfterStep

@AfterStep

public void doSomethingAfterStep(Scenario scenario){

}

## Conditional hooks

Hooks can be conditionally selected for execution based on the tags of the scenario. To run a particular hook only for certain scenarios, you can associate a Before or After hook with a [tag expression](https://cucumber.io/docs/cucumber/api/#tag-expressions).

Annotated method style:

@After("@browser and not @headless")

public void doSomethingAfter(Scenario scenario){

}

## Global hooks

Cucumber-JVM does not support global hooks.

## Running a hook only once

Cucumber-JVM does not support running a hook only once.

## AfterConfiguration

Cucumber-JVM does not support AfterConfiguration hooks.

# Tags

Tags are a great way to organise your features and scenarios.

They can be used for two purposes:

## Running a subset of scenarios

You can tell Cucumber to only run scenarios with a particular tag:

Using a JVM system property:

mvn test -Dcucumber.filter.tags="@smoke and @fast"

Or an environment variable:

# Linux / OS X:

CUCUMBER\_FILTER\_TAGS="@smoke and @fast" mvn test

# Windows:

set CUCUMBER\_FILTER\_TAGS="@smoke and @fast"

mvn test

Or changing your JUnit runner class:

@CucumberOptions(tags = "@smoke and @fast")

public class RunCucumberTest {}

Consider the following example:

@billing

Feature: Verify billing

@important

Scenario: Missing product description

Given hello

Scenario: Several products

Given hello

A feature or scenario can have as many tags as you like. Separate them with spaces:

@billing @bicker @annoy

Feature: Verify billing

Tags can be placed above the following Gherkin elements:

* Feature
* Scenario
* Scenario Outline
* Examples

It is *not* possible to place tags above Background or steps (Given, When, Then, And and But).

## Tag Inheritance

Tags are inherited by child elements.

Tags that are placed above a Feature will be inherited by Scenario, Scenario Outline, or Examples.

Tags that are placed above a Scenario Outline will be inherited by Examples.

## Ignoring a subset of scenarios

You can tell Cucumber to ignore scenarios with a particular tag:

Using JUnit runner class:

@CucumberOptions(tags = "not @smoke")

public class RunCucumberTest {}

**Filtering by line**

Another way to run a subset of scenarios is to use the file.feature:line pattern or the --scenario option.

### Tag expressions

A tag expression is an infix boolean expression. Below are some examples:

| **Expression** | **Description** |
| --- | --- |
| @fast | Scenarios tagged with @fast |
| @wip and not @slow | Scenarios tagged with @wip that aren’t also tagged with @slow |
| @smoke and @fast | Scenarios tagged with both @smoke and @fast |
| @gui or @database | Scenarios tagged with either @gui or @database |

For even more advanced tag expressions you can use parenthesis for clarity, or to change operator precedence:

(@smoke or @ui) and (not @slow)