Feature: <feature description

Multiline description>

Background : <definition>

Given ….

#indent by two spaces

@tag #comment, require at-least one tag else throws error

Scenario: <scenario desc>

Given

When

Then

And

‘’’

Multiline statement

‘’’

Scenario Outline: <scenario desc>

Given

….

Examples:

|val1|val2|val3|

|val4|val5| |

**The primary keywords are:**

Feature : high-level description of a software feature, and to group related scenarios

Rule (as of Gherkin 6)

* represent one business rule that should be implemented
* group together several scenarios that belong to this business rule
* should contain one or more scenarios that illustrate the particular rule

Feature: Highlander

Rule: There can be only One

Example: Only One -- More than one alive

Given there are 3 ninjas

And there are more than one ninja alive

Example: Only One -- One alive

Given there is only 1 ninja alive

Then he (or she) will live forever ;-)

Rule: There can be Two (in some cases)

Example: Two -- Dead and Reborn as Phoenix

Example (or Scenario) : recommend 3-5 steps per example

Given, When, Then, And, But for steps (or \*)

Background :

* allows you to add some context to the scenarios that follow it
* is placed before the first Scenario/Example
* also supported at the Rule level
* can only have one set of Background steps per Feature or Rule

Scenario Outline (or Scenario Template)

Examples (or Scenarios)

**There are a few secondary keywords as well:**

""" or ``` (Doc Strings) :

* this will be automatically passed as last argument
* possible to annotate the DocString with the type of content it contains

Ex : “””markdown , “””json

| (Data Tables) :

* Just like Doc Strings, Data Tables will be passed to the step definition as the last argument

| firstName | lastName | birthDate |

| Annie M. G. | Schmidt | 1911-03-20 |

| Roald | Dahl | 1916-09-13 |

**java type: List<List<String>>**

[

[ "firstName", "lastName", "birthDate" ],

[ "Annie M.G", "Schmidt", "1911-03-20" ]

]

**java type: List<Map<String, String>>**

[

{ "firstName": "Annie M.G", "lastName": "Schmidt", "birthDate": "1911-03-20" },

{ "firstName": "Roald", "lastName": "Dahl", "birthDate": "1916-09-13" }

]

| KMSY | Louis Armstrong New Orleans International Airport |

| KSFO | San Francisco International Airport |

**java type: Map<String,String>**

{

"KMSY": "Louis Armstrong New Orleans International Airport",

"KSFO": "San Francisco International Airport"

}

| KMSY | 29.993333 | -90.258056 |

| KSFO | 37.618889 | -122.375000 |

**java type: Map<String,List<Double>>**

{

"KMSY": [29.993333, -90.258056],

…..

}

| | lat | lon |

| KMSY | 29.993333 | -90.258056 |

| KSFO | 37.618889 | -122.375000 |

\*\* Note that the first cell has been left blank. This tells the table that the map's keys should be created from the first column rather than the header.

**java type: Map<String, Map<String, String>>**

{

"KMSY": { "lat": "29.993333", "lon": "-90.258056" },

"KSFO": { "lat": "37.618889", "lon": "-122.375000" }

}

**Custom Table Types**

There are two helpers for defining custom table types:

// Defines a DataTableType that converts an entry (map of header name to row value)

// to an object, using reflection.

registry.defineDataTableType(DataTableType#entry(Class))

// Defines a DataTableType that converts a single cell

// to an object, by calling its `String` constructor (if it exists).

registry.defineDataTableType(DataTableType#cell(Class))

@ (Tags) : Tags that are placed above a Feature will be inherited by Scenario, Scenario Outline, or Examples

Running tags , **(@smoke or @ui) and (not @slow)**

# (Comments)

# language: no ##tells german language. First line of feature file

Funksjonalitet: Gjett et ord

**Define step definition:**

Given the following “animals”:

| cow |

| horse |

@Given("the following \“(.\*)\”:")

public void the\_following\_animals(String object, List<String> animals) {

}

**Step Results:**

Success

Undefined

Pending

Failed Steps

Skipped

Ambiguous : duplicate step definition, AmbiguousStepDefinitionsException

**Hooks:** put under **features/support** directory to avoid duplicate, used for setup and teardown of the environment before and after each scenario.

Before: run before the first step of each scenario

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

Around : doesn’t support

**Step hooks:** Step hooks invoked before and after a step

BeforeStep

AfterStep

**Global hooks:** will run once before any scenario is run or after all scenario have been run

BeforeAll

AfterAll

Put the code at the top-level in your env.rb file (or any other file under **features/support** directory).

**specify an explicit order for hooks if you need to.**

@Before(order = 10)

**Conditional hooks:**

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

public void doSomethingAfter(Scenario scenario){

}

**Running Cucumber:**

**CLI runner:** need cucumber-core andall of its transitive dependencies to your classpath

java -cp "path/to/each/jar:path/to/compiled/.class/files" io.cucumber.core.cli.Main /path/to/your/feature/files --glue hellocucumber --glue anotherpackage

**using maven:**

mvn exec:java \

-Dexec.classpathScope=test \

-Dexec.mainClass=io.cucumber.core.cli.Main \

-Dexec.args="/path/to/your/feature/files --glue hellocucumber --glue anotherpackage"

**Using Junit:** cucumber-junit package

Create runner class:

package com.example;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

import org.junit.runner.RunWith;

@RunWith(Cucumber.class)

@CucumberOptions(

features = “path to feature files”,

glue = “step-def package name”,

tags = {"@foo and not @bar"},

monochrome=true,

dryRun=true,

plugin = {"pretty", "html:target/cucumber"},

snippets = CAMELCASE, //code snippet for missing step-def

)

public class RunCucumberTest {

}

**Supported properties are:**

cucumber.ansi-colors.disabled= # true or false. default: false

cucumber.execution.dry-run= # true or false. default: false

cucumber.execution.limit= # number of scenarios to execute (CLI only).

cucumber.execution.order= # lexical, reverse, random or random:[seed] (CLI only). default: lexical

cucumber.execution.wip= # true or false. default: false.

cucumber.features= # comma separated paths to feature files. example: path/to/example.feature, path/to/other.feature

cucumber.filter.name= # regex. example: .\*Hello.\*

cucumber.filter.tags= # tag expression. example: @smoke and not @slow

cucumber.glue= # comma separated package names. example: com.example.glue

cucumber.plugin= # comma separated plugin strings. example: pretty, json:path/to/report.json

cucumber.object-factory= # object factory class name. example: com.example.MyObjectFactory

cucumber.snippet-type= # underscore or camelcase. default: underscore