Сравнительная характеристика Jbehave и Cucumber

1. Documentation

|  |  |  |
| --- | --- | --- |
| Feature | Available in  JBehave | Available in  Cucumber |
| Official site | JBehave Official site | There's no dedicated site for Cucumber-JVM however it's quite stupid to say that there's no official site at all. |
| Getting started guide | Getting started page | The information about that is distributed between various samples and blog posts but there's no dedicated page showing all steps in Java |
| General Usage guide | Introduction  Core Reference | The reason is the same as before. There's no dedicated reference to Java usage as there're too few Java-specific features |
| API documentation | JBehave Javadoc modules | Javadoc main page |
| Code examples | JBehave examples: A dime a dozen modules | Cucumber examples |
| Specialized forums | JBehave - BDD LinkedIn Group | Cucumber group on LinkedIn - quite populated place with big number of active discussions  Cukes Google group |

1. Flexibility in passing parameters

|  |  |  |
| --- | --- | --- |
| Feature | Available in  JBehave | Available in  Cucumber |
| Parameter variants | Pattern Variances | Sample code |
| Parameters injection | Reference page | Something similar |
| Tabular parameters | Tabular parameters in JBehave | DataTable samples |
| Examples | Parametrized scenarios | Scenario Outlines |
| Multiline input | - | Docstrings sample |
| Formatted input | Parameter converters - nice feature for passing complex types within one instruction + the ability to customize conversion | Sample code containing an example. In particular:  @Given("^today is (.+)$")  public void today\_is(@Format("yyyy-MM-dd") Date date) {  calculator = new DateCalculator(date);  } |

1. Scoping

|  |  |  |
| --- | --- | --- |
| Feature | Available in  JBehave | Available in  Cucumber |
| Feature/Story scope | Meta-filtering | Tags |
| Scenario scope | Meta-filtering | Tags |
| Step scope | The scope of classes is defined during at the configuration so steps are initially scoped by design | For Cucumber it's treated as rather antipattern than some built-in solution |

1. Built-in reports

|  |  |  |
| --- | --- | --- |
| Report type | Available in  JBehave | Available in  Cucumber |
| Console output | + | + |
| Pretty console output | - | + |
| Structured file (e.g. XML) | + | + |
| Well-formatted readable file (e.g. HTML) | + | + |
| Usage report | + | + |
| Extra report types | - | + |

The JBehave has PrintStreamOutput which allows customizing the output and write results in your own format. At the same time the Cucumber has various extra pre-defined formats.

1. Conformity to Gherkin standards

|  |  |  |
| --- | --- | --- |
| Report type | Available in  JBehave | Available in  Cucumber |
| Feature | In JBehave the analog is Story. However, there's support of Gherkin style keywords which includes support of the Feature | + |
| Scenario | + | + |
| Given\When\Then | + | + |
| And\But | + | + |
| Scenario Outline | + | + |
| Examples | + | + |
| Background | No backgrounds no support of it | + |
| Tag | Tags have the same format but JBehave uses Meta keyword for that and that brings differences | + |
| Tables | + | + |

Differences:

1. **External Data**

It's needed when we want to customize the input for our test instructions. For this purpose JBehave has StoryLoader interface and we can implement loadStoryAsText method which can be defined the way we need and read data from whatever we can reach. I won't even dig too deep to get some example. Recently, I made an integration between JBehave and JIRA. What about Cucumber? Nothing at the moment.

1. **Inclusions**

The inclusions are needed to minimize code duplications. That's why they're are used in code, that's the purpose for scenarios as well. JBehave provides something similar to that named Given Stories it runs the stories specified in the referenced file prior to main story. It's not pure inclusion and mostly it can be omitted by groupping stories into one file. However, there're some cases where something like that is needed. Unfortunately Cucumber doesn't support this feature. Well, in some sense it's even good as such distributed scenarios are harder to trace and harder to maintain as it leads to external dependencies resolution. The funniest thing here is that such feature could be really needed for backgrounds as a lot of cases have some common pre-condition accross the features (usually it's login/logout operations). But Cucumber doesn't support inclusions and JBehave doesn't support backgrounds.

1. **Composite steps**

This feature is useful when we'd like to re-use steps so that we can call some test instructions by code bound. However, since all the steps are actually annotated methods this feature is not essential. The Composite steps in JBehave is built-in feature. It is also available in Cucumber but:

* These's no trace of it in JVM part
* There's no actual need of it in Java implementation
* If there's a real need of it it can be done using reflection

JBehave supports composite steps as full-scale feature. Cucumber-JVM can do the same but it's rather workaround than something fully supported.

1. **Backgrounds**

Backgrounds are supported by Cucumber however that's the weakest place of JBehave. There's no backgrounds there yet. That's one of the major things which don't conform the common Gherkin standards which are used accross many other BDD engines.

1. **Formatting flexibility**

JBehave doesn't ignores heading spaces before Given\When\Then statements. That's frustrating gap. However, if we dig deeper we can find that JBehave has StoryLoader interface and we can implement loadStoryAsText method to handle heading spaces and use it. Thus, we won't have such problems with formatting. Despite such flexibility isn't enabled by default it can be included using standard libraries.

+/- of JBehave:

* (+) - Very good documentation
* (+) - Pretty good HTML formatting of test results
* (-) - Only supports stories, not features

+/- of Cucumber-JVM:

* (+) - Supports features
* (-) - Doesn't support parallel JUnit tests. Will however work with parallel Maven 3 builds.
* (-) - Lacking documentation