

Gherkin Preprocessor Introduction

Preprocessing allows a different form of scenario writing and can reduce redundancy in feature files. The preprocessor can be found on <https://github.com/atdd-bdd/GherkinPreprocessor>

A different form of scenario writing

Gherkin allows you to write scenarios in different manners. Each form can document behavior more or less explicitly.

- a. For example, if a scenario is:

```
When value is over maximum  
Then report error
```

then the value of maximum is not explicit in the Gherkin.

- b. On the other hand, if the scenario is:

```
When value is over 200  
Then report error
```

then the meaning of 200 is not explicit in the Gherkin.

- c. With the preprocessor, a scenario can be written as:

```
#define MaximumValue 200  
When value is over MaximumValue  
Then report error
```

Both the meaning of 200 and its use is explicit. When these lines are preprocessed, they will appear in the output file as:

```
When value is over 200  
Then report error
```

This third form allows a triad to write a different style of feature file. Depending on the context, this form can be more maintainable.

How it Works

The preprocessor works separately from the feature runner (Cucumber/SpecFlow/etc.). The preprocessor is set up to transform a single file. The preprocessor is context insensitive, so it can work with other types of source files as well.

Computations in the preprocessor

In addition, the preprocessor allows for computations in #defines. For example:

```
#define MinimumValue 100
#define MaximumValue = MinimumValue + 200
```

Then MaximumValue is defined as 300. Local functions can also be called to create values used in the scenarios. For example:

```
#define TodaysDate = GET_TODAY()
#define ThirtyDaysLater = jTODAY_OFFSET_BY(30)
When order is placed TodaysDate
Then invoice is sent ThirtyDaysLater
```

The preprocessor has a few local functions. You can make up as many local functions as you wish. They receive an array of strings as the parameter and return a string.

Include Files and CSV Files

To reduce duplication, files can be included into feature files. For example, if “definitions.txt” had:

```
Background:
Given values are set
```

then a file that contains:

```
#include "definitions.txt"
Another line
```

Would be transformed into

```
Background:
Given values are set
Another line
```

The include feature also converts CSV files into tables. For example, if “values.txt” contained:

```
A,B,C,  
1,2,3
```

Then a file that included it:

```
#include "values.txt"
```

Would be transformed into:

```
|A|B|C|  
|1|2|3|
```

An Excel spreadsheet can be saved as a CSV file and that file used in a feature file. The CSV file would be checked in to the source code repository and treated just like other feature files.