How to describe a REST API using the Gherkin language for usage with the Gherkin2OAS tool

Dimanidis Tasos

anasdima@issel.ee.auth.gr

dhmtasos@gmail.com

skype: tasdhm.subscriptions@gmail.com

github: https://github.com/anasdima/gherkin2oas

youtube demo: https://www.youtube.com/watch?v=G5TNixy-dEc

Table of Contents

Table of Figures

1 Introduction

This document contains instructions on how to write Gherkin files in order to use them with the Gherkin2OAS tool. The instructions should be followed closely in order to generate a valid OpenAPI Specification.

Gherkin2OAS works as a simple converter. It takes as input a set of files written in Gherkin and produces an OpenAPI Specification from the natural language descriptions in them.

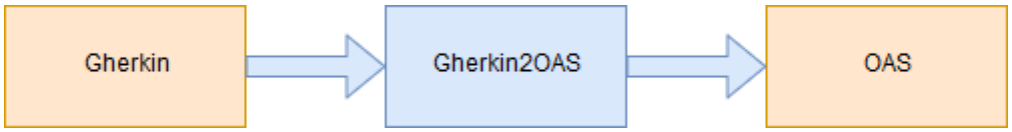

Figure 1 Gherkin2OAS

In order to perform a successful conversion, Gherkin2OAS expects a writing style inside the Gherkin files. In the following sections, we are going to describe what this writing style is and how we can use it to produce useful OpenAPI Specifications.

2 Gherkin

Gherkin[1] is a "*Business Readable, Domain Specific Language that lets you describe software's behaviour without detailing how that behaviour is implemented*". In Gherkin, software features are described in feature files. Inside A feature file looks like this:

```
Feature: Serve coffee
  Coffee should not be served until paid for
  Coffee should not be served until the button has been pressed
  If there is no coffee left then money should be refunded

Scenario: Buy last coffee
  Given there are 1 coffees left in the machine
  And I have deposited 1$
  When I press the coffee button
  Then I should be served a coffee
```

Figure 2 Gherkin

[1] https://github.com/cucumber/cucumber/wiki/Gherkin

As we can see, feature files contain scenarios that are described in a Given…When…Then sequence of steps. We are going to stick with this simple core concept of Gherkin for the rest of the document. For a detailed explanation, we suggest a read of Gherkin's Github page[2].

3 Gherkin writing style for Gherkin2OAS

In our case, we do not use the same rules as Gherkin does. That is because we use Gherkin to describe only REST APIs and not software in general. More importantly, we want these descriptions to be convertible by the Gherkin2OAS tool. So first, the file extension is different: instead of .feature we use .resource. Moreover, we adopt only the following keywords:

Feature
Background
Scenario
Given
When
Then
And
But

Figure 3 Gherkin2OAS understands only specific keywords

We do not adopt variable annotations in brackets: <variable name>.

Bear in mind that Gherkin2OAS takes each sentence and analyzes with Natural Language Processing. Therefore, the user of the tool must carefully construct sentences, so that Gherkin2OAS can understand them (more on that later). We should also mention that Gherkin2OAS ignores some types of sentences: sentences next to **Feature:** and **Scenario:** keywords, sentences below the **Feature** keyword and sentences inside comments.

```
Feature: Serve coffee
  Coffee should not be served until paid for
  Coffee should not be served until the button has been pressed
  If there is no coffee left then money should be refunded

Scenario: Buy last coffee
# This is a comment
  Given there are 1 coffees left in the machine
  And I have deposited 1$
  When I press the coffee button
  Then I should be served a coffee
```

Figure 4 Gherkin2OAS ignores highlighted sentences

Please note that Gherkin2OAS does not ignore comments next to sentences

[2] https://github.com/cucumber/cucumber/wiki/Gherkin

```
Scenario: Buy last coffee
  Given there are 1 coffees left in the machine
  And I have deposited 1$
  When I press the coffee button # Don't comment like this
  Then I should be served a coffee
```

Figure 5 Gherkin2OAS does not ignore comments next to sentences

You can use these ignored sections to describe better the REST API, without worrying about natural language analysis.

Now inside a file with a .resource extension, you can describe an API resource. A resource is any abstract organization of information that is relevant to a REST API. A resource can be a book, a list, an order, a schedule, a classroom, a person, a tweet, a video etc.

From a practical point of view, you can organize resources in folders and subfolders, according to your operating system's directory structure**. Additionally, Gherkin2OAS translates the name of the file (without the .resource part) into an API path**. Keep in mind that this is going to be the base of your path. For example, a book.resource file will produce a /book path base. Finally, filenames can have spaces, underscores and dashes. For example, an 'order list.resource' file name will be converted to a /order_list base path.

In the following sections, we are going to describe how to use each Gherkin keyword. **Keep in mind that you are describing a REST API and not a User Interface**.

3.1 Feature

The **Feature** keyword must be included in every file and it must be the first keyword. In reality, Gherkin2OAS does not extract any information from this keyword. However, the Gherkin parser used by Gherkin2OAS, requires this keyword to exist.

3.2 Background

You can optionally describe a **Background** section, below a Feature section and before any **Scenario** section. In the **Background** section, you can only use two types of **Given** sentences: A **Given** sentence that describes a role and a **Given** sentence that describes a relation with another resource.

A sentence that describes a role looks like this:

```
Feature: Manage administrators

    Background:
        Given that I am authorized as system
```

Figure 6 Gherkin2OAS Role description

Gherkin2OAS detects the role from the noun of the sentence. Therefore, you need to keep the sentence plain and clear. In this example, the role is "system". A role described in a **Background** section has authority on all of the scenarios described later in the file.

On the other hand, a sentence that describes a relation with another resource looks like this:

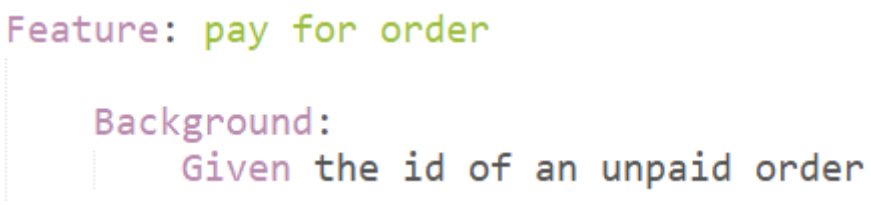

Figure 7 Gherkin2OAS relation with other resources

In this case, Gherkin2OAS knows that this is a sentence describing a relation, **because there is a name of another resource in the sentence**. This means that there is a file named 'order.resource'. In the OpenAPI Specification world, resource relation means path hierarchy. The resource described in the **Given** sentence is of higher hierarchy than the resource described in the file. In our case, the file is called 'payment.resource'. As a result, Gherkin2OAS will detect the path hierarchy /order/{order_id}/payment. Notice that Gherkin2OAS will also detect the id as a linking parameter, which is the only noun in the sentence besides the resource 'order'.

In conclusion, you can use the **Background** section for only two reasons: to describe a role and to describe a resource relation. If you mention the name of another resource in a **Given** sentence of a **Background** section, then you are automatically describing a resource relation. If you do not, then you are describing a role. Try to keep the sentences simple, short and clear. If you want to be verbose, use the comments.

3.3 Scenario

You must describe at least one **Scenario** section in the resource file. Each scenario must have one **When** and one **Then** step. This is the most important part of the resource file. You can use the **When**…**Then** sequence to describe an interaction with the API. In the **When** step, you can describe the request to the API, while in the **Then** step you can describe the response of the API to that request.

3.3.1 Scenario Given

You can use the **Given** sentence of a scenario to describe a role. That is the only usage of this keyword in a **Scenario**. Contrary to a **Given** sentence that describes a role in the **Background**, the role described here has authorization only to the operation(s) described in the current **Scenario**. The role here is again the noun of the sentence. That is why you need to keep your sentence simple and clear. **Given** as a word, does not have any other application in the REST world, since Roy Fielding's definition of REST dictates the stateless[3] nature of interactions. Therefore, describing something like "Given I am on state A", is not RESTful.

3.3.2 Scenario When

Let us look at the structure of a **When** sentence. A **When** sentence must always contain a verb. Gherkin2OAS translates that verb to one of these four HTTP verbs: POST, GET, PUT and DELETE. As a result, the verb must have a meaning that is similar to CREATE, READ, UPDATE and DELETE.

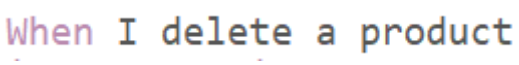

Figure 8 Gherkin2OAS mandatory verb in a When sentence

[3] https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm#sec_5_1_3

Gherkin2OAS decides for the meaning of the verb using word lists. There is a word list for each one of the four verbs:

CREATE/POST	`create, add, produce, make, put, write, pay, send, build, raise, develop, register, post, submit, apply, order, insert, comment, start`
READ/GET	`retrieve, check, choose, request, search, contact, get, take, see, ask, show, watch, read, open, reach, return, receive, view, load, review, select`
UPDATE/PUT	`perform, mark, evaluate, update, set, change, edit`
DELETE/DELETE	`delete, destroy, kill, remove, cancel, stop`

Figure 9 Gherkin2OAS mapping between HTTP verbs and meanings

If you use one of the verbs inside the word lists, then Gherkin2OAS will translate it to one of the HTTP verbs. Notice how there are no verbs that resemble UI usage such as click, drag, maximize, minimize, draw etc. That is because these UI verbs can be either POST or PUT or GET or DELETE.

Lastly, you can describe more than one verbs in the same **When** sentence (do not separate them with commas). If you choose to describe more than one, then all the verbs will have the same request and response details described in the scenario.

3.3.3 Scenario Parameters & Data Tables

You can describe parameters in a **When** or a **Then** part of a Scenario. The simplest way to describe a parameter is with a noun in the sentence. Gherkin2OAS ignores nouns that have the same name as the name of the resource of the current file.

```
Scenario: update unpaid order
    # Given I have ordered
    # And the order status is "unpaid"
    When I update an order by its id
```

Figure 10 Gherkin2OAS noun parameter (id) in a When sentence

Gherkin2OAS translates nouns to parameters of type string. If you want to control the data types of your parameters, you should use **Data Tables**. You can describe a Data Table below any sentence in a **When** or a **Then** step. When you describe a **Data Table** below a sentence, then Gherkin2OAS will not look into this sentence for noun parameters. As with Gherkin, **Data Table** columns are separated with the symbol '|' and **Data Table** rows with lines. At the top row (Figure 11) or at the most left

column (Figure 12), you can specify in that order (starting from top to bottom or from left to right respectively): a parameter name, a parameter example and parameter limits.

```
| parameter_A        | parameter_B        | parameter_C        | parameter_D        |
| example A          | example B          | example C          | example D          |
| parameter_A_limits | parameter_B_limits | parameter_C_limits | parameter_D_limits |
```

Figure 11 Gherkin2OAS Data Table horizonal orientation

```
| parameter_A | example A | parameter_A_limits |
| parameter_B | example B | parameter_B_limits |
| parameter_C | example C | parameter_C_limits |
| parameter_D | example D | parameter_D_limits |
```

Figure 12 Gherkin2OAS Data Table vertical orientation

The parameter name can have spaces, underscores and dashes. Moreover, Gherkin2OAS will use the parameter example to extract the data type of the parameter. Gherkin2OAS supports the OpenAPI Specification data types shown in

Figure 13.

String
Integer
Number
File
Date
Date-time
Boolean
Password
Array

Figure 13 Gherkin2OAS supported OpenAPI Specification data types

- To declare a string, use single or double quotation marks
- To declare an integer, the example must be an integer number
- To declare a number, the example must be a number with a decimal point
- To declare a file, just write the word “file” without quotation marks in the cell
- To declare date or date-time, you should know that python’s dateutil.parser is used. That means that if you specify a date example with the common format YYYY-MM-DD HH:MM:SS, then Gherkin2OAS will detect a date/date-time data type.
- To declare a Boolean, write “true” or “false” without quotation marks
- To declare an array, by enclosing a list of items in square bracket symbols: []. You can separate the items inside the array with spaces, commas, tabs and pipes |. The actual items can be of any of the supported data types. **However, all the items inside an array must be of the same data type**. In addition, as of this version, you cannot declare arrays inside arrays.
- To declare a password, just type a random series of asterisks * and no other symbols.

Lastly, you can declare parameter ranges. Parameter ranges are relevant for the following data types: String, Integer, Number, Password and array. Depending on your chosen **Data Table** orientation (see Figure 11 & Figure 12), you can specify parameter ranges in the third row or the third column of the **Data Table**. There are two ways to declare parameter ranges:

1. By specifying two numbers in the cell. Gherkin2OAS will detect those two numbers and it will use the smaller one as minimum and the larger one as maximum. You can write any number of words in the cell, **as long as you type in two and only two numbers**.

   By specifying a single number in the cell. However, since Gherkin2OAS cannot tell if the given number is a maximum or a minimum, you must declare it in the cell by writing one of the phrases shown in

2. Figure 14, along with the number.

Minimum	min	minimum	can't /cannot be less than	floor	
Maximum	max	maximum	can't/cannot be more than	ceiling	can't/ cannot be greater than

Figure 14 Gherkin2OAS single maximum or minimum in a Data Table

In Figure 15 you can see a complete example of a **Data Table**.

```
| string parameter  | "brown"                    | maximum length of 10          |
| integer_parameter | 1                          | between 1 and 10              |
| number-parameter  | 4.5                        | can't be less than 1.0        |
| document          | file                       |                               |
| today             | 2018-01-01                 |                               |
| timestamp         | 2018-01-01 00:00:01        |                               |
| arrived           | true                       |                               |
| password          | *******                    | at least 5 chars and at most 16 |
| products          | ['rock', 'paper', 'scissors'] | maximum of 10              |
```

Figure 15 Gherkin2OAS complete Data Table example

3.3.4 Scenario Parameter locations

Parameters in a **When** step can also have a location (path, query or body) depending on the HTTP verb that you are describing. The table in Figure 16 sums up the location of parameters, in relation to the four HTTP verbs. Gherkin2OAS will decide the location of the parameters, after detecting the HTTP verb in the **When** step.

	Single parameter	Multiple parameters	Parameters in Data Table
POST	body	body	body
GET	path	query	query

PUT	path	body	body
DELETE	path	query	query

Figure 16 Gherkin2OAS parameter location

3.3.5 Scenario Models

If you want your parameters to be organized in a model in the generated OpenAPI Specification, then you should end the sentence before a Data Table with a semicolon symbol ":", as shown in Figure 17. If you end your sentence with a semicolon symbol, Gherkin2OAS will try to find a noun in that sentence that will be the name of the model.

```
When I add a new product:
| name         | 'Chair'              |
| description  | 'Made in Stockholm'  |
| color        | 'Deep dark brown'    |
| category     | 'furniture'          |
| on sale      | false                |
| available in | 48                   |
| shipping     | 'worldwide'          |
| doa          | 7                    |
| id           | 45                   |
| cost         | 15000.1              |
```

Figure 17 Gherkin2OAS model example

Again, try to keep your sentence simple a clear, ideally with a single noun. You can use models in any sentence of a **When** or a **Then** step.

3.3.6 Scenario Then

Let us look at the structure of a **Then** sentence. A **Then** sentence describes the response to the API. It can contain one of the three following things:

1. A sentence with parameters or with a **Data Table**. The same rules apply here as with a **When** sentence. You can also use models.
2. A sentence that describe a status code. These sentences must have one and only one phrase in quotation marks (see **Error! Reference source not found.Error! Reference source not found.**). Gherkin2OAS will immediately regard a sentence with quotation marks as a status code sentence. It will then analyse the sentence for the status code, using word lists shown in Figure 19. If Gherkin2OAS finds any of the words in that table, then it will organize the entire response describe in the **Then** section under that status code.

```
Then I should see a message telling me "the product doesn't exist"
```

Figure 18 Gherkin2OAS status code in a Then sentence

200	success	ok	updated	deleted	retrieved	cancelled
201	created					
202	accepted					
400	failed	unsuccessful	wrong	mistake	error	not

401	unauthorized	not allowed	rejected	denied		
404	not found	doesn't/does not exist	unable to find	can't/cannot find	didn't/did not find	none was found

Figure 19 Gherkin2OAS status codes

3. A sentence that describes an operation to another resource (Figure 20). If the sentence contains a verb and the name of another, already documented, resource, then that sentence is describing an operation to another resource. This means that Gherkin2OAS will include in the response an HTTP link. This is in accordance with Roy Fielding's *Hypermedia as the Engine of Application State* concept that makes APIs completely REST. The verb must be one of the verbs shown in Figure 9. Additionally, the resource file of the other resource must exist and it must have a description of the operation in a scenario.

```
Then I should be prompted to submit a payment
```

Figure 20 Gherkin2OAS operation to another resource in a Then sentence

Notice how each sentence type has a unique writing style. As long as you follow this writing style, then Gherkin2OAS will be able to detect the correct type.

3.3.7 Scenario And, But

You can use **And** or **But** keywords below a **When**, a **Then** or another **And**/**But** sentence for additional information. The same rules apply as with a **When** or a **Then** sentence. However, you cannot describe an operation/API request/HTTP verb in an **And**/**But** sentence. You must only describe the verb of an API request in a When step. **And** and **But** keywords are treated as the exact same by Gherkin2OAS. You can see a complete example of a scenario in Figure 21.

```
Scenario: update unpaid order
    # Given I have ordered
    # And the order status is "unpaid"
    When I update an order by its id
    And I specify the following basket
        | product ids | [3433,1212,1276] |                  |
        | quantities  | [1,1,1,2]        | maximum of 100 |
    Then I should see a success message saying "Order updated"
    And I should be prompted to submit a payment
    And I have the option to review the order
    And I have the option to cancel the order
    And I have the option to update the order
```

Figure 21 Gherkin2OAS complete scenario with And sentences

3.3.8 Scenario Parameter requirements (BETA)

If you describe the same request in a **When** sentence in more than one **Scenario**, then Gherkin2OAS will try to detect which parameters are required for that HTTP request and which are not. If the parameters detected from the nouns or from the **Data Tables** exist in all the mentioned scenarios, then Gherkin2OAS will flag them as required. If not it will flag them is not required.

4 Contact

Please do not hesitate to contact me for any problems you may face, or for any questions you may have under any of the following info:

- email1: anasima@issel.ee.auth.gr
- email2: dhmtasos@gmail.com
- skype: tasdhm.subscriptions@gmail.com
- github: https://github.com/anasdima/gherkin2oas

5 Demo

You can view a usage demo of the whole system here:
https://www.youtube.com/watch?v=G5TNixy-dEc