How Important are Usage Examples in REST API Documentation?

S M Sohan, Frank Maurer*
* Dept. of Computer Science
University of Calgary
Canada
{smsohan, frank.maurer}@ucalgary.ca

Craig Anslow
School of Eng. and Computer Science
Victoria University of Wellington
New Zealand
craig@ecs.vuw.ac.nz

Martin Robillard
School of Computer Science
McGill University
Canada
martin@cs.mcgill.ca

***Abstract*—**

I. Introduction

II. Methodolody

A. Research Goals

This research is aimed at answering the following research questions:

- **RQ1**. What impact does usage examples have on REST API client developer productivity?
- **RQ2**. What problems API client developers face when using a REST API documentation that lacks usage examples?

.

B. Requirements

To answer the aforementioned research questions, the study has the following requirements:

- **R1. Realistic.** We had to choose an existing REST API that is currently used by API client developers. Selecting a mature REST API for this study reduces the possibility of an underdeveloped solution that may be found if a new or a seldom used API is selected. A familiar domain needed to be selected so that participants are able to relate to the API features without requiring upfront training. In addition to selecting the API, we had to select tasks that are related to the core features provided by the API to represent reality.
- **R2. Time bound.** The total time spent by each participant is required to be time bound to measure productivity. As a result, the study needs to be setup such that participants are able to focus on performing the tasks minimizing any overhead.
- **R3. Experienced developers.** Developers with prior experience on REST APIs need to be recruited as study participants to perform the study within a limited amount of time and in a realistic setup. Furthermore, to reduce a learning bias, only participants with no prior experince of using the WordPress REST API V2 are accepted for this study.

C. Study Object

We selected the WordPress REST API V2 for this study. WordPress is a blog-like open-source framework that is used by over 409 million people to visit 23.6 billion pages each month. The API allows programmatic access to list, create, update, and delete WordPress data such as blog posts, comments, users, images, tags, etc.

The WordPress REST API V2 has been published and maintained since May 2015. Before January 2017, the WordPress REST API V2 was distributed as a plug-in where WordPress users could optionally install the API component. The following statistics are for the plug-in install numbers between May 2015 and October 2016:

- Total installs: aprox. 248K installs of the plug-in.
- Average daily installs: approx. 500.

Starting January 2017, the WordPress REST API V2 is no longer required to be installed as a separate plug-in since it's pre-bundled with every WordPress install. As per the code repository on GitHub, there are a total of 99 and 46 contributors that had at least one commit to the code repository behind the API and and it's documentation, respectively. These properties satisfy R1, our requirements for being realistic.

By selecting an open-source project we are able to access the source-code to inspect the implementation and documentation technique of the WordPress REST API. The API implements a self-documenting feature where API developers expose API endpoints over HTTP OPTIONS verb to explain the API elements. To implement this feature, the API developers describe the API elements in the code. For example:

Listing 1. Example of self-documenting API Code

```
1  public function get_item_schema() {
2    $schema = array(
3      '$schema'      => 'http://json-schema.org/draft
           -04/schema#',
4      'title'        => $this->post_type,
5      'type'         => 'object',
6      /*
7       * Base properties for every Post.
8       */
9      'properties' => array(
10         'date'              => array(
11             'description' => __( "The date the
                   object was published, in the
                   site's timezone." ),
```

```
12                'type'          => 'string',
13                'format'        => 'date-time',
14                'context'       => array( 'view', 'edit
                      ', 'embed' ),
15            ),...
```

On Listing 1, line 4 specifies that this is a schema definition for the API element $Post$. Then, on Line 10, it defines $date$, one of the properties of $Post$, followed by a human readable description and and type information. Then, on line 14, the context of this property is mentioned as $view, edit, embed$, meaning that this property will be returned when the $Post$ object is returned, embedded, or can be used as an input for editing.

This self-documenting feature is leveraged to generate and publish the official API documentation as an HTML website. Figure 1 shows a screenshot of the published documentation for the $date$ attribute of the $Post$ API element[1].

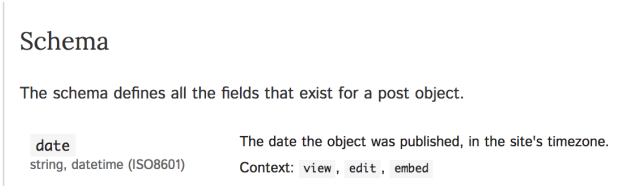

Fig. 1. Screenshot showing auto-generated documentation for Post/date

In addition to the auto-generated documentation of the the schema and API actions, custom content is added by the contributors to the API documentation project to provide prosaic overviews and usage examples.

D. Study Design

1) Tasks: For each task, define the task. Information that provided to the participants and what is required to perform it successfully.

2) Subject Selection:

3) Process:

4) Data Collection:

5) Data Analysis: Success, Failure, Partial:

III. Results

Observation and Implications For each task, describe the results. Provide examples of failed attempts with counts, and relate to the documentation sources to share an observation. Imply what needs to be done.

IV. Threats to Validity

V. Related Work

VI. Conclusion

Acknowledgment

[1]http://v2.wp-api.org/reference/posts/