My Product Manager said "write a documentation"

For many of us it's the most annoying sentence we could ever hear. Project is just finished, everything works great, but then comes the moment when you have to document all the stuff you coded for the past months. Can it get worse than that? If you said "no, writing a documentation is the worst thing ever" then definitely this article is a must-read for you!

Applying those 3 principles leads to success

Before we start doing some magic I have to tell you about 3 golden principles which every PHP programmer should follow:
This syntax is widely supported by many IDE's and tools - the only possible solution. In some cases you could use libraries which have defined custom tags (e.g. using Swagger about which I previously wrote an article).

```php
/**
 * Add two numbers.
 *
 * @author Wojciech Mleczek
 * @param int $x
 * @param int $y
 * @return int Result of x+y operation.
 */

function add($x, $y)
{
    return $x + $y;
}
```

2. Document as you go

Whenever you think "this information should be in the documentation" you have to put it there immediately, otherwise you will probably never do this. The bigger the backlog in the documentation, the greater the unwillingness to write it.

3. DRY (Don't Repeat Yourself)

Automation is the basis of success, who likes to repeat exactly the same thing over and over again? This is best seen in the example of the class diagram we could prepare at the beginning of the project. If we don't have a tool that will synchronize our existing class diagram with code changes, then the diagram sooner or later will be out of date. Few people like to apply the same changes in 2 places, and even if someone do it then he can just forget to copy something or make a mistake while moving changes.
This type of documentation can and should be fully automated. Nowadays there are plenty of tools to generate such documentation and their usage is as simple as possible. I'll show you how to use the FriendsOfPHP/sami generator which is used by Symfony and Laravel framework. If you're looking for alternatives to the Sami generator then you could check the phpDox or phpDocumenter.
### FileStore

**Traits**

- InteractsWithTime
- RetrievesMultipleKeys

**Properties**

- protected \$_files
- protected string \$_directory

**Methods**

- `int secondsUntilDate(int $time)`: Get the number of seconds until the given DateTime.
- `int availableAt(int $time)`: Get the "available at" UNIX time stamp.
- `int parseDateTime(int $time)`: If the given value is an interval, convert it to a DateTime instance.

---

**Synopsis**

```php
class Parser {
    // members
    protected \$_factory;
    protected \$_current;
    protected array \$_aliasMap;

    // methods
    public void __construct()
    public void parse()
    protected void prepare()
    protected void startParser()
}
```

---

**Coverage**

<table>
<thead>
<tr>
<th>Methods</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines</td>
<td>92.31%</td>
</tr>
</tbody>
</table>
Ok, it's time to do some work in practise. Our example will be based on the Deplink project, but you could also choose almost any project from the GitHub trending PHP repositories or your project on which you're currently working on.

Open command line and create new directory dedicated for this sample:

```
mkdir sami-sample && cd sami-sample
```

Clone your project (in my case it's an example project from GitHub repository):

```
git clone https://github.com/deplink/deplink
```
curl -O http://get.sensiolabs.org/sami.phar

Write basic Sami configuration in the `sami-sample/config.php` file:

```php
<?php

// Point where Sami should search for PHP source code.
$srcDir = 'deplink/src';

return new Sami\Sami($srcDir);

```

Now, let's generate documentation from our source code:

```
php sami.phar update config.php
```

Done. To view it open `build/index.html` and browse objects in the navigation pane on the left. Sample class view:
Generating wiki documentation

In my opinion this kind of documentation is the most important for most of projects (not only PHP related), but can be only partially automated (our task is to write the content). Let's have a look at sample documentations online:

Vue Documentation
Introduction

What is Vue.js?

Vue (pronounced /vjuː/, like view) is a progressive framework for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is very easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries.

If you are an experienced frontend developer and want to know how Vue compares to other libraries/Frameworks, check out the Comparison with Other Frameworks.

Getting Started

The official guide assumes intermediate level knowledge of HTML, CSS, and JavaScript. If you are totally new to frontend development, it might not be the best idea to jump right into a framework as your first step - grasp the basics then come back! Prior experience with other frameworks helps, but is not required.

Windows 10 IoT Core Official Website
Do you see any similarities? Both pages are a markdown file on GitHub as evidenced by the message at the bottom of the page:

Microsoft:

| Edit this page on GitHub |

Vue:

| Caught a mistake or want to contribute to the documentation? Edit this page on Github! |

Using this solution you can make that whenever you edit markdown file and push a commit then a new documentation will be generated and uploaded to your web server. It's an amazing possibility to eliminate repeatable work, reduce human errors and gain time for programmers to work on the other stuff.
You could ask “how to start”? Possibilities are many: Jekyll, Hugo, Hexo and solutions like GitHub Pages or GitLab Pages (available in GitLab Community Edition). We’ll focus on Hexo.

Create and go to the new working directory:

```bash
mkdir hexo-sample && cd hexo-sample
```

Install Hexo CLI globally (Node.js with NPM required):

```bash
npm install hexo-cli -g
```

Init new Hexo project in current directory:

```bash
hexo init .
```

Run server and go to `http://localhost:4000/` to see your new empty project:

```bash
hexo server
```

Or simply build project without running server. This command generates HTML files which you can upload to your server:

```bash
hexo generate
```

After every modification to markdown files you can freely use `generate` command to update the HTML files.
The page content you see is located in a `hexo-sample/source/_posts/hello-world.md` file. Content of this file is in markdown with frontmatter header. Markdown is a simplified language that enables creating formatted HTML documents using plain text. Try to edit it, markdown syntax and front matter basic information could be helpful to get started. Have fun while exploring new possibilities of the Hexo static HTML generator!

You can be a bit confused that this is an example of the blog. Don't be afraid, it's possible to change it to standard wiki articles grouped in categories, but you have to do it manually by modifying theme. Once you prepare a theme you can reuse it in any project and focus only on content.

### Generating documentation of database

Database documentation is often missing for PHP application. The reasons could be a few:

- we've documented ORM models,
- we've documented PHP migrations,
- we think that columns are self-describing,
- or simply lack of time
I won't tell you now how much documentation of the database is needed. I assume that if you're reading an article about documenting software projects, then you would like to know how to do it without all this unnecessary introductions.

Download, install and run Dataedo. If you have any issues with one of the steps then please click on it and follow the instructions contained therein. You shouldn't have any problems to download and install Dataedo as well as creating a first file repository (which is simply a file that will be used to store documentation for your database).

For us, the "3. Connect To Your Database" and "7. Export To HTML (Pro feature)" section in the "How to Document MySQL Database in 5 Minutes with Dataedo [Free Tool]" article will be important. Follow the steps in both of these sections to produce HTML like below one:

Open sample documentation

Repository

1. AdventureWorks

Want to get such documentation for your database?

No problem, create it in fast and easy with Dataedo.

Try It Free Now   Learn more   Close
Maybe at least I convinced you that documenting code doesn't have to be so terrible. I keep my fingers crossed that next time it's up to you to surprise the boss saying that the documentation is ready. Good luck!

There are no comments. Click here to write the first comment.

Recommendations

PRODUCTS AND NEWS

How to document REST API project written in PHP using Swagger and Dataedo

How to Document Stored Procedures and Functions in Database
Extended Properties for Database Documentation - Good or Bad?

Create a Data Dictionary

Create a Data Dictionary with Dataedo

Who do You Document Databases For

Captain Obvious' Guide to Column Descriptions - Data Dictionary Best Practices
Subscribe our blog and knowledge base

Get new articles and tips on modeling, designing and documenting databases to your inbox.

Subscribe