Table of Contents

[GitHub 1](#_Toc509830442)

[Composer 2](#_Toc509830443)

[Propel 3](#_Toc509830444)

[Slim 4](#_Toc509830445)

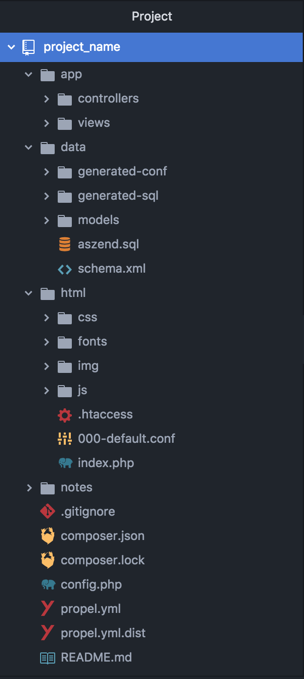
[Works Cited 5](#_Toc509830446)

# Directories

**Overview**

The structure which all projects must follow for readability and consistency.

**Structure and explanation**



# GitHub

**Overview**

GitHub is a web-based hosting service for version control using git. It is mostly used for computer code. It offers all of the distributed version control and source code management functionality of Git as well as adding its own features [1].

**Installation**

To install git on your machine follow the instructions at <https://git-scm.com/downloads>. Once it’s installed, run the command *git* in the terminal, and you should receive instructions on how to use the commands.

**Important git commands**

|  |  |
| --- | --- |
| Command | Explanation |
| git init . | Initialize a Git repository in the current directory |
| git clone repo\_url.git | Create a local copy of a remote repository |
| git status | Check status since last pull |
| git add [file] | Add a file to the staging area |
| git add –A | Add all new and changed files to the staging area |
| git commit –m “commit message” | Commit your messages |
| git push -u origin [branch name] | Push a branch to your remote repository (almost always master) and remember the branch |
| git push | Push changes to remote repository of remembered branch |
| git pull | Update local repository to the newest commit |
| git remote add origin repo\_url.git | Add a remote repository |
| git rm –r [file] | Remove a file or directory |

**Gitignore file**

Create a .gitignore file in the root of the local repository. Git will look at the contents of this file and decide what items to ignore, such as large files not suitable for pushing. For example, a .gitignore file that excludes the large vendor folder from being pushed will look like this:



Make sure to **never push vendor**, it will cause headaches and reduce efficiency.

**Github notes**

These commands will be mostly used while working in a project, if you encounter a problem or would like to learn more, go to <https://try.github.io/>. Unlimited private remote repositories are available through a fee, or by registering as a student, which is recommended. Merge conflicts can be a pain to deal with, and therefore should be resolved with atom/sublime or be avoided completely. Make sure to **always pull before you push**, and therefore obligating you to commit any changes.

# Composer

**Overview**

Composer is an application-level package manager for the PHP programming language that provides a standard format for managing dependencies of PHP software and required libraries [2].

**Installation**

Composer can be installed following the guide in <https://getcomposer.org/doc/00-intro.md>; make sure it’s a global installation. Once it’s installed, run *composer* in the terminal, and you should receive the composer screen. In case of an error, make sure you follow the steps correctly with admin privileges (e.g. using sudo on a Unix machine).

**Important composer commands**

|  |  |
| --- | --- |
| Command | Explanation |
| composer require package/library | Will update composer.json with the new dependency or create a new composer.json and composer.lock if they don’t exist |
| composer install | Installs the vendor packages according to composer.lock (or creates composer.lock file if not present) |
| composer update | Will regenerate composer.lock with the new composer.json dependencies and versions, no matter if composer.lock exists or not |
| composer dump-autoload -o | Regenerates the list of all classes that need to be included in the project (vendor/composer/autoload\_classmap.php). |

**Example composer.json file**



*Explanation of composer.json:*

**require**: {…} lists the dependencies required for the project (propel, slim, etc...)

**autoload**: {…} lists extras to be included when *composer dump-autoload –o* is ran. classmap allows directories to be included in the autoload process. The psr-4 autoload is used to define the mapping from namespaces to directories. The example filename would be app/hello.php containing an App\Hello class.

**Composer notes**

Make sure that any of the composer commands are ran in the same directory as the composer.json, or the commands will fail. Composer will download the dependencies into a vendor/ directory, which it will create if it doesn’t exist. Every dependency will have a directory, for example, propel will be inside vendor/propel. *composer dump-autoload –o* will **not** work in the previously shown composer.json example if data/models and app/ directories don’t exist. If you want to test it, simply remove everything except require: {…}.

# Propel

**Overview**

Propel is a free, open-source object-relational mapping toolkit written in PHP. It is also an integral part of the PHP framework Symfony and was the default ORM up to, and including version 1.2 [3].

**Installation**

Require it in composer.json:



Run composer update to download propel and its dependencies. The dependencies will be downloaded into the vendor/ directory; if you see a propel/ directory in vendor/ you’re set.

**Important Propel commands**

|  |  |
| --- | --- |
| Command | Explanation |
| vendor/bin/propel init | Initialize propel installation |
| vendor/bin/propel reverse | Create a new schema.xml with the updated database information (inside generated-reversed-database/ directory) |
| vendor/bin/propel model:build | Rebuild the models according to schema.xml |

**Propel notes**

When running *vendor/bin/propel init* make sure you have your apache server on, or propel won’t be able to establish a connection with the database; an example run of *vendor/bin/propel init* can be found at <https://goo.gl/ej4nt5>. Once you initialize Propel, two new folders will be created in the current directory (generated-conf/ and generated-sql/), move them into the data/ directory to keep the project consistent.

# Slim

**Overview**

Slim is a PHP micro framework that helps you quickly write simple yet powerful web applications and APIs. At its core, Slim is a dispatcher that receives an HTTP request, invokes an appropriate callback routine, and returns an HTTP response. That’s it [4].

**Installation**

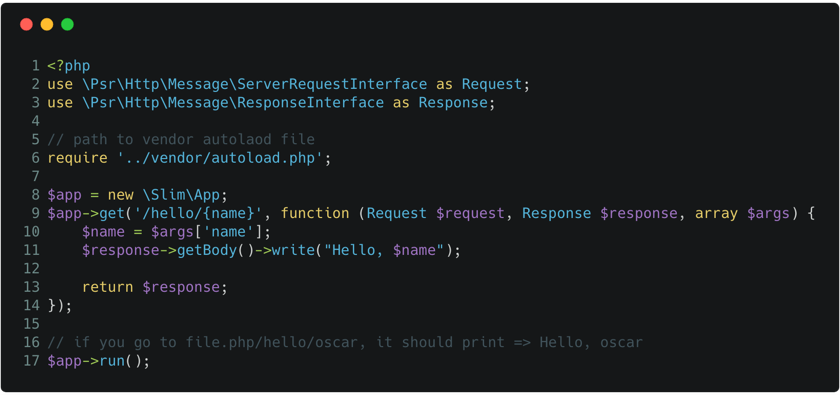
Require it in composer.json:



Run composer update to download slim and its dependencies. The dependencies will be downloaded into the vendor/ directory; if you see a slim/ directory in vendor/ you’re set.

**Getting started**

Create an index.php file and type the following:



# Works Cited

[1] Wikipedia. March 23, 2018. Github. <https://en.wikipedia.org/wiki/Composer_(software)>

[2] Wikipedia. March 24, 2018. Composer. <https://en.wikipedia.org/wiki/Composer_(software)>

[3] Wikipedia. March 12, 2018. Propel (PHP). <https://en.wikipedia.org/wiki/Propel_(PHP)>

[4] Slim Framework. March 26, 2018. Documentation. <https://www.slimframework.com/docs/>