# Contributing

## Introduction to the project

\*\*tesseract-ocr-for-php\*\* is just a convenient way to generate and execute

`tesseract` commands inside PHP scripts, offering some [syntactic sugar][] for

lengthy options and accounting for differences required by different Tesseract

versions.

In a nutshell, it simply gathers the options set by the user, builds a valid

`tesseract` command as a string, runs it with [exec][] and returns the command

output.

I plan to eventually interact with Tesseract via its [baseapi][] instead of

relying on its command-line through [exec][], but for now I'd say that the

current implementation is good enough, serves well its purpose and is easy to

maintain.

## How the project is organized

src/

??? TesseractOCR.php

??? Command.php

??? Option.php

###### `src/TesseractOCR.php`

This is the only class exposed for consumption, the project's public [API][].

The user sets Tesseract options by invoking methods to an instance of this

class. Those method calls are collected as command options.

###### `src/Command.php`

This class is responsible for building the `tesseract` command, with all flags

and options on the expected order.

###### `src/Option.php`

Each method of this class represents a corresponding command-line option

offered by the `tesseract` binary.

## Tests

Historically I've been very strict about pull requests without tests, but

some time ago I decided to stop using [PHPUnit][] for testing this project

(you can find the reasons on the message of commit [23336c6][]).

So it no longer makes no sense to have such strict policy. And more than that,

enforcing tests was discouraging novice programmers from contributing.

That means you are free to place pull requests without tests, and I should just

be grateful that someone is willing to help.

But if you are comfortable writing tests, I definitely encourage you to write

some!

If that is the that case, here is an overview of the kinds of tests currently

present on the project:

### Unit

It reflects the same structure of `src`, having one test class for each

corresponding `src` class:

tests/Unit/

??? TesseractOCRTest.php

??? CommandTest.php

??? OptionTest.php

Important to highlight that Unit Tests will never invoke the real `tesseract`

binary.

To run them:

$ php tests/run.php unit

### End To End

As the name implies, this kind of test will run the whole thing, using real

images and invoking the `tesseract` binary. Nothing is stubbed nor mocked.

The purpose of those tests is to simulate exactly how a user of this project

would interact with it.

To run them:

$ php tests/run.php e2e

## Thank you very much for taking the time to contribute!

:green\_heart::yellow\_heart::heart::purple\_heart::blue\_heart::black\_heart:

[syntactic sugar]: https://en.wikipedia.org/wiki/Syntactic\_sugar

[exec]: http://php.net/exec

[baseapi]: https://github.com/tesseract-ocr/tesseract/blob/master/api/baseapi.h

[API]: https://github.com/thiagoalessio/tesseract-ocr-for-php#api

[PHPUnit]: https://phpunit.de

[23336c6]: https://github.com/thiagoalessio/tesseract-ocr-for-php/commit/23336c658f162a73cf961fecf3cb42f6ee1fdf6e