BUILDING EXTRAORDINARY PACKAGES

THE STORY OF THE PHP LEAGUE

Phil Sturgeon

Framework Interoperability Advocate

THE LEAGUE OF EXTRAORDINARY PACKAGES

NAME	DESCRIPTION	PROJECT LEAD
BooBoo	Because everybody makes mistakes	Brandon Savage
CLImate	Terminal output made easy	Joe Tannenbaum
Color Extractor	Extract colors from an image	Mathieu Lechat
CommonMark	Markdown parser for PHP based on the CommonMark spec	Colin O'Dell
Container	Fast and intuitive dependency injection container	Phil Bennett
CSV	CSV data manipulation made easy	Ignace Nyamagana Butera
Event	Event package for your app and domain	Frank de Jonge
Factory Muffin	Enables the rapid creation of objects for testing	Scott Robertson
Flysystem	Abstraction for local and remote filesystems	Frank de Jonge
Fractal	Output complex, flexible, RESTful data structures	Phil Sturgeon
Geotools	Perform geo-related tasks	Antoine Corcy
Glide	HTTP based image manipulations	Jonathan Reinink
Monga	Simple and swift MongoDB abstraction	Bryan Crowe
OAuth 1 Client	Integrate with OAuth 1.0 providers	Ben Corlett
OAuth 2 Server	Build an OAuth 2.0 server	Alex Bilbie
OAuth 2 Client	Integrate with OAuth 2.0 providers	Ben Ramsey
Omnipay	Multi-gateway payment processing library	Kayla Daniels
Period	Time range API for PHP	Ignace Nyamagana Butera
Plates	Native PHP template system	Jonathan Reinink
Route	Router and Dispatcher built on FastRoute	Phil Bennett
Shunt	Execute SSH commands on remote machines	Taufan Aditya
Squery	PHP wrapper for osquery	Christopher Pitt
Statsd	Library for working with StatsD	Marc Qualie
Tactician	A simple, flexible command bus	Ross Tuck
URL	URL manipulation made easy	Ignace Nyamagana Butera

1. Use `League` as the PSR-4 autoloader namespace. Shove code in a `src` folder.
2. Adhere to PSR-2 as the coding style guide.
3. List on Packagist with `league` as the vendor namespace.
4. Write unit tests. Aim for at least 80% coverage in version 1.
5. DocBlock all the things.
6. Use Semantic Versioning to manage version numbers.
7. Keep a Changelog.
8. Use Travis-CI to automatically check coding standards and run tests.
9. Have an extensive README.
10. Exclude non-essential files in .gitattributes.

How to successfully release an open-source PHP package

(and become a better developer for it)

The goods

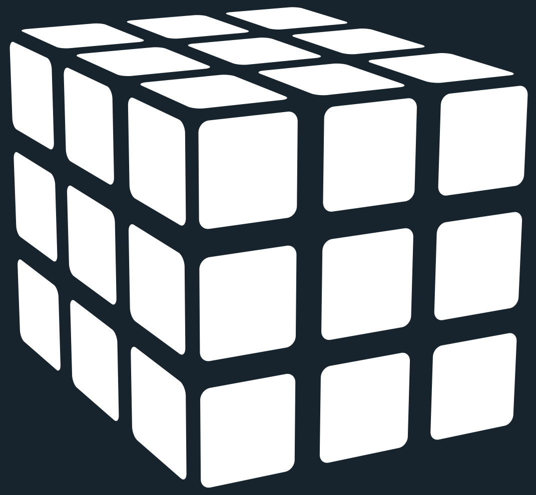
1. **Make**
2. **Market**
3. **Maintain**

Things to consider before you start

Why you should and why you shouldn’t.

Does it exist already?

Don’t clone, send pull requests instead.



Share your unique way of solving a problem

Push the status quo.

Do you have the time?

Releasing open source code requires a time commitment.

You will meet people

Yay for nerd friends!

PHP-FIG

Code Coverage

Changelogs

Issues

TDD

Packagist

Licensing

Rebasing

PSR

Code Quality

Scrutinizer CI

Shields.io

DocBlocks

Jekyll

Semantic Versioning

Pull Requests

Travis

Milestones

You will learn, a lot

GIT

Contributing an open source package will push you as a developer.

Releases

Code Sniffer

Cohesion

Coupling

GitHub

Testing

Dependency Injection

Coding Standards

Composer

1. Make

Design an API developers will want to use

The cornerstone to a successful package.

Send an email with Swift

```php
// Create the transport
$transport = Swift_SmtpTransport::newInstance('smtp.example.org', 25);
$transport->setUsername('your username');
$transport->setPassword('your password');

// Create the email
$message = Swift_Message::newInstance();
$message->setSubject('Your subject');
$message->setFrom(array('john@doe.com' => 'John Doe'));
$message->setTo(array('foo@example.com'));
$message->setBody('Here is the message itself');
$message->attach(Swift_Attachment::fromPath('document.pdf'));

// Send the email
$mailer = Swift_Mailer::newInstance($transport);
$result = $mailer->send($message);
```

Send an email with Laravel

```
Mail::send('emails.welcome', $data, function ($message) {

    $message->subject('Welcome!')
            ->from('john@doe.com', 'John Doe')
            ->to('foo@example.com')
            ->attach('document.pdf');
});
```

Name things right

It’s easy, like cache validation.

Whoops

```php
// Current library
$whoops = new Whoops\Run;
$whoops->pushHandler(new Whoops\Handler\PrettyPageHandler);
$whoops->register();


// Better class name
$whoops = new Whoops\ErrHandler;
$whoops->pushHandler(new Whoops\Handler\PrettyPageHandler);
$whoops->register();


// Better example variable
$errHandler = new Whoops\ErrHandler;
$errHandler->pushHandler(new Whoops\Handler\PrettyPageHandler);
$errHandler->register();
```

Have a clear focus

Pull in other libraries when needed.

Utilize common design patterns

Techniques like dependency injection make your library easier use, maintain, read and test.

Break apart large classes

Create more focused classes, and more of them.

Framework agnostic

Don’t limit yourself to just one framework.

What versions of PHP should I support?

Is PHP 5.3 worth the effort?

Source code on GitHub

Sorry Bitbucket, Google Code & SourceForge.

Write tests

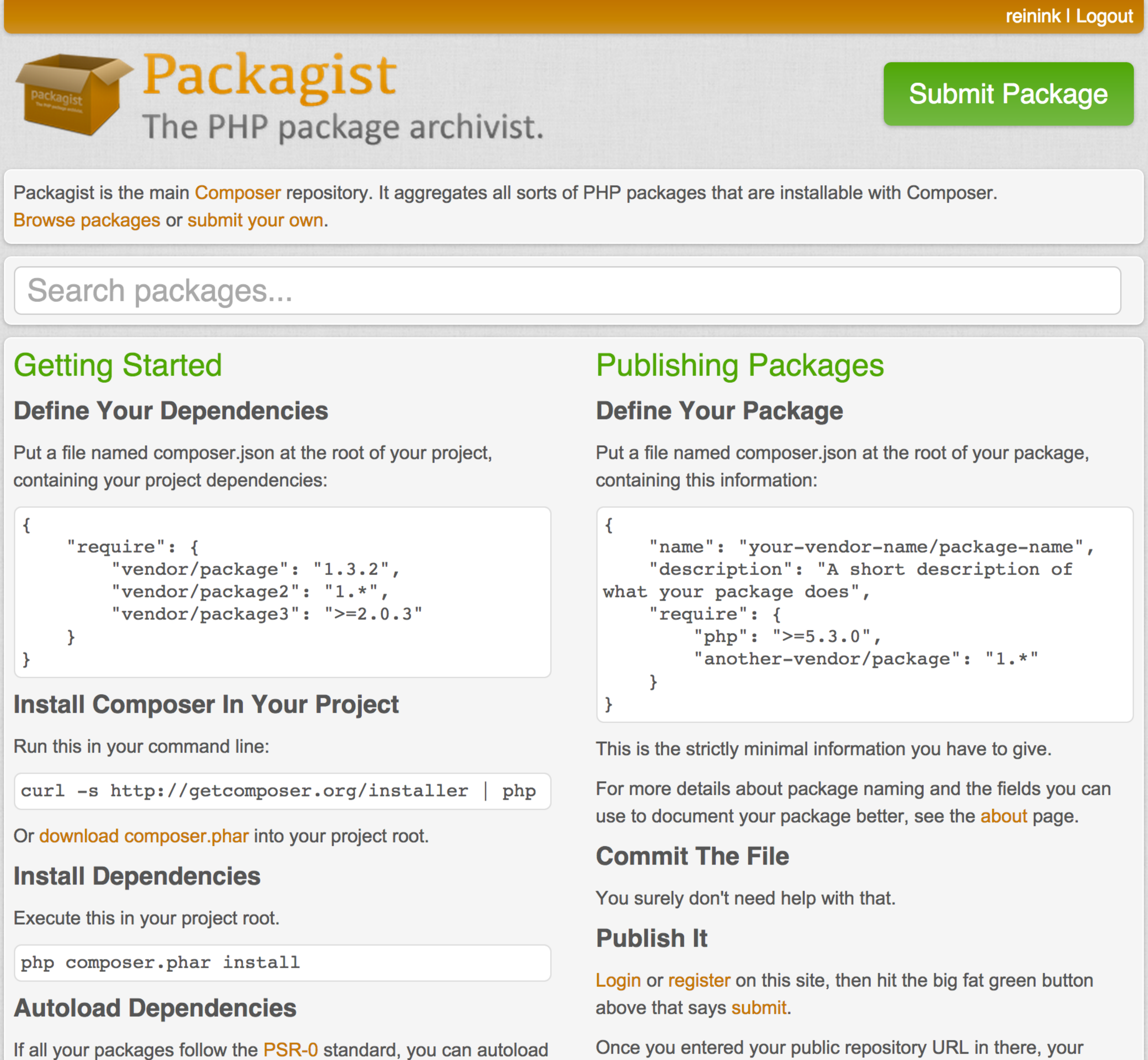
Automated tests allow you to make stress-free changes.

Composer & Packagist

The primary delivery mechanism for your library.

composer.json

```
{
    "name": "league/fractal",
    "description": "Handle the output of complex data structures ready
for API output.",
    "homepage": "http://fractal.thephpleague.com/",
    "license": "MIT",
    "author": [{
        "name": “Phil Sturgeon”,
        "email": “me@philsturgeon.uk"
    }],
    "autoload": {
        "psr-4": {
            "League\\Fractal\\": "src"
        }
    }
}
```



Packagist is the main Composer repository. It aggregates all sorts of PHP packages that are installable with Composer.
Browse packages or submit your own.

Getting Started

Define Your Dependencies

Put a file named composer.json at the root of your project, containing your project dependencies:

```
{
    "require": {
        "vendor/package": "1.3.2",
        "vendor/package2": "1.*",
        "vendor/package3": ">=2.0.3"
    }
}
```

Install Composer In Your Project

Run this in your command line:

```
curl -s http://getcomposer.org/installer | php
```

Or download composer.phar into your project root.

Install Dependencies

Execute this in your project root.

```
php composer.phar install
```

Autoload Dependencies

If all your packages follow the PSR-0 standard, you can autoload

Publishing Packages

Define Your Package

Put a file named composer.json at the root of your package, containing this information:

```
{
    "name": "your-vendor-name/package-name",
    "description": "A short description of
what your package does",
    "require": {
        "php": ">=5.3.0",
        "another-vendor/package": "1.*"
    }
}
```

This is the strictly minimal information you have to give.

For more details about package naming and the fields you can use to document your package better, see the about page.

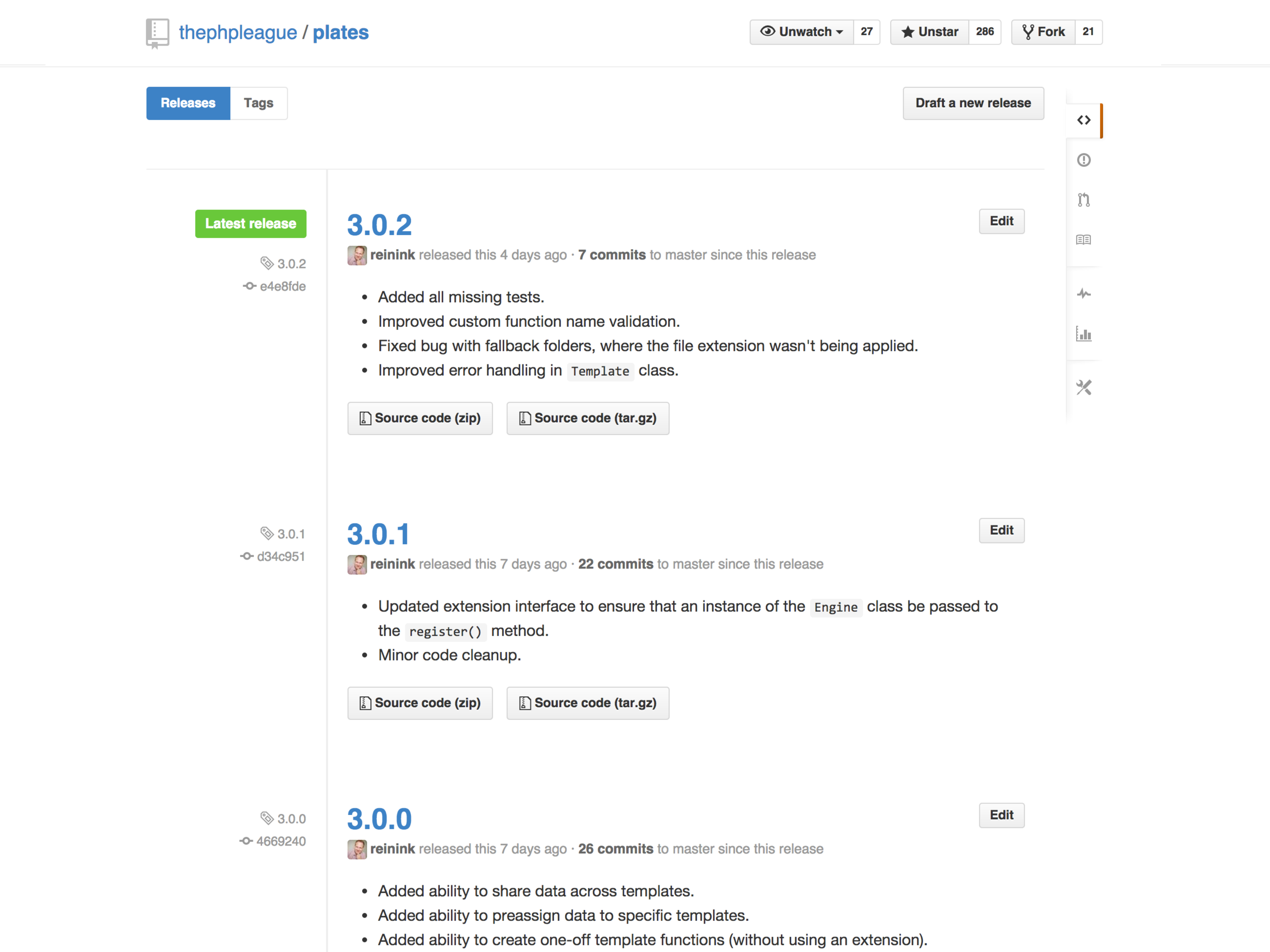
Commit The File

You surely don't need help with that.

Publish It

Login or register on this site, then hit the big fat green button above that says submit.

Once you entered your public repository URL in there, your



thephpleague / plates

Unwatch 27 · Unstar 286 · Fork 21

Releases · Tags · Draft a new release

Latest release

3.0.2 · e4e8fde

3.0.2

Edit

reinink released this 4 days ago · 7 commits to master since this release

- Added all missing tests.
- Improved custom function name validation.
- Fixed bug with fallback folders, where the file extension wasn't being applied.
- Improved error handling in `Template` class.

Source code (zip) · Source code (tar.gz)

3.0.1 · d34c951

3.0.1

Edit

reinink released this 7 days ago · 22 commits to master since this release

- Updated extension interface to ensure that an instance of the `Engine` class be passed to the `register()` method.
- Minor code cleanup.

Source code (zip) · Source code (tar.gz)

3.0.0 · 4669240

3.0.0

Edit

reinink released this 7 days ago · 26 commits to master since this release

- Added ability to share data across templates.
- Added ability to preassign data to specific templates.
- Added ability to create one-off template functions (without using an extension).

.gitattributes

```
/tests                  export-ignore
/.gitattributes         export-ignore
/.gitignore             export-ignore
/.scrutinizer.yml       export-ignore
/.travis.yml            export-ignore
/phpunit.xml            export-ignore
```

MAJOR.MINOR.PATCH

BREAKING.NEW-FEATURES.BUG-FIXES

Semantic Versioning

Allows developers to upgrade versions safely.

Coding Standards

Adhere to PSR-2 as the coding style guide.

DocBlocks

Allows for automated API documentation.

Continuous Integration

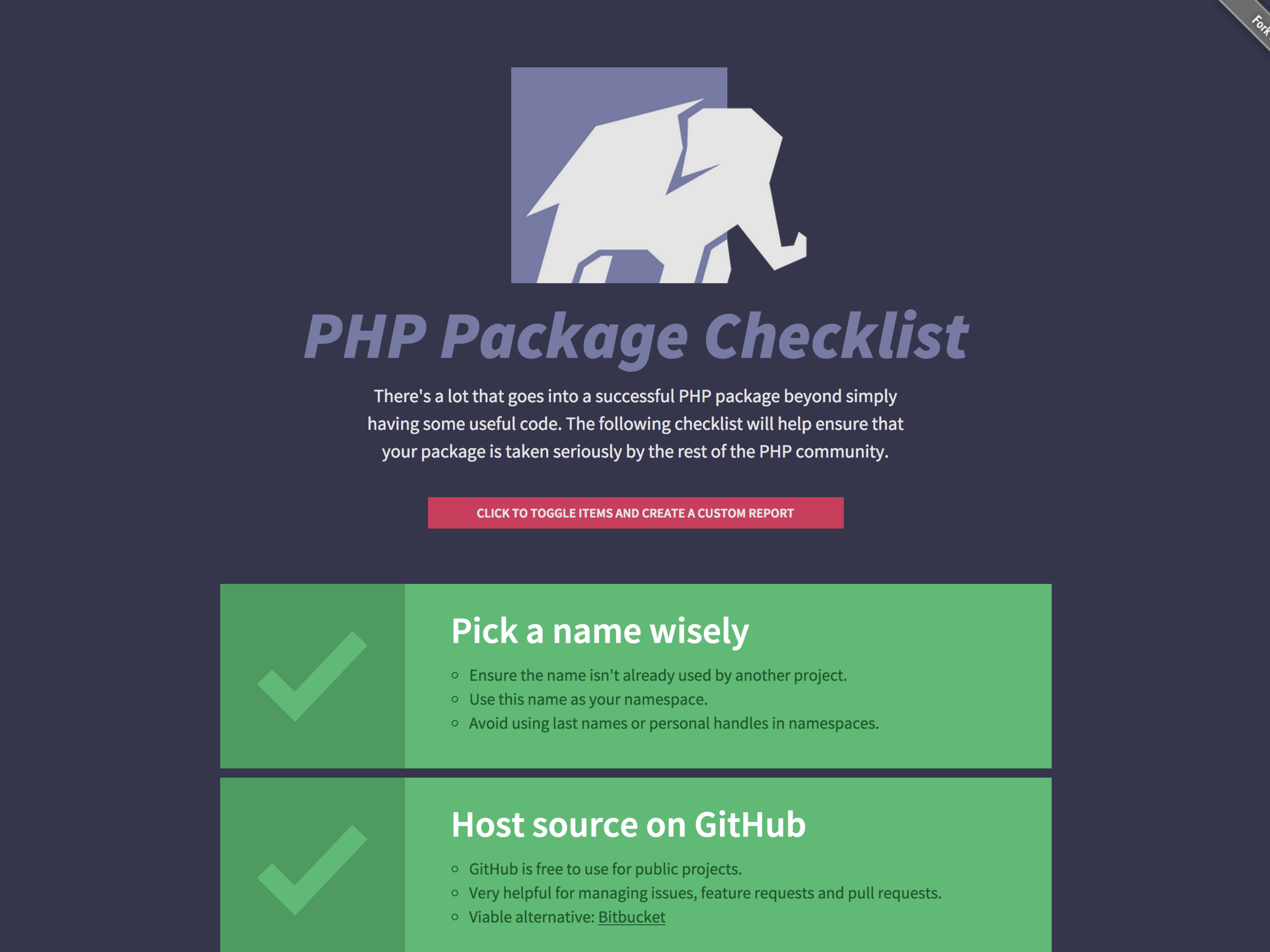
Automate tests, PSR compliance checks, code coverage analysis & more.

Have a license

An important step to protect your hard work.

Contributor instructions

Help them, help you!



PHP Package Checklist

There's a lot that goes into a successful PHP package beyond simply having some useful code. The following checklist will help ensure that your package is taken seriously by the rest of the PHP community.

✓

Pick a name wisely

- Ensure the name isn't already used by another project.
- Use this name as your namespace.
- Avoid using last names or personal handles in namespaces.

✓

Host source on GitHub

- GitHub is free to use for public projects.
- Very helpful for managing issues, feature requests and pull requests.
- Viable alternative: Bitbucket

2. Market

Choosing a name

Memorable, short and cool (without being too hipster).

The documentation

Your most important marketing tool.

Documentation myth #1

“Read the code” is an acceptable answer
to“Where are the docs?”

Documentation myth #2

“Auto-generated docs are good enough”

Documentation myth #3

“All you need a README file”

Documentation myth #4

“Documentation is easy.”

Documentation “must-haves”

How to do documentation right!

The elevator speech

What it is and why it matters, in 160 characters or less.

<?

The simple example

Show me the code!!!

Installation instructions

Make it easy for someone to get started.

Via Composer

```
$ composer require league/fractal
```

Keep a changelog

Include upgrade instructions for
backwards breaking changes.

Links to source & author

This is open source after all, make yourself available!

Badges!

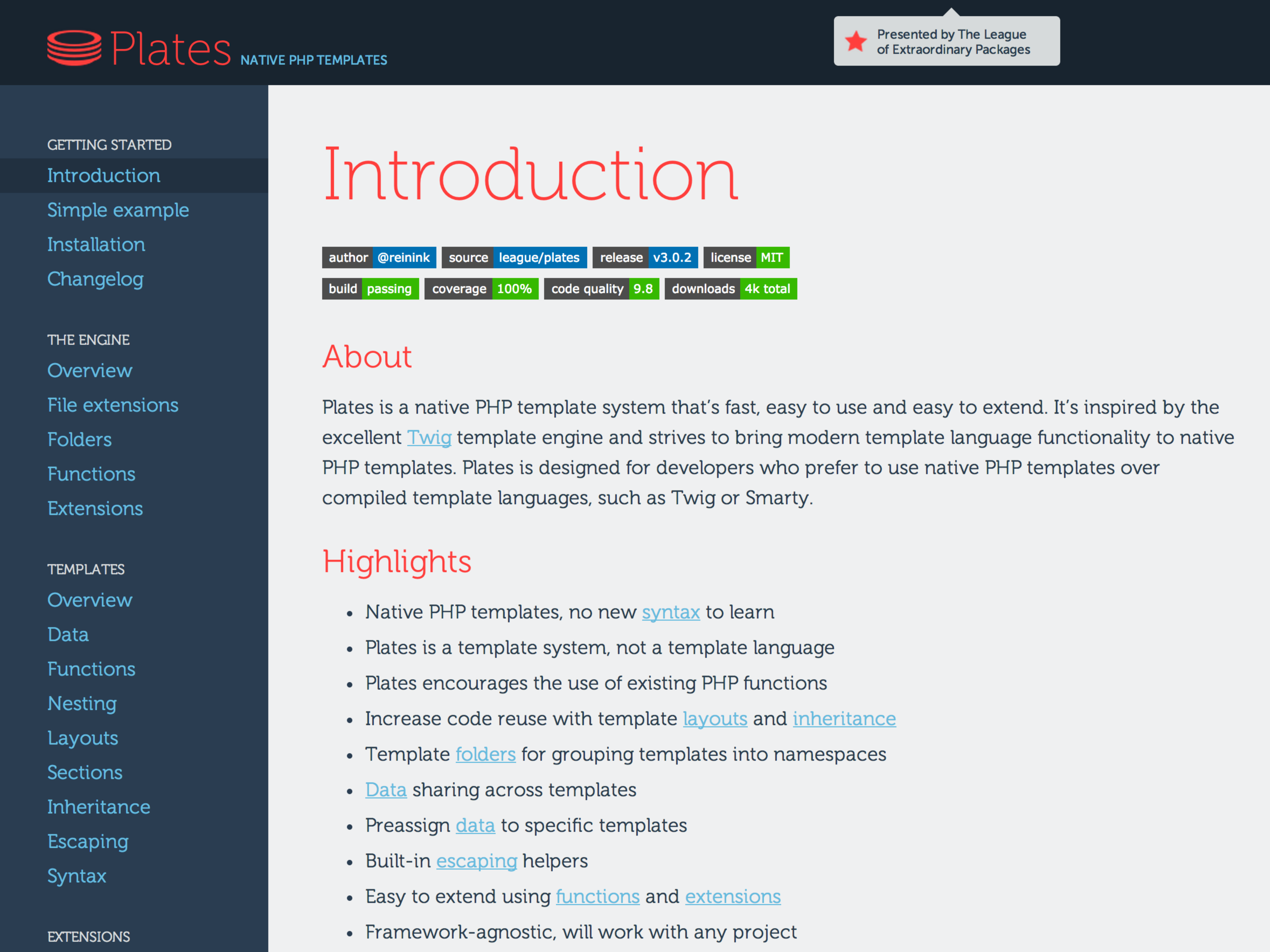
Badges help full in real-time information about your project.

source league/plates

build passing   coverage 100%

author @reinink   downloads 4k total

code quality 9.8   license MIT   release v3.0.2



Plates NATIVE PHP TEMPLATES

Presented by The League of Extraordinary Packages

GETTING STARTED
Introduction
Simple example
Installation
Changelog

THE ENGINE
Overview
File extensions
Folders
Functions
Extensions

TEMPLATES
Overview
Data
Functions
Nesting
Layouts
Sections
Inheritance
Escaping
Syntax

EXTENSIONS

Introduction

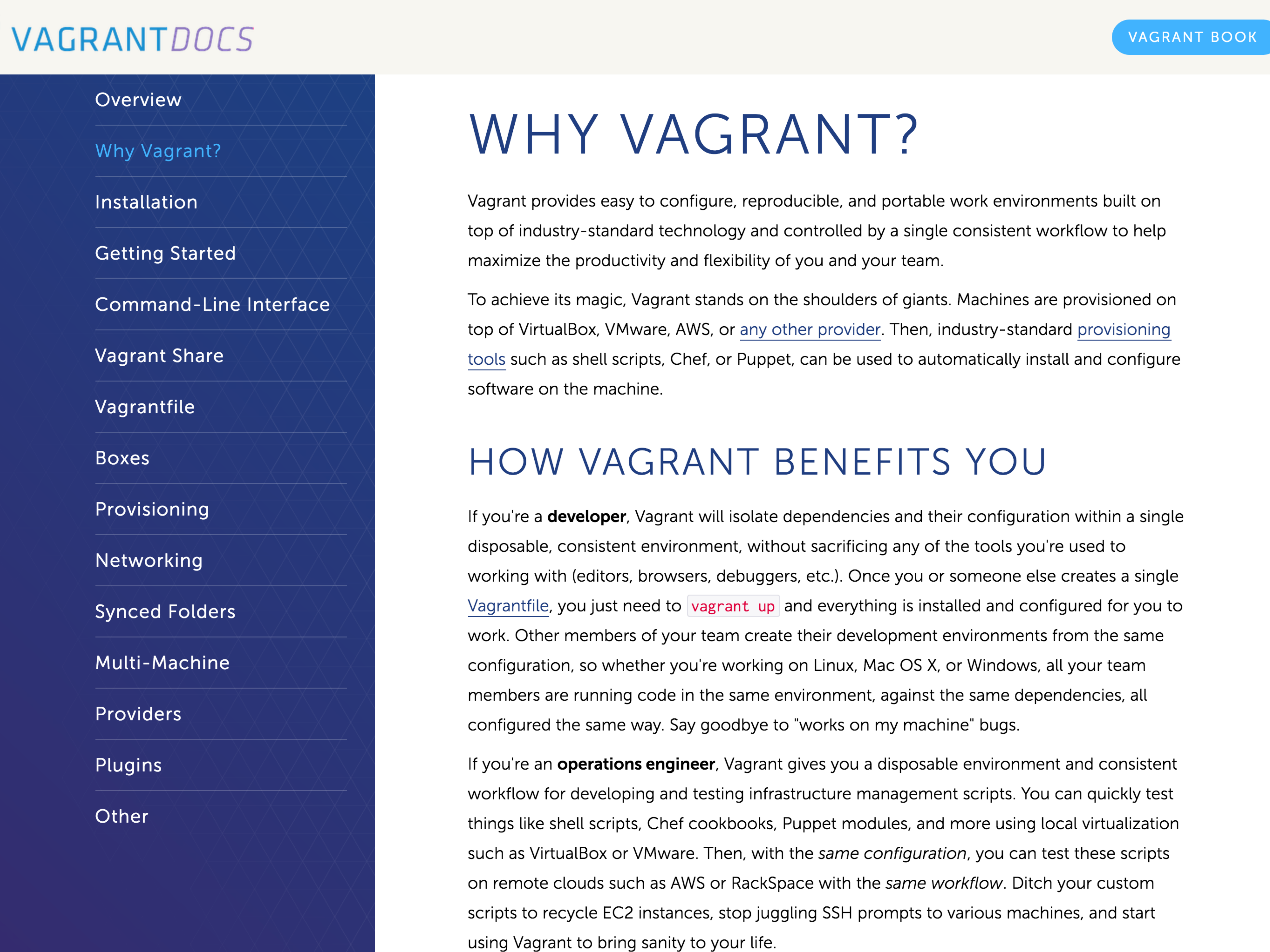
author @reinink | source league/plates | release v3.0.2 | license MIT
build passing | coverage 100% | code quality 9.8 | downloads 4k total

About

Plates is a native PHP template system that's fast, easy to use and easy to extend. It's inspired by the excellent Twig template engine and strives to bring modern template language functionality to native PHP templates. Plates is designed for developers who prefer to use native PHP templates over compiled template languages, such as Twig or Smarty.

Highlights

- Native PHP templates, no new syntax to learn
- Plates is a template system, not a template language
- Plates encourages the use of existing PHP functions
- Increase code reuse with template layouts and inheritance
- Template folders for grouping templates into namespaces
- Data sharing across templates
- Preassign data to specific templates
- Built-in escaping helpers
- Easy to extend using functions and extensions
- Framework-agnostic, will work with any project



WHY VAGRANT?

Vagrant provides easy to configure, reproducible, and portable work environments built on top of industry-standard technology and controlled by a single consistent workflow to help maximize the productivity and flexibility of you and your team.

To achieve its magic, Vagrant stands on the shoulders of giants. Machines are provisioned on top of VirtualBox, VMware, AWS, or any other provider. Then, industry-standard provisioning tools such as shell scripts, Chef, or Puppet, can be used to automatically install and configure software on the machine.

HOW VAGRANT BENEFITS YOU

If you're a **developer**, Vagrant will isolate dependencies and their configuration within a single disposable, consistent environment, without sacrificing any of the tools you're used to working with (editors, browsers, debuggers, etc.). Once you or someone else creates a single Vagrantfile, you just need to `vagrant up` and everything is installed and configured for you to work. Other members of your team create their development environments from the same configuration, so whether you're working on Linux, Mac OS X, or Windows, all your team members are running code in the same environment, against the same dependencies, all configured the same way. Say goodbye to "works on my machine" bugs.

If you're an **operations engineer**, Vagrant gives you a disposable environment and consistent workflow for developing and testing infrastructure management scripts. You can quickly test things like shell scripts, Chef cookbooks, Puppet modules, and more using local virtualization such as VirtualBox or VMware. Then, with the *same configuration*, you can test these scripts on remote clouds such as AWS or RackSpace with the *same workflow*. Ditch your custom scripts to recycle EC2 instances, stop juggling SSH prompts to various machines, and start using Vagrant to bring sanity to your life.

DOCUMENTATION
- Getting Started
  - Embedded Form
  - Custom Forms
  - Mobile Apps
- Charging Cards
- Sending Transfers
- Your Account

REFERENCES
- Stripe.js
- Checkout
- Webhooks
- Testing
- Examples
- API Upgrades
- API Libraries
- Full API Reference

SUBSCRIPTIONS
- Overview
- Getting started
- Integration guide

CONNECT
- Overview
- Getting Started
- Integrating OAuth
- Collecting Fees
- Shared Customers
- Reference

FAQ
- Getting Paid
- Disputes
- SSL

Getting Started

Below you'll find tutorials that will teach you how to use Stripe, and reference documentation for all the moving parts.

On your website

Start accepting payments on your website with our JavaScript libraries. Learn more ▸

In your mobile app

We have toolkits for native iPhone and Android applications. Learn more ▸

Stripe plugins for 3rd party software
As well as the official API libraries listed above, there are a number of third-party plugins and libraries built by our community, such as for Wordpress and Drupal. Learn more ▸

Use Stripe with other services
Stripe has lots of third-party integrations which require no programming, hosting or complicated setup on your behalf, such as Shopify and Wufoo. Learn more ▸

Need help?

We're always happy to help out with code or any other questions you might have. Learn more ▸

API mailing list

Building a third party integration or Connect application? Join our mailing list to be notified of API changes and stay up to date. View mailing list ▸

Bourbon

MIXINS

Animation

Animation-delay

Animation-direction

Animation-duration

Animation-fill-mode

Animation-iteration-count

Animation-name

Animation-play-state

Animation-timing-function

Appearance

Backface-visibility

Background

Background-image

Border-image

Border-radius

Box-sizing

Calc

Columns

Filter

Flexbox

MIXINS

Animation

View Spec View Source

The animation mixin supports comma separated lists of values, which allows different transitions for individual properties to be described in a single style rule. Each value in the list corresponds to the value at that same position in the other properties.

```
box:hover {
  // Animation shorthand works the same as the CSS3 animation shorthand
  @include animation(scale 1.0s ease-in, slide 2.0s ease);

  // The above outputs the same CSS as using independent, granular mixins.
  @include animation-name(scale, slide);
  @include animation-duration(2s);
  @include animation-timing-function(ease);
  @include animation-iteration-count(infinite);
}
```

Demo

Animation-delay

View Spec View Source

The `animation-delay` property specifies when an animation should start.

Getting Started
Using Less
- Installation
- Command-line usage
- Usage in Code
- Configuration
- Third party tools

Command-line With Rhino
Client-side usage
Get Less.js
License FAQs

Using Less

> Less can be used on the command line via npm, downloaded as a script file for the browser or used in a wide variety of third party tools. See the Usage section for more detailed information.

Installation

The easiest way to install Less on the server, is via npm, the node.js package manager, as so:

```
$ npm install -g less
```

Command-line usage

Once installed, you can invoke the compiler from the command-line, as such:

```
$ lessc styles.less
```

This will output the compiled CSS to `stdout`, you may then redirect it to a file of your choice:

```
$ lessc styles.less > styles.css
```

To output minified CSS, simply pass the `-x` option. If you would like more involved minification, the Clean CSS is also available with the `--clean-css` option.

To see all the command line options run lessc without parameters.

Usage in Code

You can invoke the compiler from node, as such:

```
var less = require('less');

less.render('.class { width: (1 + 1) }', function (e, css) {
  console.log(css);
});
```

Preface
↳ Introduction
↳ Quickstart
↳ Release Notes
↳ Upgrade Guide
↳ Contribution Guide

Getting Started
↳ Installation
↳ Configuration
↳ Homestead
↳ Request Lifecycle
↳ Routing
↳ Requests & Input
↳ Views & Responses
↳ Controllers
↳ Errors & Logging

Learning More
↳ Authentication
↳ Billing
↳ Cache
↳ Core Extension
↳ Events
↳ Facades
↳ Forms & HTML
↳ Helpers
↳ IoC Container
↳ Localization

Templates

Controller Layouts
Blade Templating
Other Blade Control Structures
Extending Blade

Controller Layouts

One method of using templates in Laravel is via controller layouts. By specifying the `layout` property on the controller, the view specified will be created for you and will be the assumed response that should be returned from actions.

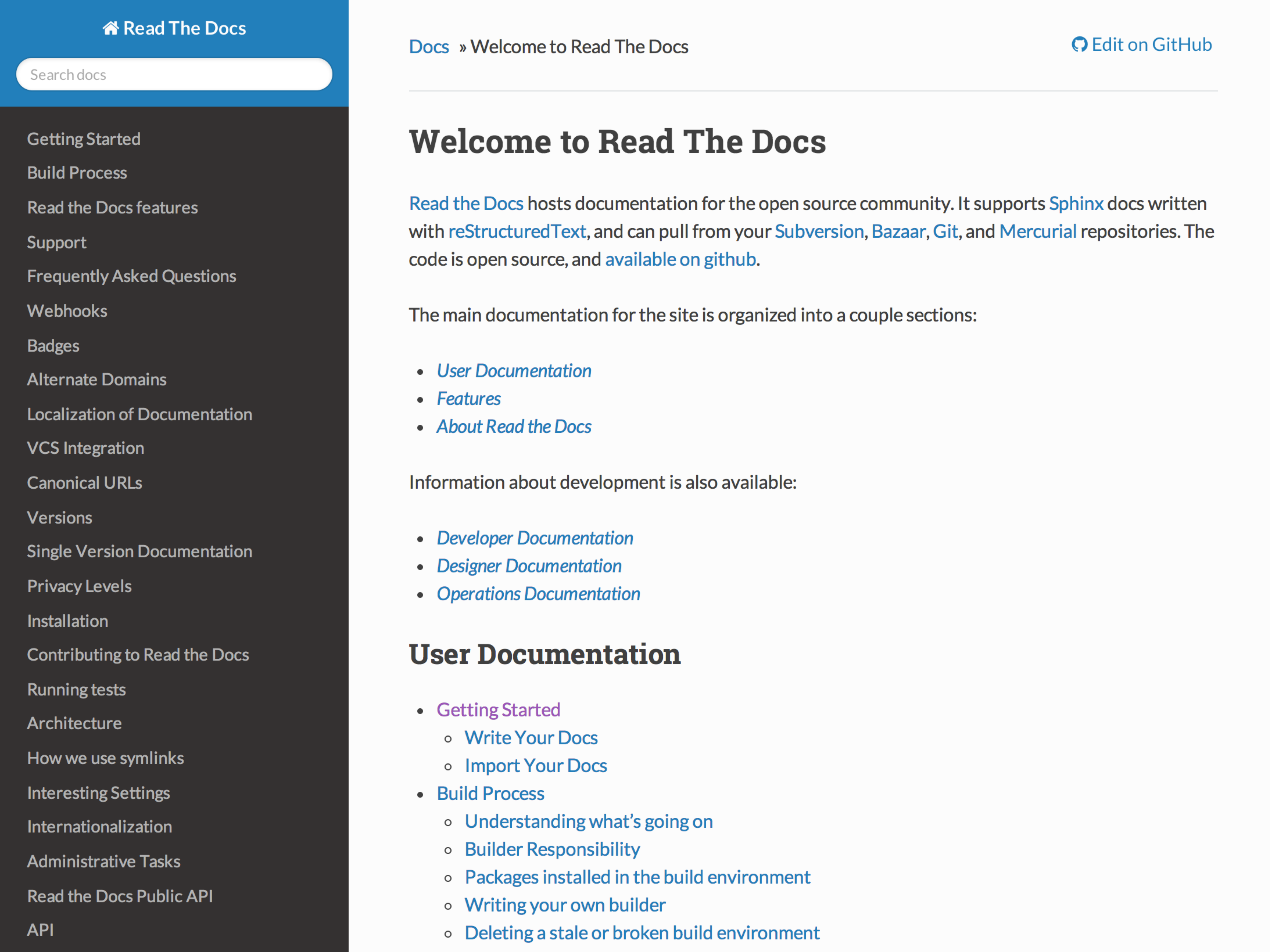
Defining A Layout On A Controller

```
class UserController extends BaseController {

    /**
     * The layout that should be used for responses.
     */
    protected $layout = 'layouts.master';

    /**
     * Show the user profile.
     */
    public function showProfile()
    {
        $this->layout->content = View::make('user.profile');
    }

}
```



Read The Docs

Search docs

- Getting Started
- Build Process
- Read the Docs features
- Support
- Frequently Asked Questions
- Webhooks
- Badges
- Alternate Domains
- Localization of Documentation
- VCS Integration
- Canonical URLs
- Versions
- Single Version Documentation
- Privacy Levels
- Installation
- Contributing to Read the Docs
- Running tests
- Architecture
- How we use symlinks
- Interesting Settings
- Internationalization
- Administrative Tasks
- Read the Docs Public API
- API

Docs » Welcome to Read The Docs

Edit on GitHub

Welcome to Read The Docs

Read the Docs hosts documentation for the open source community. It supports Sphinx docs written with reStructuredText, and can pull from your Subversion, Bazaar, Git, and Mercurial repositories. The code is open source, and available on github.

The main documentation for the site is organized into a couple sections:

- *User Documentation*
- *Features*
- *About Read the Docs*

Information about development is also available:

- *Developer Documentation*
- *Designer Documentation*
- *Operations Documentation*

User Documentation

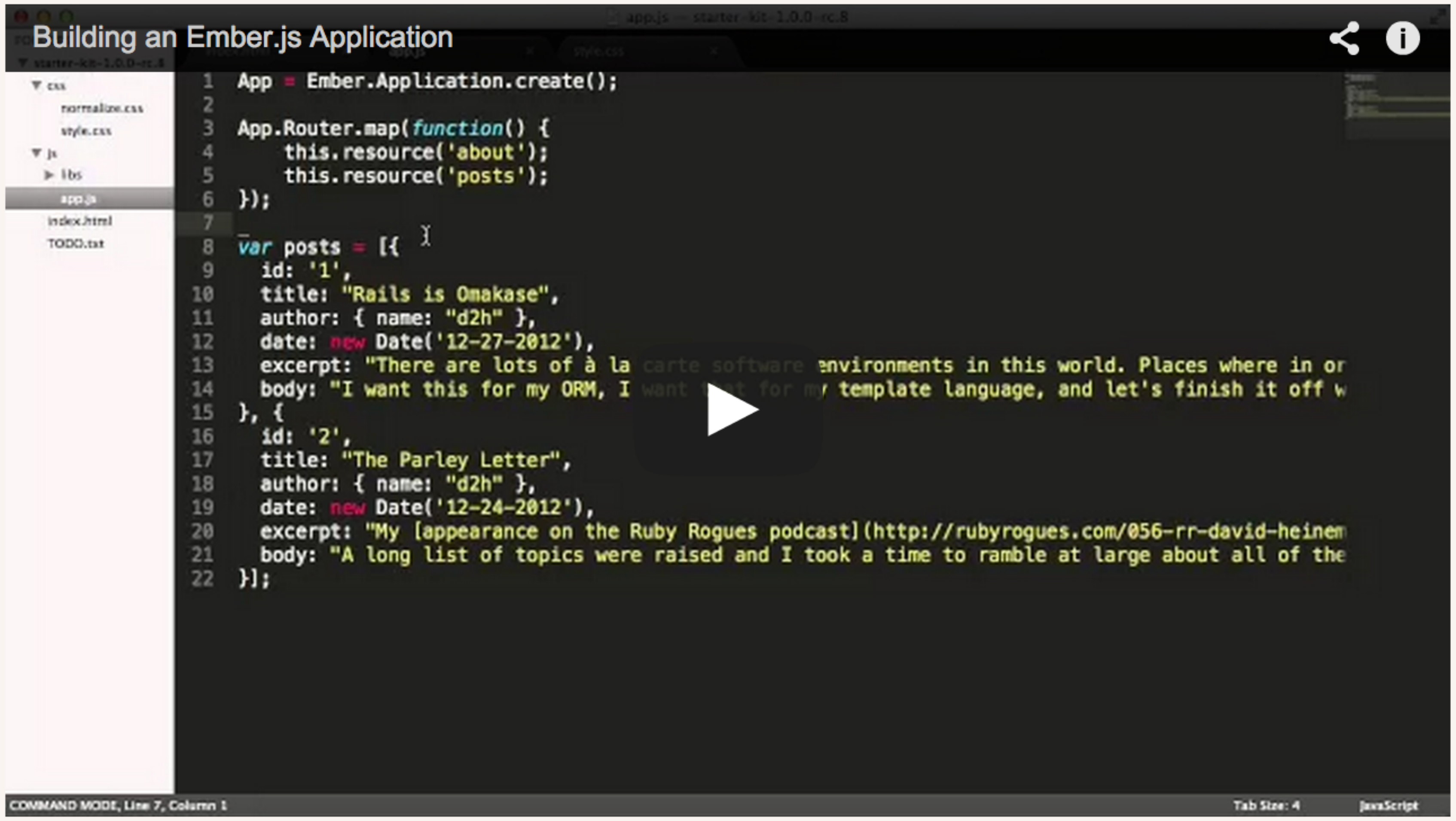
- Getting Started
  - Write Your Docs
  - Import Your Docs
- Build Process
  - Understanding what's going on
  - Builder Responsibility
  - Packages installed in the build environment
  - Writing your own builder
  - Deleting a stale or broken build environment

GETTING STARTED
GETTING EMBER
CONCEPTS
THE OBJECT MODEL
APPLICATION
TEMPLATES
ROUTING
COMPONENTS
CONTROLLERS
MODELS
VIEWS
ENUMERABLES
TESTING
CONFIGURING EMBER.JS
COOKBOOK
UNDERSTANDING EMBER.JS
CONTRIBUTING TO EMBER.JS

EMBER.JS GUIDES

Welcome to the Ember.js guides! This documentation will take you from total beginner to Ember expert. It is designed to start from the basics, and slowly increase to more sophisticated concepts until you know everything there is to know about building awesome web applications.

To help you get started, we've also made a 30-minute screencast that will guide you through building a full-featured Ember.js application:



```
App = Ember.Application.create();

App.Router.map(function() {
    this.resource('about');
    this.resource('posts');
});

var posts = [{
  id: '1',
  title: "Rails is Omakase",
  author: { name: "d2h" },
  date: new Date('12-27-2012'),
  excerpt: "There are lots of à la carte software environments in this world. Places where in or
  body: "I want this for my ORM, I want that for my template language, and let's finish it off w
}, {
  id: '2',
  title: "The Parley Letter",
  author: { name: "d2h" },
  date: new Date('12-24-2012'),
  excerpt: "My [appearance on the Ruby Rogues podcast](http://rubyrogues.com/056-rr-david-heinem
  body: "A long list of topics were raised and I took a time to ramble at large about all of the
}];
```

Source code for the app we build in the video is available at https://github.com/tildeio/bloggr-client

Most of these guides are designed to help you start building apps right away. If you'd like to know more about the thinking behind Ember.js, you'll find what you're looking for in the Understanding Ember.js section.

Some helpful design tools

Just a few of my favourites.

Tell people!

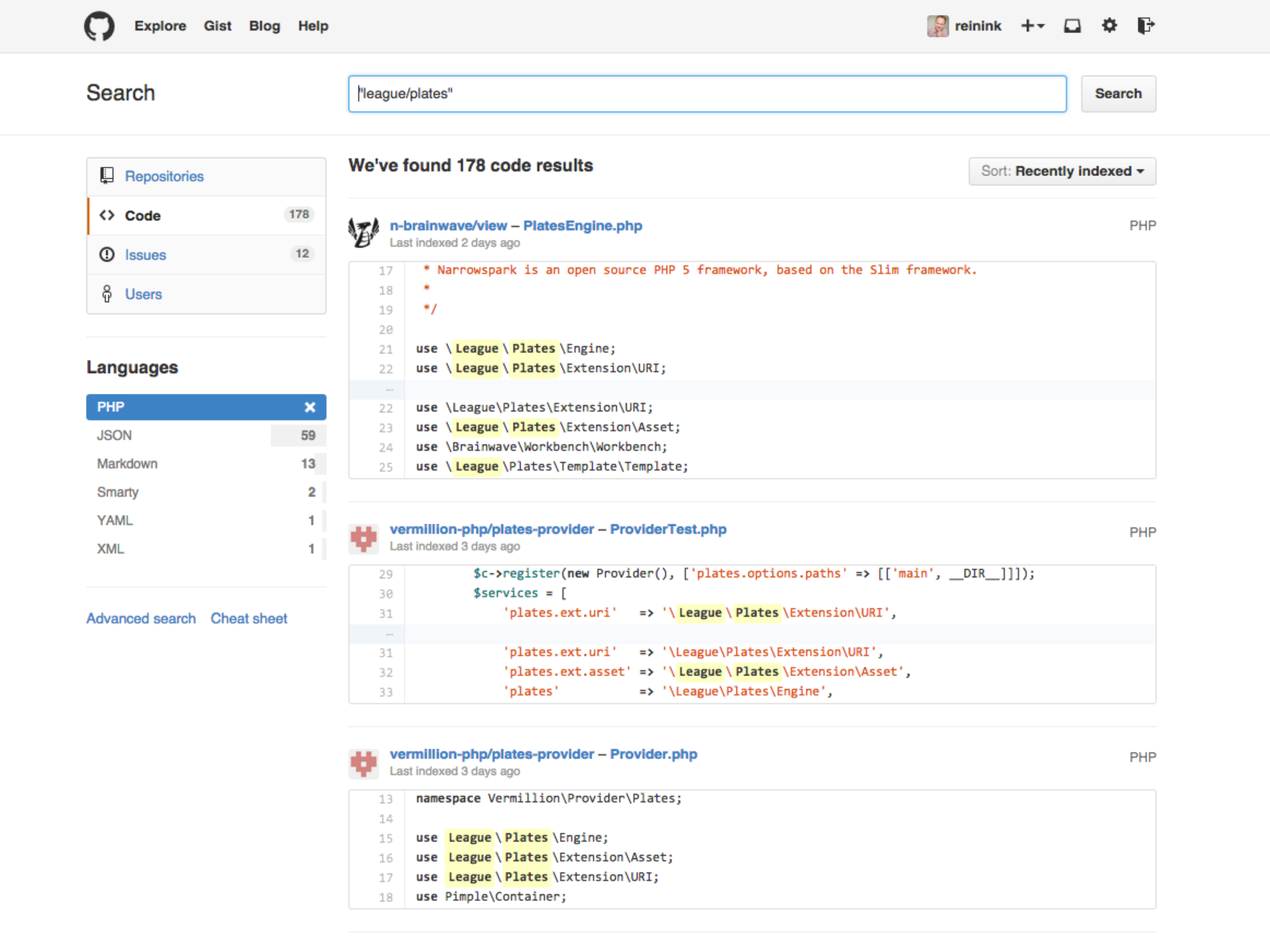
Reddit
Twitter
Hacker News

SitePoint
phpweekly.com
phpdeveloper.org

3. Maintain

Watch it spread

See how your package is actually being used in the real world.



We've found 178 code results

n-brainwave/view – PlatesEngine.php
Last indexed 2 days ago
PHP

```
 * Narrowspark is an open source PHP 5 framework, based on the Slim framework.
 *
 */

use \League\Plates\Engine;
use \League\Plates\Extension\URI;
…
use \League\Plates\Extension\URI;
use \League\Plates\Extension\Asset;
use \Brainwave\Workbench\Workbench;
use \League\Plates\Template\Template;
```

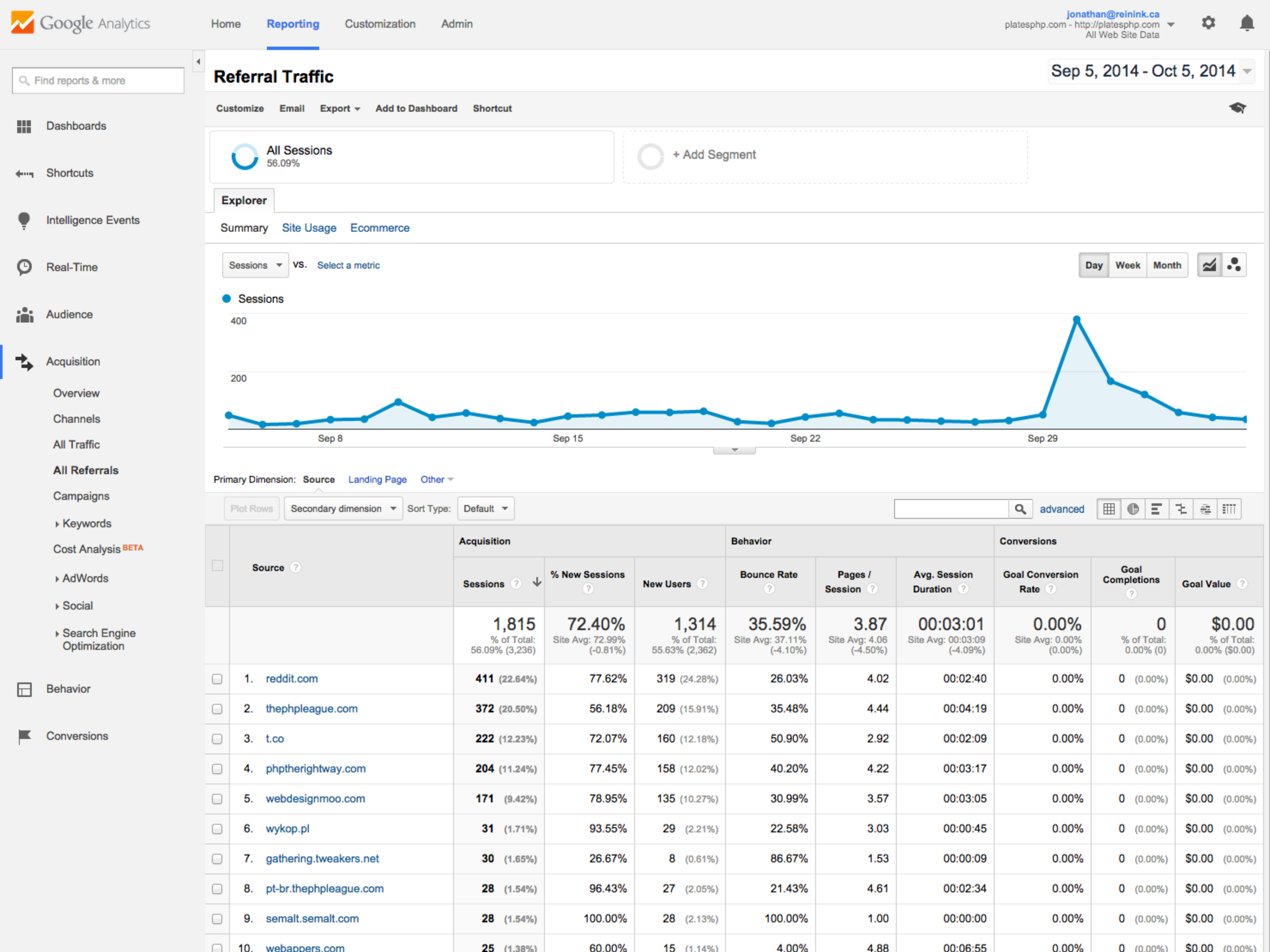
vermillion-php/plates-provider – ProviderTest.php
Last indexed 3 days ago
PHP

```
        $c->register(new Provider(), ['plates.options.paths' => [['main', __DIR__]]]);
        $services = [
            'plates.ext.uri'   => '\League\Plates\Extension\URI',
…
            'plates.ext.uri'   => '\League\Plates\Extension\URI',
            'plates.ext.asset' => '\League\Plates\Extension\Asset',
            'plates'           => '\League\Plates\Engine',
```

vermillion-php/plates-provider – Provider.php
Last indexed 3 days ago
PHP

```
namespace Vermillion\Provider\Plates;

use League\Plates\Engine;
use League\Plates\Extension\Asset;
use League\Plates\Extension\URI;
use Pimple\Container;
```



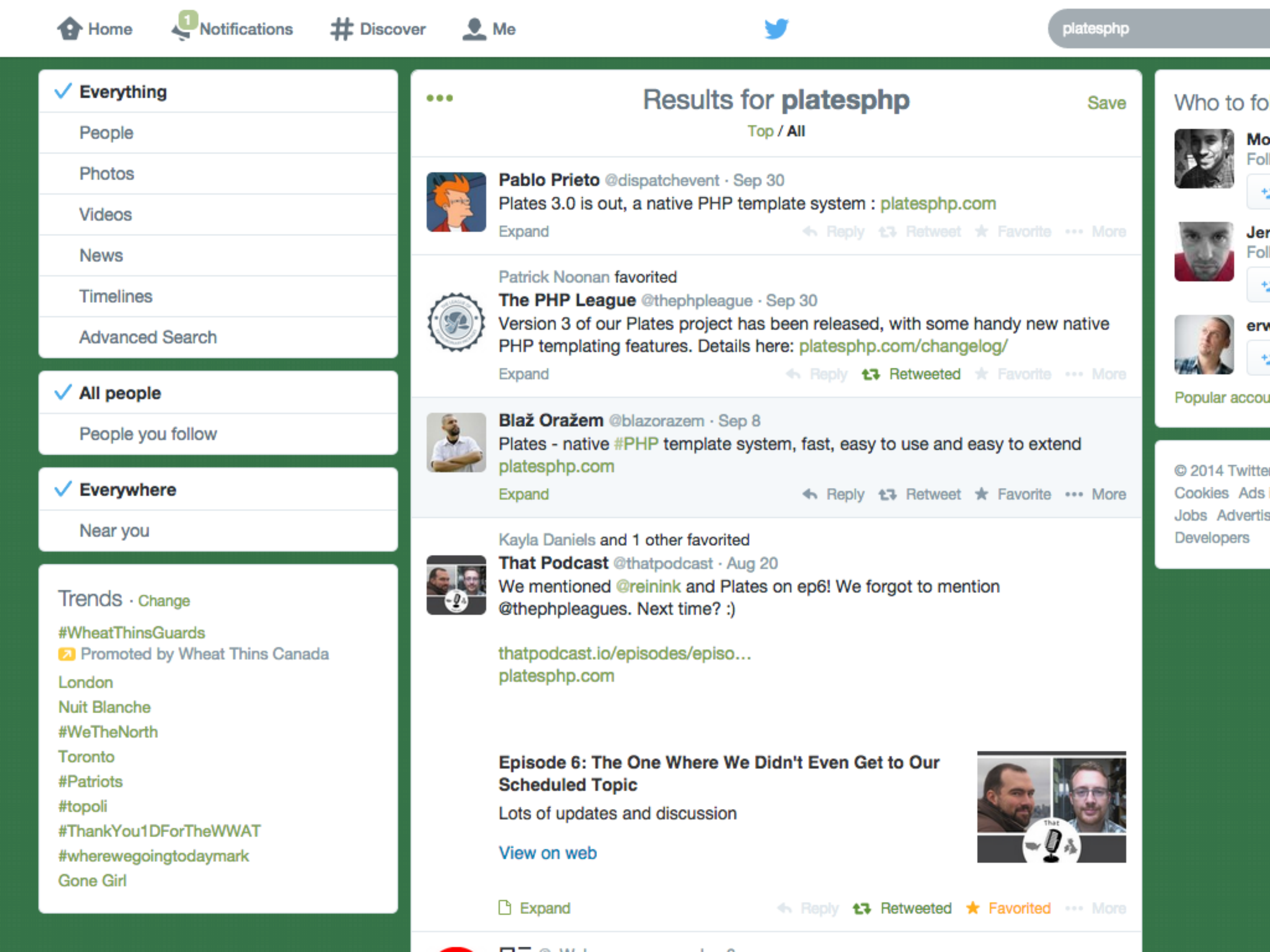
Referral Traffic

Sep 5, 2014 - Oct 5, 2014

All Sessions
56.09%

Primary Dimension: **Source** Landing Page Other

	Acquisition			Behavior			Conversions		
Source	Sessions	% New Sessions	New Users	Bounce Rate	Pages / Session	Avg. Session Duration	Goal Conversion Rate	Goal Completions	Goal Value
	1,815 % of Total: 56.09% (3,236)	72.40% Site Avg: 72.99% (-0.81%)	1,314 % of Total: 55.63% (2,362)	35.59% Site Avg: 37.11% (-4.10%)	3.87 Site Avg: 4.06 (-4.50%)	00:03:01 Site Avg: 00:03:09 (-4.09%)	0.00% Site Avg: 0.00% (0.00%)	0 % of Total: 0.00% (0)	$0.00 % of Total: 0.00% ($0.00)
1. reddit.com	411 (22.64%)	77.62%	319 (24.28%)	26.03%	4.02	00:02:40	0.00%	0 (0.00%)	$0.00 (0.00%)
2. thephpleague.com	372 (20.50%)	56.18%	209 (15.91%)	35.48%	4.44	00:04:19	0.00%	0 (0.00%)	$0.00 (0.00%)
3. t.co	222 (12.23%)	72.07%	160 (12.18%)	50.90%	2.92	00:02:09	0.00%	0 (0.00%)	$0.00 (0.00%)
4. phptherightway.com	204 (11.24%)	77.45%	158 (12.02%)	40.20%	4.22	00:03:17	0.00%	0 (0.00%)	$0.00 (0.00%)
5. webdesignmoo.com	171 (9.42%)	78.95%	135 (10.27%)	30.99%	3.57	00:03:05	0.00%	0 (0.00%)	$0.00 (0.00%)
6. wykop.pl	31 (1.71%)	93.55%	29 (2.21%)	22.58%	3.03	00:00:45	0.00%	0 (0.00%)	$0.00 (0.00%)
7. gathering.tweakers.net	30 (1.65%)	26.67%	8 (0.61%)	86.67%	1.53	00:00:09	0.00%	0 (0.00%)	$0.00 (0.00%)
8. pt-br.thephpleague.com	28 (1.54%)	96.43%	27 (2.05%)	21.43%	4.61	00:02:34	0.00%	0 (0.00%)	$0.00 (0.00%)
9. semalt.semalt.com	28 (1.54%)	100.00%	28 (2.13%)	100.00%	1.00	00:00:00	0.00%	0 (0.00%)	$0.00 (0.00%)
10. webappers.com	25 (1.38%)	60.00%	15 (1.14%)	4.00%	4.88	00:06:55	0.00%	0 (0.00%)	$0.00 (0.00%)



Results for **platesphp**

Pablo Prieto @dispatchevent · Sep 30
Plates 3.0 is out, a native PHP template system : platesphp.com

Patrick Noonan favorited
The PHP League @thephpleague · Sep 30
Version 3 of our Plates project has been released, with some handy new native PHP templating features. Details here: platesphp.com/changelog/

Blaž Oražem @blazorazem · Sep 8
Plates - native #PHP template system, fast, easy to use and easy to extend platesphp.com

Kayla Daniels and 1 other favorited
That Podcast @thatpodcast · Aug 20
We mentioned @reinink and Plates on ep6! We forgot to mention @thephpleagues. Next time? :)

thatpodcast.io/episodes/episo...
platesphp.com

Episode 6: The One Where We Didn't Even Get to Our Scheduled Topic
Lots of updates and discussion

Trends · Change
#WheatThinsGuards
Promoted by Wheat Thins Canada
London
Nuit Blanche
#WeTheNorth
Toronto
#Patriots
#topoli
#ThankYou1DForTheWWAT
#wherewegoingtodaymark
Gone Girl

© 2014 Twitte
Cookies Ads
Jobs Advertis
Developers

Issues and Pull Requests

Open source collaboration is amazing.

Dealing with strong personalities

Sometimes open source collaboration can suck.

Listen to those actually using it

Lots of people will have opinions, but have they ever used your package?

Dealing with backwards compatibility

How to make improvements when they will break existing code.

What to do when you lose interest

Pass off to someone with a vested interest.

✓

Thanks!

Follow me on Twitter at @philsturgeon

https://joind.in/14935