slimframework.com

Documentation - Slim Framework

4 minutu

This documentation is for **Slim 3**. Looking for Slim 2 Docs?.

Welcome

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.

What's the point?

Slim is an ideal tool to create APIs that consume, repurpose, or publish data. Slim is also a great tool for rapid prototyping. Heck, you can even build full-featured web applications with user interfaces. More importantly, Slim is super fast and has very little code. In fact, you can read and understand its source code in only an afternoon!

> 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.

You don't always need a kitchen-sink solution like Symfony or Laravel. These are great tools, for sure. But they are often overkill. Instead, Slim provides only a minimal set of tools that do what you need and nothing else.

How does it work?

First, you need a web server like Nginx or Apache. You should configure your web server so that it sends all appropriate requests to one “front-controller” PHP file. You instantiate and run your Slim app in this PHP file.

A Slim app contains routes that respond to specific HTTP requests. Each route invokes a callback and returns an HTTP response. To get started, you first instantiate and configure the Slim application. Next, you define your application routes. Finally, you run the Slim application. It’s that easy. Here’s an example application:

```
<?php$config = ['settings' => [
    'addContentLengthHeader' => false,
]];
$app = new \Slim\App($config);
$app->get('/hello/{name}', function ($request,
$response, $args) {
    return $response->write("Hello " .
$args['name']);
});
$app->run();
```

Figure 1: Example Slim application

Request and response

When you build a Slim app, you are often working directly with Request and Response objects. These objects represent the actual HTTP request received by the web server and the eventual HTTP response returned to the client.

Every Slim app route is given the current Request and Response objects as arguments to its callback routine. These objects implement the popular PSR 7 interfaces. The Slim app route can inspect or manipulate these objects as necessary. Ultimately, each Slim app route **MUST** return a PSR 7 Response object.

Bring your own components

Slim is designed to play well with other PHP components, too. You can register additional first-party components such as Slim-Csrf, Slim-HttpCache, or Slim-Flash that build upon Slim's default functionality. It's also easy to integrate third-party components found on Packagist.

How to read this documentation

If you are new to Slim, I recommend you read this documentation from start to finish. If you are already familiar with Slim, you can instead jump straight to the appropriate section.

This documentation begins by explaining Slim's concepts and architecture before venturing into specific topics like request and response handling, routing, and error handling.