Enough Said - See For Yourself

Hello, World: The Less-Than-A-Minute
Fat-Free Recipe

Getting Started

> *A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away.*
>
> *– Antoine de Saint-Exupéry*

Fat-Free Framework makes it easy to build entire Web sites in a jiffy. With the same power and brevity as modern Javascript toolkits and libraries, F3 helps you write better-looking and more reliable PHP programs. One glance at your PHP source code and anyone will find it easy to understand, how much you can accomplish in so few lines of code, and how powerful the results are.

F3 is one of the best documented frameworks around. Learning it costs next to nothing. No strict set of difficult-to-navigate directory structures and obtrusive programming steps. No truck load of configuration options just to display `Hello, World` in your browser. Fat-Free gives you a lot of freedom - and style - to get more work done with ease and in less time.

F3's declarative approach to programming makes it easy for novices and experts alike to understand PHP code. If you're familiar with the programming language Ruby, you'll notice the resemblance between Fat-Free and Sinatra micro-framework because they both employ a simple Domain-Specific Language for ReSTful Web services. But unlike Sinatra and its PHP incarnations (Fitzgerald, Limonade, Glue - to name a few), Fat-Free goes beyond just handling routes and requests. Views can be in any form, such as plain text, HTML, XML or an e-mail message. The framework comes with a fast and easy-to-use template engine. F3 also works seamlessly with other template engines, including Twig, Smarty, and PHP itself. Models communicate with F3's data mappers and the SQL helper for more complex interactions with various database engines. Other plug-ins extend the base functionality even more. It's a total Web development framework - with a lot of muscle!

> *Any intelligent fool can make things bigger and more complex... It takes a touch of genius - and a lot of courage to move in the opposite direction.*
>
> *– E. F. Schumacher*

Enough Said - See For Yourself

Unzip the contents of the distribution package (https://github.com/bcosca/fatfree/archive/master.zip) anywhere in your hard drive. If you are using composer, you can `require bcosca/fatfree` to get that package as well. By default, the framework file and optional plug-ins are located in the `lib/` path. Organize your directory structures any way you want. You may move the default folders to a path that's not Web-accessible for better security. Delete the plug-ins that you don't need. You can always restore them later and F3 will detect their presence automatically.

If you are already familiar with composer and fat-free, you probably just want the raw core files instead of the demo package. Therefore you can `require bcosca/fatfree-core` or fetch this fatfree-composer-app (https://github.com/F3Community/fatfree-composer-app) to get started.

> **Important:** If your application uses APC, Memcached, WinCache, XCache, or a filesystem cache, clear all cache entries first before overwriting an older version of the framework with a new one. This can also be done (except XCache) by calling `$f3->clear('CACHE')` .

Make sure you're running the right version of PHP. F3 3.7 does not support versions earlier than PHP 5.4. You'll be getting syntax errors (false positives) all over the place because new language constructs and closures/anonymous functions are not supported by outdated PHP versions. To find out, open your console (`bash` shell on Linux, or `cmd.exe` on Windows):-

```
/path/to/php -v
```

PHP will let you know which particular version you're running and you should get something that looks similar to this:-

```
PHP 5.6.22-1 (cli)
Copyright © 1997-2016 The PHP Group
Zend Engine v2.6.0, Copyright © 1998-2016 Zend Technologies
    with Zend OPcache v7.0.6-dev, Copyright © 1999-2016, by Zend Technologies
```

Upgrade if necessary, check additional system requirements (system-requirements) and come back here if you've made the jump to PHP 5.4 or a later release.

Hello, World: The Less-Than-A-Minute Fat-Free Recipe

Time to start writing our first application:-

```
<?php
$f3 = require('path/to/base.php');
$f3->route('GET /',
    function() {
        echo 'Hello, world!';
    }
);
$f3->run();
```

Prepend `base.php` on the first line with the appropriate path. Save the above code fragment as `index.php` in your Web root folder. We've written our first Web page.

Using composer? Then just run `composer require bcosca/fatfree` and use the following:

```
require 'vendor/autoload.php';
$f3 = \Base::instance();
$f3->route('GET /',
    function() {
        echo 'Hello, world!';
    }
);
$f3->run();
```

The first command tells the PHP interpreter that you want the framework's functions and features available to your application. The `$f3->route()` method informs Fat-Free that a Web page is available at the relative URL indicated by the slash (`/`). Anyone visiting your site located at `http://www.example.com/` will see the `'Hello, world!'` message because the URL `/` is equivalent to the root page. To create a route that branches out from the root page, like `http://www.example.com/inside/` , you can define another route with a simple `GET /inside` string.

The route described above tells the framework to render the page only when it receives a URL request using the HTTP `GET` method. More complex Web sites containing forms use other HTTP methods like `POST` , and you can also implement that as part of a `$f3->route()` specification.

If the framework sees an incoming request for your Web page located at the root URL `/` , it will automatically route the request to the callback function, which contains the code necessary to process the request and render the appropriate HTML stuff. In this example, we just send the string `'Hello, world!'` to the user's Web browser.

So we've established our first route. But that won't do much, except to let F3 know that there's a process that will handle it and there's some text to display on the user's Web browser. If you have a lot more pages on your site, you need to set up different routes for each group. For now, let's keep it simple. To instruct the framework to start waiting for requests, we issue the `$f3->run()` command.

Can't Get the Example Running? If you're having trouble getting this simple program to run on your server, you may have to tweak your Web server settings a bit. Take a look at the sample Apache configuration (routing-engine#sample-apache-configuration) in the following section (along with the Nginx and Lighttpd equivalents).

Still having trouble? Make sure the `$f3 = require('path/to/base.php');` assignment comes before any output in your script. `base.php` modifies the HTTP headers, so any output that has been sent to the browser before this assignment will cause errors.

2. Routing Engine → (routing-engine)