FuelPHP

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia https://twitter.com/tutorialspoint

About the Tutorial

FuelPHP is an open source web application framework, designed for developers who need a simple and elegant toolkit to create full-featured web applications. The development of FuelPHP started in 2010 and it was initially developed by a small team of Philip Sturgeon, Harro Verton, Jelmer Schreuder, and Dan Horrigan.

This tutorial introduces you to FuelPHP framework and makes you comfortable with its various components.

Audience

This tutorial has been prepared for professionals who are aspiring to make a career in FuelPHP framework. This will give you enough understanding on how to create and develop a website using FuelPHP.

Prerequisites

We assume that the readers of this tutorial have a basic knowledge of HTML, PHP, the fundamental concepts of Object-Oriented Programming.

Copyright & Disclaimer

Table of Contents

1. FuelPHP – Introduction

FuelPHP is an open source web application framework. It is written in PHP 5.3 and implements HMVC pattern. **HMVC** is **Hierarchical Model-View-Controller framework** that allows to sub-request the controller, which returns the partial page such as comments, menus, etc., instead of the complete page as in normal MVC.

FuelPHP is created with a desire to incorporate best practices from frameworks such as **CodeIgniter** and **Kohana** with improvements and ideas of its own. FuelPHP database migration tool and scaffolding functionalities are inspired by the popular **Ruby on Rails** framework.

- FuelPHP leverages the **power of command line** through a utility called "**Oil**". The utility is designed to help speed up development, increase efficiency, testing, debugging, and HTML support.

- FuelPHP is purely an object-oriented approach. Its architecture is based on the idea of modularity. Applications can be divided into modules and every component can be extended or replaced without rewriting a single line of code. Fuel supports any template parser such as **Smarty**, **Twig**, **PHPTal**, etc. for parsing views.

- **FuelPHP community is large** and active with over 300 contributors. Its large community regularly creates and improves packages and extensions. The main objective of FuelPHP framework is to provide flexibility and compatibility. It is fast, easy to learn, and a complete solution for developing web applications.

- What makes FuelPHP one of the premier frameworks used by PHP developers is that – the **new version of FuelPHP is reverse-compatible with its older versions** because of its stable API. It is extremely flexible.

- **Packages and modules** make it easy and simple to reuse an existing code in a systematic way. FuelPHP offers maximum performance through a small library. Its interactive debugging allows to easily eliminate the errors in development. Also, its clean and stable code makes programming easier.

FuelPHP – Features

FuelPHP offers lot of features to create a full-fledged web application. It provides flexible components, simple configuration, easy-to-use ORM, REST based application development mode, etc. Following are some of the salient features:

- Flexible and community driven web framework
- Easy to configure and use
- FuelPHP is extremely portable, works on almost any server
- Flexible URI routing system
- FuelPHP provides RESTful API development support
- Lightweight ORM model
- Input filtering and prevents SQL injection

- Secure authentication and authorization framework
- Code reusable and easier to maintain
- Autoloading classes, Session management, and Exception handling.

FuelPHP – Advantages

FuelPHP is an elegant HMVC PHP 5.3 framework that provides a set of components for building web applications with the following advantages:

- **Modular structure** – Fuel doesn't force you to use modules or an HMVC file structure. If you want to use, the process is quite easy to integrate. FuelPHP apps are created in a modular structure and becomes easier for developers with clear benefits.

- **HMVC pattern** – The most important feature of this framework is HMVC (Hierarchical Model View Controller) which makes it easy to access or use any properties, class methods, functions, files at higher level.

- **Secure hashing function** – FuelPHP supports strong cryptography tools and password hashing techniques. It handles encryption, decryption, and hashing using the powerful PHPSecLib.

- **Scaffolding functionality** – Scaffolding is a meta-programming method for building database operations. Fuel's scaffolding is pretty easy. It allows you to get a basic CRUD application with very simple steps.

The following popular products use FuelPHP Framework:

- **Matic Technology** – Global provider of offshore custom software development solutions. At Matic Technologies, they provide all the best possible solutions through FuelPHP according to the requirements of the client.

- **Kroobe** – Kroobe is a social networking classifieds company. Fuel offers extremely low development costs and services to Kroobe team to achieve efficient solution.

2. FuelPHP – Installation

This chapter explains how to install FuelPHP framework on your machine. FuelPHP installation is very simple and easy. You have two methods to create FuelPHP applications:

- The first method is **Command line** installation using FuelPHP tool called **Oil**.
- The second method is **Composer based** installation. FuelPHP uses **Composer**, both for installation and for package dependencies, so make sure the composer is installed locally before continuing the process.

Let's go through each of the methods one by one in detail in the subsequent sections.

System Requirements

Before moving to installation, the following system requirements have to be satisfied.

Web server (Any of the following)

- WAMP (Windows)
- Microsoft IIS (Windows)
- LAMP (Linux)
- MAMP (Macintosh)
- XAMP (Multi-platform)
- Nginx (Multi-platform)
- PHP in-built development web server (Multi-platform)

Browser support (Any of the following)

- IE (Internet Explorer 8+)
- Firefox
- Google Chrome
- Safari

PHP compatibility: PHP 5.3 or later. To get the maximum benefit, use the latest version.

Let us use PHP's in-built development web server for this tutorial. The built-in development web server is easy to start as well as quite adequate to understand the basics of FuelPHP web application without getting into the complexity of the world of web server and configurations.

Command Line Installation

The command line installation of FuelPHP is very easy and takes maximum of five minutes.

Install Oil Package

Oil is a special package/command provided by FuelPHP framework to do lot of tasks needed in the development of FuelPHP application including installation, development, and testing the application.

To install the **Oil** package, open up a shell and run the following command:

```
sudo curl https://get.fuelphp.com/oil | sh
```

The command uses curl to download and install the *oil* package. The command will show result similar to the following information and finally install the oil package.

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   479  100   479    0     0    353      0  0:00:01  0:00:01 --:--:--   353
```

Create a New Project

To create a new project using Oil, use the following command:

```
oil create <project_name>
```

Let's create a new project named "HelloWorld" using the following command.

```
oil create HelloWorld
```

Now, you can see response similar to the following and finally create a simple skeleton FuelPHP application.

```
composer create-project fuel/fuel HelloWorld
Installing fuel/fuel (1.8.0.1)
  - Installing fuel/fuel (1.8.0.1)
    Loading from cache

Created project in HelloWorld

Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing composer/installers (v1.3.0)
    Loading from cache
```

```
- Installing fuelphp/upload (2.0.6)
  Loading from cache

- Installing michelf/php-markdown (1.4.0)
  Loading from cache

- Installing psr/log (1.0.2)
  Loading from cache

- Installing monolog/monolog (1.18.2)
  Loading from cache

- Installing phpseclib/phpseclib (2.0.0)
  Loading from cache

- Installing fuel/core (1.8.0.4)
  Loading from cache

- Installing fuel/auth (1.8.0.4)
  Loading from cache

- Installing fuel/email (1.8.0.4)
  Loading from cache

- Installing fuel/oil (1.8.0.4)
  Loading from cache

- Installing fuel/orm (1.8.0.1)
  Loading from cache

- Installing fuel/parser (1.8.0.4)
  Loading from cache

- Installing fuel/docs (1.8.0.4)
  Loading from cache
```

```
  ..............
  ..............

  Writing lock file
  Generating autoload files
```

Oil Version

To test whether Oil is available and to check the version, use the following command:

```
$ cd HelloWorld
$ php oil -v
```

The above command produces the following result:

```
Fuel: 1.8 running in "development" mode
```

Oil Help Command

To obtain Oil's basic help documentation, use the following command:

```
$ php oil help
```

The above command will show the help documentation similar to the following result:

```
Usage:
  php oil [cell|console|generate|package|refine|help|server|test]


Runtime options:
  -f, [--force]    # Overwrite files that already exist
  -s, [--skip]     # Skip files that already exist
  -q, [--quiet]    # Supress status output
  -t, [--speak]    # Speak errors in a robot voice


Description:
  The 'oil' command can be used in several ways to facilitate quick development, help
  with testing your application and for running Tasks.


Environment:
  If you want to specify a specific environment oil has to run in, overload the
  environment variable on the commandline: FUEL_ENV=staging php oil <commands>
```

```
More information:

  You can pass the parameter "help" to each of the defined command to get
  information about that specific command: php oil package help


Documentation:

  http://docs.fuelphp.com/packages/oil/intro.html
```

As of now, you have an idea of how to install Fuel using Oil. Let's go through the composer based installation in the next section.

Composer-based Installation

The following command is used to install FuelPHP using Composer.

```
$ composer create-project fuel/fuel --prefer-dist .
```

Git Repository Clones

To install the latest development version as local git repository clones, use the following command.

```
$ composer create-project fuel/fuel:dev-1.9/develop --prefer-source .
```

Running the Application

Move to the project directory public folder, run the application using the production server with the following command.

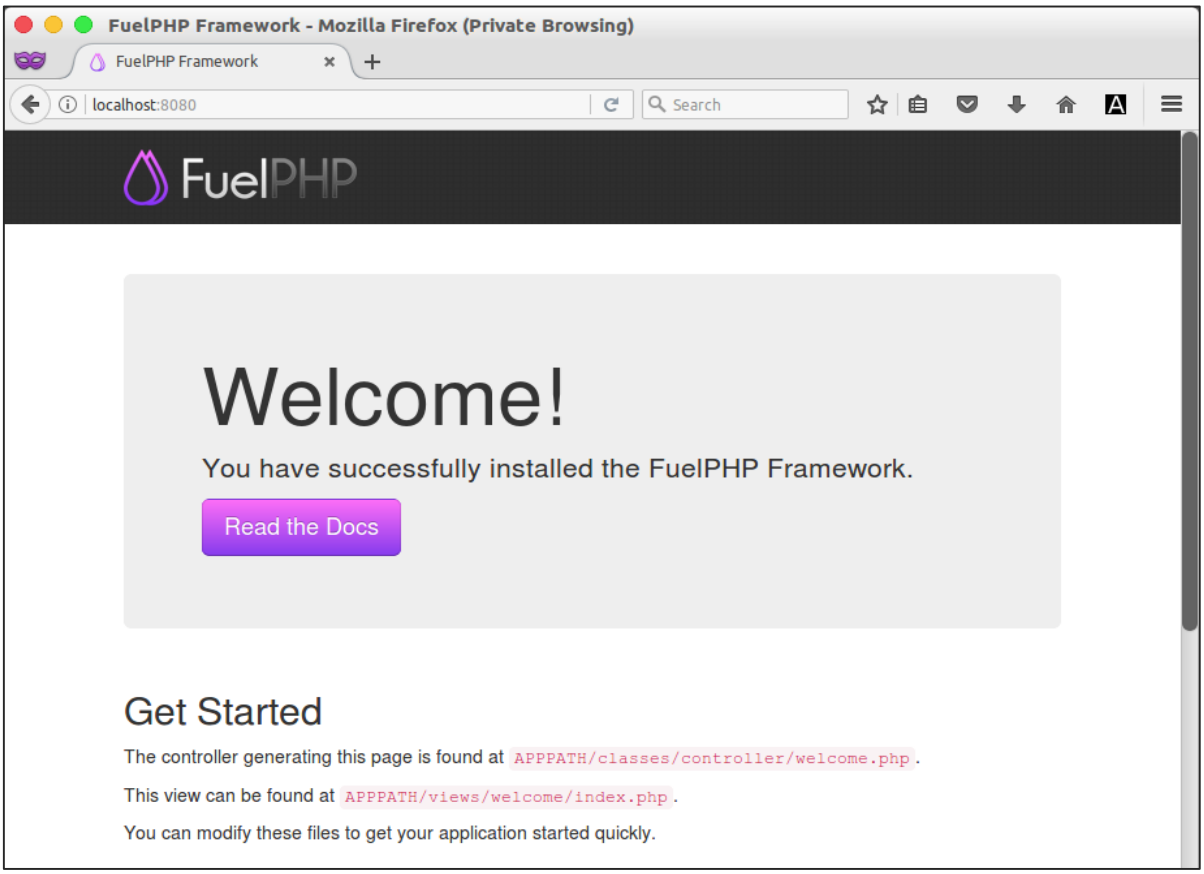
```
$ cd path/to/HelloWorld/public
$ php -S localhost:8080 index.php
```

It produces the following response.

```
PHP 5.5.31 Development Server started at Sun May 21 12:26:10 2017
Listening on http://localhost:8080
Document root is /Users/workspace/php-fuel/HelloWorld/public
Press Ctrl-C to quit.
```

Now, request the URL, *http://localhost:8080* and it will produce the following result.

Result



This is the simplest way to run FuelPHP application in the development environment. If you create your application in this way in the production environment, you will face security issues. The recommended way is setting up a virtual host configuration. It is explained for apache web server in the next section.

Setting Up a Virtual Host

It is more secure way to access FuelPHP application. To set up a virtual host, you need to link apache virtual host file to your application. In case of intranet application, redirect system host file URL to virtual host.

Virtual Host File

Open the virtual host and add the following changes.

```
<VirtualHost *:80>
    ServerName hello.app
    DocumentRoot /path/to/public
    SetEnv FUEL_ENV “development”
    <Directory /path/to/public>
```

```
        DirectoryIndex index.php
        AllowOverride All
        Order allow,deny
        Allow from all
    </Directory>
</VirtualHost>
```

System Host File

Now, add a host entry to your machine using the following command.

```
sudo vi /etc/hosts
```

Then, add the following line to the end of the file.

```
127.0.0.1 hello.app
```

To make all the changes available, restart your Apache server and request the url, *http://hello.app.* It produces the FuelPHP home page.

3. FuelPHP – Architecture Overview

FuelPHP is based on battle tested **Model-View-Controller** architecture along with **HMVC (Hierarchical MVC)** support. While MVC provides flexible and layered application development, HMVC goes one step further to enable widgetization of the web application.

The strength of FuelPHP is that it does not enforce specific ways to develop an application. It just provides a simple and easy-to-use standard structure. Developers are free to use the pre-defined set of functionality provided by FuelPHP or modify it whenever needed. All the features provided by FuelPHP including the core feature can be changed according to the requirement of the application.

Model

Model is the business entity of the application. Controller and View exchange data in the form of Model. Model enables uniform representation of our business data. It enables the database layer to interact with the web application layer in the standard way and provides an option to select, save, edit, and delete our database entities.

Controller

A typical MVC application starts from a Controller. Once a user sends a request to the FuelPHP web application, the application gathers all the information about the request and sends it to the Controller. Controller does the required business logic of the requested page and then calls the relevant View along with the processed data in the form of Models.

View

View is the presentation layer of the MVC application. View decides how to show the Model to the user. It supports simple data rendering to the advanced layout, which enables the website to normalize the design across all the pages. View also provides theming support, which enables quick design change across the application.

Presenter

Presenter is a special feature provided by FuelPHP. It is the glue between Controller and View. Controller can share some of its low level responsibility such as retrieving model from database, generating data for the view, etc. Controller calls Presenter instead of View, which in turn calls View. Presenter enables pure separation of business logic and presentation layer.

Hierarchical MVC

FuelPHP provides an option to call one controller from another controller, similar to the request from the client (browser). If any controller calls another controller, the called controller will return the response to the calling controller instead of rendering it to the client (browser). This enables **widgetization** of the web application. For example, the comment section can be showed as a stand-alone page as well as a sub-section of the main (blog) page.

Module

One of the salient features of FuelPHP is that a section of the web application can be converted into modules, which can be shared among the different application. For example, a blog module created for an application can be reused in another application by just copying the module code from source application to target application.

Note that creating a new module is as simple as developing the main application. The structure is similar to the main application with the only exception that the module should be coding a separate folder.

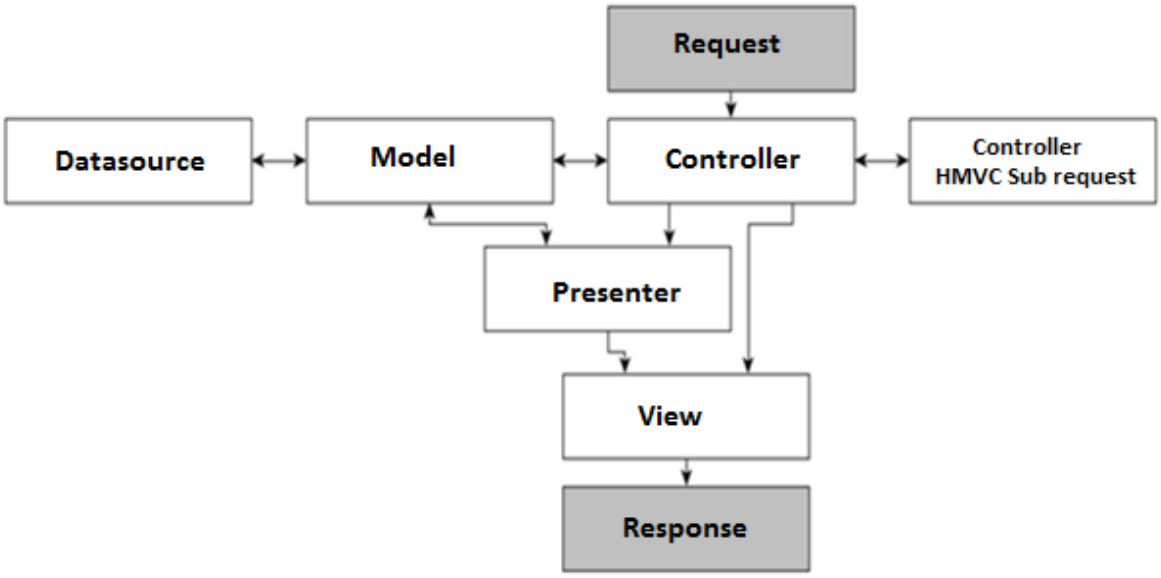
Package

FuelPHP provides an option to organize the code into a single unit called Package. A package can contain one or more functionality needed for the web application. For example, a database component such as ORM, email, etc., can be organized into a package and used whenever needed.

A Package is different from a Module in the sense that the Package does not contain any web pages or partial web pages. Package can be used in FuelPHP as well as any other PHP framework.

Workflow

The workflow of the FuelPHP is simple and easy to understand. It is depicted in the following diagram.



- User sends a request to the application.
- Controller receives the request and gathers information by interacting with the model, which in turn interacts with the database.
- Controller gathers information by interacting with other controller by sending a sub-request to the other controllers.

- Controller sends the retrieved model to the view, which in turn generates the presentation and sends it to the client as a response.

- In some cases, the controller may pass the control to the presenter. In that case, the presenter gathers information from the model and sends it to the client. Here, the presenter does not perform any business logic, except retrieve the model from the database.

4. FuelPHP – Simple Web Application

In this chapter, we will see how to create a simple application in FuelPHP framework. As discussed earlier, you know how to create a new project in Fuel. We can take an example of Employee details.

Let's start by creating a project named Employee using the following command.

```
oil create employee
```

After executing the command, an **employee** project is created with the following **file structure**:

```
employee
├── CHANGELOG.md
├── composer.json
├── composer.lock
├── composer.phar
├── CONTRIBUTING.md
├── fuel
│   ├── app
│   │   ├── bootstrap.php
│   │   ├── cache
│   │   ├── classes
│   │   ├── config
│   │   ├── lang
│   │   ├── logs
│   │   ├── migrations
│   │   ├── modules
│   │   ├── tasks
│   │   ├── tests
│   │   ├── themes
│   │   ├── tmp
│   │   ├── vendor
│   │   └── views
│   ├── core
│   │   ├── base56.php
│   │   ├── base.php
│   │   ├── bootstrap.php
```

```
│   │   ├── bootstrap_phpunit.php
│   │   ├── classes
│   │   ├── composer.json
│   │   ├── config
│   │   ├── CONTRIBUTING.md
│   │   ├── lang
│   │   ├── phpunit.xml
│   │   ├── tasks
│   │   ├── tests
│   │   ├── vendor
│   │   └── views
│   ├── packages
│   │   ├── auth
│   │   ├── email
│   │   ├── oil
│   │   ├── orm
│   │   └── parser
│   └── vendor
│       ├── autoload.php
│       ├── composer
│       ├── fuelphp
│       ├── michelf
│       ├── monolog
│       ├── phpseclib
│       └── psr
├── LICENSE.md
├── oil
├── public
│   ├── assets
│   │   ├── css
│   │   ├── fonts
│   │   ├── img
│   │   └── js
│   ├── favicon.ico
│   ├── index.php
│   └── web.config
```

```
├── README.md
└── TESTING.md


42 directories, 21 files
```

Structure of the Application

FuelPHP framework provides a well-organized application structure. Let us check some of the important files and folders of the application.

- **fuel** - Contains all the PHP files.
- **public** - Contains all the assets which are directly accessed through the browser such as JavaScript, CSS, images, etc.
- **oil** - An executable used to run command line tasks such as generating code or interactive debugging within your application. It's optional.
- **fuel/app/** - Contains all application-specific PHP files. It contains Models, Views, and Controllers.
- **fuel/core/** - This is where Fuel framework itself lives.
- **fuel/packages/** - Contains all fuel packages. By default, fuel will contain three packages: oil, auth, and orm. These packages will not be loaded unless you require them.
- **fuel/app/config/** - Contains all application-related configuration files. The main application configuration file, config.php file is located here.
- **fuel/app/classes/** - Contains all the application specific MVC based PHP files. It contains controllers, models, helper classes, libraries, etc.
- **fuel/app/classes/controller/** - Controllers are placed here.
- **fuel/app/classes/model/** - Models are placed here.
- **fuel/app/views/** - Contains view files. There are no specific naming conventions for views.

Add a Controller

As discussed earlier, FuelPHP is based on the Model-View-Controller (MVC) development pattern. MVC is a software approach that separates application logic from presentation. In MVC pattern, the controller plays an important role and every webpage in an application needs to be handled by a controller. By default, controllers are located in **fuel/app/classes/controller/** folder. You can create your own Controller class here.

Move to the location fuel/app/classes/controller/ and create employee.php file. To create a new controller, just extend the Controller class provided by FuelPHP, defined as follows.

employee.php

```
<?php
class Controller_Employee extends Controller {
    public function action_home()
    {
        // functionality of the home page
    echo "FuelPHP-Employee application!";
    }
}
```

Now, we have created an Employee Controller and added a public method, action_home, which prints a simple text.

Routing

Routing resolves the web page URI into specific controller and action. Every webpage in a FuelPHP application should go through routing before the actual execution of the controller. By default, each controller can be resolved using the following URI pattern.

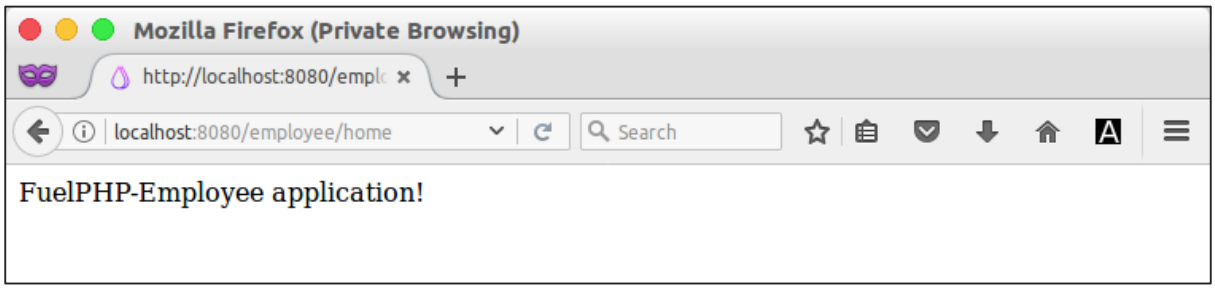
```
<controller>/<action>
```

where,

- **controller** is the name of the controller minus namespace, employee
- **action** is the name of the method minus action_ keyword, home

The newly created controller can be accessed by *http://localhost:8080/employee/home* and it will produce the following result.

Result



==========

End of ebook preview

If you liked what you saw…

Buy it from our store @ https://store.tutorialspoint.com