Symfony

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

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

About the Tutorial

Symfony is an open-source PHP web application framework, designed for developers who need a simple and elegant toolkit to create full-featured web applications. Symfony is sponsored by SensioLabs. It was developed by Fabien Potencier in 2005. This tutorial will give you a quick introduction to Symfony framework and make you comfortable with its various components.

Audience

This tutorial has been prepared for beginners who want to learn the fundamental concenpts of Symfony framework. The readers will get enough understanding on how to create and develop a website using Symfony.

Prerequisites

Before proceeding with the various types of components given in this tutorial, it is being assumed that the readers are already aware about what a Framework is. In addition to this, it will also be very helpful if you have a sound knowledge on HTML, PHP, and the OOPS concepts.

Copyright & Disclaimer

© Copyright 2017 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

tutorialspoint
SIMPLYEASYLEARNING

Table of Contents

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

1. Symfony – Introduction

A PHP web framework is a collection of classes, which helps to develop a web application. Symfony is an open-source MVC framework for rapidly developing modern web applications. Symfony is a full-stack web framework. It contains a set of reusable PHP components. You can use any Symfony components in applications, independently from the framework.

Symfony has a huge amount of functionality and active community. It has a flexible configuration using YAML, XML, or annotations. Symfony integrates with an independent library and PHP Unit. Symfony is mainly inspired by Ruby on Rails, Django, and Spring web application frameworks. Symfony components are being used by a lot of open source projects that include Composer, Drupal, and phpBB.

The Symfony framework consists of several components, such as the HttpFoundation component that understands HTTP and offers a nice request and response object used by the other components. Others are merely helper components, such as the Validator, that helps to validate data. Kernel component is the heart of the system. Kernel is basically the 'main class' that manages the environment and has the responsibility of handling a http request.

Symfony's well-organized structure, clean code, and good programming practices make web development easier. Symfony is very flexible, used to build micro-sites and handle enterprise applications with billions of connections.

Symfony Framework – Features

Symfony is designed to optimize the development of web applications and grows in features with every release.

Some of the salient features of Symfony Framework is as follows:

- Model-View-Controller based system
- High-performance PHP framework
- Flexible URI routing
- Code reusable and easier to maintain
- Session management
- Error logging
- Full-featured database classes with support for several platforms
- Supports a huge and active community
- Set of decoupled and reusable components
- Standardization and interoperability of applications
- Security against cross-site request forgery and other attacks
- Twig template engine.

Symfony offers a lot of flexibility to developers. It has great features for debugging, code readability, and developing extensible programs.

tutorialspoint
SIMPLYEASYLEARNING

Symfony is a full-stack web framework; it is a very effective tool for creating web applications. Numerous companies offer Symfony services to clients.

Following are some of the benefits that you get by using the Symfony Framework.

- **Microframework** - Symfony can be used to develop a specific functionality. You don't need to redevelop or install the entire framework.
- Reduces development time overhead.
- Extremely mature templating engine and quickly delivers content to the users.
- **Compatible and extensible** – Programmers can easily extend all framework classes.

Symfony Framework — Applications

Symfony components can be used as a part of other applications such as Drupal, Laravel, phpBB, Behat, Doctrine, and Joomla.

- **Drupal 8** — Drupal is an open source content management PHP framework. Drupal 8 uses core layers of Symfony and extends it to provide support for Drupal modules.
- **Thelia** — Thelia is a Symfony-based e-commerce solution. Initially, Thelia was written in PHP code and MySQL, however, it was lagging to produce faster applications. To overcome this drawback, Thelia integrated with Symfony to develop the applications in a customizable way.
- **Dailymotion** — Dailymotion is one of the world's largest independent video entertainment website based in France. Once they decided to migrate open source framework with a large community, Dailymotion developers decided to use Symfony components features for its flexibility.

tutorialspoint
SIMPLYEASYLEARNING

2. Symfony – Installation

This chapter explains how to install Symfony framework on your machine. Symfony framework installation is very simple and easy. You have two methods to create applications in Symfony framework. First method is using Symfony Installer, an application to create a project in Symfony framework. Second method is composer-based installation. Let's go through each of the methods one by one in detail in the following sections.

System Requirements

Before moving to installation, you require the following system requirements.

- Web server (Any one of the following)
  - WAMP (Windows)
  - LAMP (Linux)
  - XAMP (Multi-platform)
  - MAMP (Macintosh)
  - Nginx (Multi-platform)
  - Microsoft IIS (Windows)
  - PHP built-in development web server (Multi-platform)
- Operating System: Cross-platform
- Browser Support: IE (Internet Explorer 8+), Firefox, Google Chrome, Safari, Opera
- PHP Compatibility: PHP 5.4 or later. To get the maximum benefit, use the latest version.

We will use PHP built-in development web server for this tutorial.

Symfony Installer

Symfony Installer is used to create web applications in Symfony framework. Now, let's configure the Symfony installer using the following command.

```
$ sudo mkdir -p /usr/local/bin

$ sudo curl -LsS https://symfony.com/installer -o /usr/local/bin/symfony

$ sudo chmod a+x /usr/local/bin/symfony
```

Now, you have installed Symfony installer on your machine.

Create Your First Symfony Application

Following syntax is used to create a Symfony application in the latest version.

Syntax

```
symfony new app_name
```

Here, app_name is your new application name. You can specify any name you want.

Example

```
symfony new HelloWorld
```

After executing the above command, you will see the following response.

```
Downloading Symfony...

0 B/5.5 MiB ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒

……

……

Preparing project...

 ✓  Symfony 3.2.7 was successfully installed. Now you can:

    * Change your current directory to /Users/../workspace/firstapp

    * Configure your application in app/config/parameters.yml file.

    * Run your application:
        1. Execute the php bin/console server:run command.
        2. Browse to the http://localhost:8000 URL.

    * Read the documentation at http://symfony.com/doc
```

This command creates a new directory called "firstapp/" that contains an empty project of Symfony framework latest version.

tutorialspoint
SIMPLYEASYLEARNING

Install Specific Version

If you need to install a specific Symfony version, use the following command.

```
symfony new app_name 2.8
symfony new app_name 3.1
```

Composer-based Installation

You can create Symfony applications using the Composer. Hopefully, you have installed the composer on your machine. If the composer is not installed, download and install it.

The following command is used to create a project using the composer.

```
$ composer create-project symfony/framework-standard-edition app_name
```

If you need to specify a specific version, you can specify in the above command.

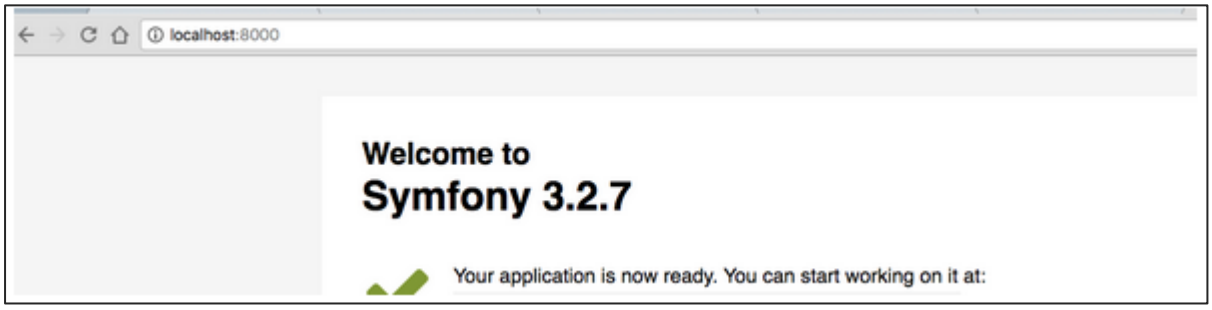
Running the Application

Move to the project directory and run the application using the following command.

```
cd HelloWorld
php bin/console server:run
```

After executing the above command, open your browser and request the url http://localhost:8000/. It produces the following result.

Result



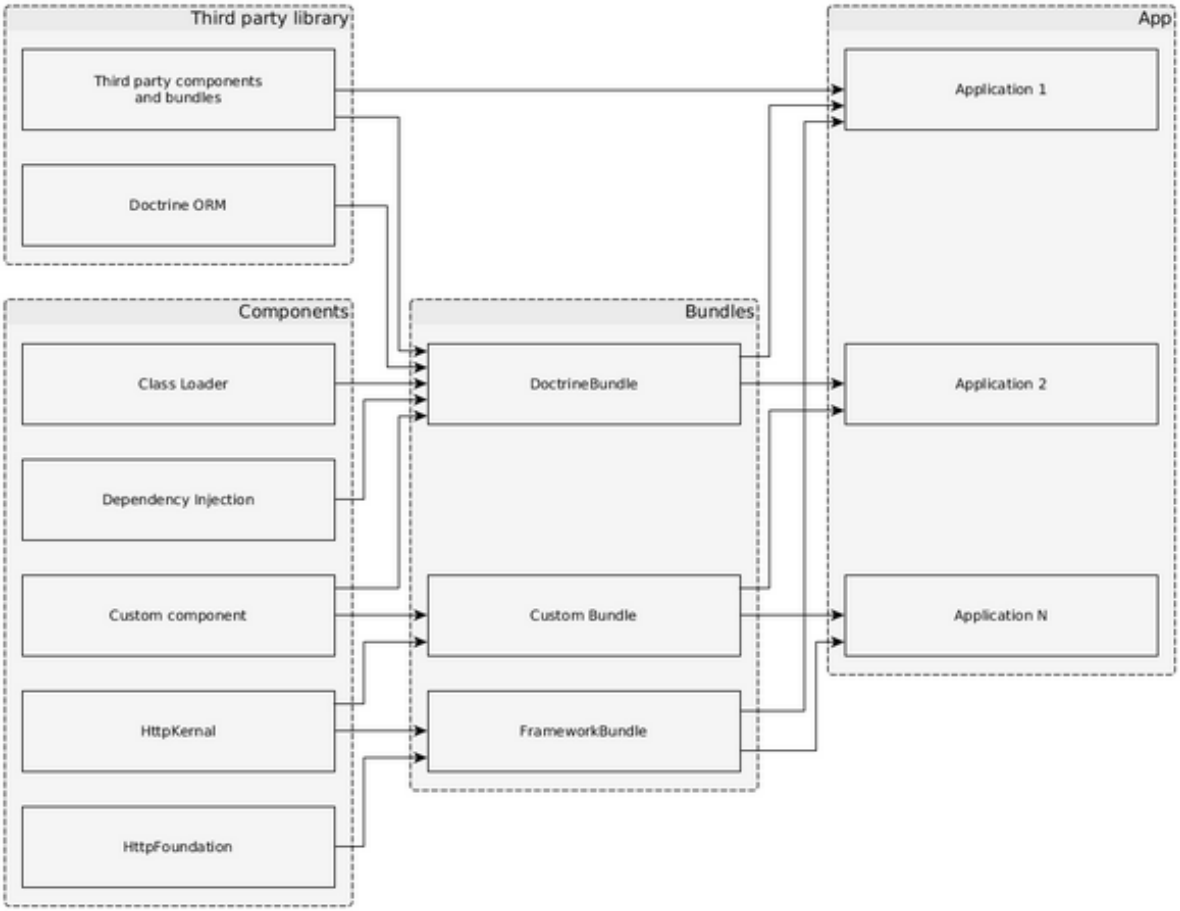
tutorialspoint
SIMPLYEASYLEARNING

3. Symfony – Architecture

Symfony is basically a collection of high quality components and bundles. Components are collection of classes providing a single core functionality. For example, **Cache component** provides cache functionality, which can be added to any application. Components are building blocks of a Symfony application. Symfony has 30+ high quality components, which are used in many PHP framework such as Laravel, Silex, etc.

Bundles are similar to plugin but easy to create and easy to use. Actually, a Symfony application is itself a bundle composed of other bundles. A single bundle can use any number of Symfony component and also third-party components to provide features such as Webframework, database access, etc. Symfony core web-framework is a bundle called FrameworkBundle and there is a bundle called FrameworkExtraBundle, which provides more sophisticated options to write a web application.

The relationship between the Components, Bundles, and Symfony application is specified in the following diagram.



tutorialspoint
SIMPLYEASYLEARNING

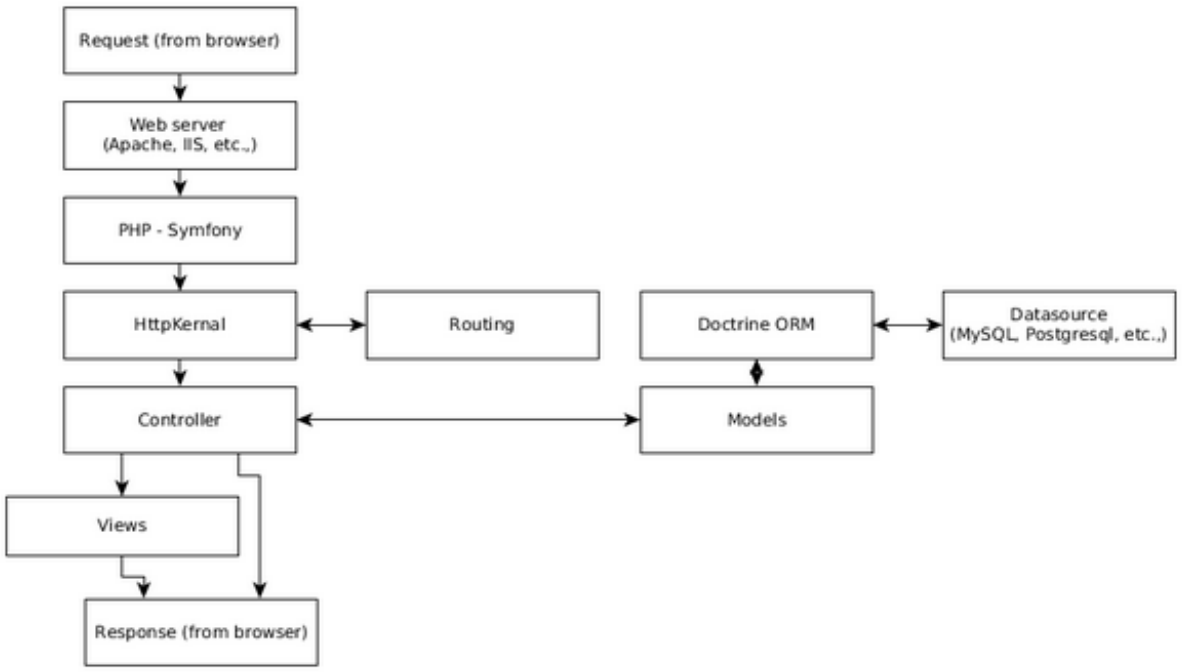
Web Framework

Symfony is mainly designed to write high-quality web application with relative ease. It provides various options to write different types of web application from simple web site to advanced REST based web services. Symfony provides web framework as separate bundles. The common bundles used in Symfony web framework are as follows:

- FrameworkBundle
- FrameworkExtraBundle
- DoctrineBundle

Symfony web framework is based on Model-View-Controller (MVC) architecture. **Model** represents the structure of our business entities. **View** shows the models to the user in the best possible way depending on the situation. **Controller** handles all the request from the user, does the actual work by interacting with Model and finally provides the View with the necessary data to show it to the user.

Symfony web framework provides all the high-level features required for an enterprise-grade application. Following is a simple workflow of Symfony web application.



The workflow consists of the following steps.

Step 1: The user sends a request to the application through the browser, say http://www.symfonyexample.com/index.

Step 2: The browser will send a request to the web server, say Apache web server.

Step 3: The web server will forward the request to the underlying PHP, which in turn sends it to Symfony web framework.

tutorialspoint
SIMPLYEASYLEARNING

Step 4: HttpKernel is the core component of the Symfony web framework. HttpKernel resolves the controller of the given request using Routing component and forward the request to the target controller.

Step 5: All the business logic takes place in the target controller.

Step 6: The controller will interact with Model, which in turn interacts with Datasource through Doctrine ORM.

Step 7: Once the controller completes the process, it either generates the response itself or through View Engine, and sends it back to the web server.

Step 8: Finally, the response will be sent to the requested browser by the web server.

4. Symfony – Components

As discussed earlier, Symfony components are standalone PHP library providing a specific feature, which can be used in any PHP application. Useful new components are being introduced in each and every release of Symfony. Currently, there are 30+ high quality components in Symfony framework. Let us learn about the usage of Symfony components in this chapter.

Installing a Symfony Component

Symfony components can be installed easily using the composer command. Following generic command can be used to install any Symfony component.

```
cd /path/to/project/dir
composer require symfony/<component_name>
```

Let us create a simple php application and try to install **Filesystem** component.

Step 1: Create a folder for the application, **filesystem-example**

```
cd /path/to/dev/folder
mdkir filesystem-example
cd filesystem-example
```

Step 2: Install Filesystem component using the following command.

```
composer require symfony/filesystem
```

Step 3: Create a file **main.php** and enter the following code.

```
<?php
require_once __DIR__ . '/vendor/autoload.php';
use Symfony\Component\Filesystem\Filesystem;
use Symfony\Component\Filesystem\Exception\IOExceptionInterface;
$fs = new Filesystem();
try {
      $fs->mkdir('./sample-dir');
      $fs->touch('./sample-dir/text.txt');
    } catch (IOExceptionInterface $e) {
    echo $e;
  }
?>
```

The first line is very important, which loads all the necessary classes from all the components installed using the Composer command. The next lines use the Filesystem class.

Step 4: Run the application using the following command and it will create a new folder **sample-dir** and a file **test.txt** under it.

```
php main.php
```

Details of Symfony Components

Symfony provides components ranging from simple feature, say file system to advanced feature, say events, container technology, and dependency injection. Let us know about all the components one by one in the following sections.

Filesystem

Filesystem component provides a basic system command related to files and directories such as file creation, folder creation, file existence, etc. Filesystem component can be installed using the following command.

```
composer require symfony/filesystem
```

Finder

Finder component provides fluent classes to find files and directories in a specified path. It provides an easy way to iterate over the files in a path. Finder component can be installed using the following command.

```
composer require symfony/finder
```

Console

Console component provides various options to easily create commands, which can be executed in a terminal. Symfony uses the **Command** component extensively to provide various functionalities such as creating a new application, creating a bundle, etc. Even the PHP build in web server can be invoked using Symfony command, **php bin/console server:run** as seen in the installation section. The **Console** component can be installed using the following command.

```
composer require symfony/console
```

Let us create a simple application and create a command, **HelloCommand** using the **Console** component and invoke it.

Step 1: Create a project using the following command.

```
cd /path/to/project
composer require symfony/console
```


tutorialspoint
SIMPLYEASYLEARNING

Step 2: Create a file **main.php** and include the following code.

```
<?php
require __DIR__ . '/vendor/autoload.php';
use Symfony\Component\Console\Application;
$app = new Application();
$app->run();
?>
```

Application class sets up the necessary functionality of a bare-bone console application.

Step 3: Run the application, **php main.php**, which will produce the following result.

```
  Console Tool

  Usage:
      command [options] [arguments]

  Options:
      -h, --help            Display this help message
      -q, --quiet           Do not output any message
      -V, --version         Display this application version
              --ansi            Force ANSI output
              --no-ansi         Disable ANSI output
      -n, --no-interaction  Do not ask any interactive question
      -v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for normal
output, 2 for more verbose output and 3 for debug

  Available commands:
      help  Displays help for a command
      list  Lists commands
```

Step 4: Create a class called **HelloCommand** extending **Command** class in the **main.php** itself.

```
  use Symfony\Component\Console\Command\Command;
  use Symfony\Component\Console\Input\InputInterface;
  use Symfony\Component\Console\Output\OutputInterface;
  use Symfony\Component\Console\Input\InputArgument;
```


tutorialspoint
SIMPLYEASYLEARNING

```
class HelloCommand extends Command
{
}
```

The application uses following four classes available in **Command** component.

- **Command** - Used to create a new command
- **InputInterface** - Used to set user inputs
- **InputArgument** - Used to get user inputs
- **OutputInterface** - Used to print output to the console

Step 5: Create a function **configure()** and set name, description, and help text.

```
protected function configure()
{
    $this
        ->setName('app:hello')
        ->setDescription('Sample command, hello')
        ->setHelp('This command is a sample command')
}
```

Step 6: Create an input argument, **user** for the command and set as mandatory.

```
protected function configure()
{
    $this
        ->setName('app:hello')
        ->setDescription('Sample command, hello')
        ->setHelp('This command is a sample command')
        ->addArgument('name', InputArgument::REQUIRED, 'name of the user');
}
```

Step 7: Create a function **execute()** with two arguments **InputArgument** and **OutputArgument**.

```
protected function execute(InputInterface $input, OutputInterface $output)
{
}
```


tutorialspoint
SIMPLYEASYLEARNING

Step 8: Use **InputArgument** to get the user details entered by the user and print it to the console using **OutputArgument**.

```
protected function execute(InputInterface $input, OutputInterface $output)
{
    $name = $input->getArgument('name');
    $output->writeln('Hello, ' . $name);
}
```

Step 9: Register the **HelloCommand** into the application using the **add** method of **Application** class.

```
$app->add(new HelloCommand());
```

The complete application is as follows.

```
<?php
require __DIR__ . '/vendor/autoload.php';
use Symfony\Component\Console\Application;
use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;
use Symfony\Component\Console\Input\InputArgument;

class HelloCommand extends Command
{
    protected function configure()
    {
        $this
            ->setName('app:hello')
            ->setDescription('Sample command, hello')
            ->setHelp('This command is a sample command')
            ->addArgument('name', InputArgument::REQUIRED, 'name of the user');
    }

    protected function execute(InputInterface $input, OutputInterface $output)
    {
        $name = $input->getArgument('name');
        $output->writeln('Hello, ' . $name);
```


tutorialspoint
SIMPLYEASYLEARNING

```
    }
}

$app = new Application();
$app->add(new HelloCommand());
$app->run();
?>
```

Step 10: Now, execute the application using the following command and the result will be Hello, Jon as expected.

```
php main.php app:hello Jon
```

Symfony comes with a pre-built binary called **console** in the bin directory of any Symfony web application, which can be used to invoke the commands in an application.

tutorialspoint
SIMPLYEASYLEARNING

End of ebook preview

If you liked what you saw…

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

tutorialspoint
SIMPLYEASYLEARNING