Introduction

Symfony2_ is a reusable set of standalone, decoupled, and cohesive PHP components that solve common web development problems.

Instead of using these low-level components, you can use the ready-to-be-used Symfony2 full-stack web framework, which is based on these components... or you can create your very own framework. This book is about the latter.

.. note::

If you just want to use the Symfony2 full-stack framework, you'd better
read its official `documentation`_ instead.

Why would you like to create your own framework?

Why would you like to create your own framework in the first place? If you look around, everybody will tell you that it's a bad thing to reinvent the wheel and that you'd better choose an existing framework and forget about creating your own altogether. Most of the time, they are right but I can think of a few good reasons to start creating your own framework:

- To learn more about the low level architecture of modern web frameworks in general and about the Symfony2 full-stack framework internals in particular;
- To create a framework tailored to your very specific needs (just be sure first that your needs are really specific);
- To experiment creating a framework for fun (in a learn-and-throw-away approach);
- To refactor an old/existing application that needs a good dose of recent web development best practices;
- To prove the world that you can actually create a framework on your own (... but with little effort).

I will gently guide you through the creation of a web framework, one step at a time. At each step, you will have a fully-working framework that you can use as is or as a start for your very own. We will start with simple frameworks and more features will be added with time. Eventually, you will have a fully-featured full-stack web framework.

And of course, each step will be the occasion to learn more about some of the Symfony2 Components.

.. tip::

If you don't have time to read the whole book, or if you want to get
started fast, you can also have a look at `Silex`_, a micro-framework
based on the Symfony2 Components. The code is rather slim and it leverages
many aspects of the Symfony2 Components.

Many modern web frameworks advertize themselves as being MVC frameworks. We won't talk

about the MVC pattern as the Symfony2 Components are able to create any type of frameworks, not just the ones that follow the MVC architecture. Anyway, if you have a look at the MVC semantics, this book is about how to create the Controller part of a framework. For the Model and the View, it really depends on your personal taste and I will let you use any existing third-party libraries (Doctrine, Propel, or plain-old PDO for the Model; PHP or Twig for the View).

When creating a framework, following the MVC pattern is not the right goal. The main goal should be the **Separation of Concerns**; I actually think that this is the only design pattern that you should really care about. The fundamental principles of the Symfony2 Components are focused on the HTTP specification. As such, the frameworks that we are going to create should be more accurately labelled as HTTP frameworks or Request/Response frameworks.

Before we start

Reading about how to create a framework is not enough. You will have to follow along and actually type all the examples we will work on. For that, you need a recent version of PHP (5.3.8 or later is good enough), a web server (like Apache or NGinx), a good knowledge of PHP and an understanding of Object Oriented programming.

Ready to go? Let's start.

Bootstrapping

Before we can even think of creating our first framework, we need to talk about some conventions: where we will store our code, how we will name our classes, how we will reference external dependencies, etc.

To store our framework, create a directory somewhere on your machine:

.. code-block:: sh

```
$ mkdir framework
$ cd framework
```

Dependency Management ~~~~~~~~~~~~~~~~~~~~

To install the Symfony2 Components that we need for our framework, we are going to use Composer_, a project dependency manager for PHP. If you don't have it yet,download and install_ Composer now:

.. code-block:: sh

```
$ curl -sS https://getcomposer.org/installer | php
```

Then, generate an empty composer.json file, where Composer will store the framework dependencies:

.. code-block:: sh

```
$ php composer.phar init -n
```

Our Project

Instead of creating our framework from scratch, we are going to write the same "application" over and over again, adding one abstraction at a time. Let's start with the simplest web application we can think of in PHP::

```
<?php

// framework/index.php

$input = $_GET['name'];

printf('Hello %s', $input);
```

Use the PHP built-in server to test this great application in a browser (http://localhost:4321/index.php?name=Fabien):

.. code-block:: sh

```
$ php -S 127.0.0.1:4321
```

In the next chapter, we are going to introduce the HttpFoundation Component and see what it brings us.

.. _Symfony2: http://symfony.com/ .. _documentation: http://symfony.com/doc .. _Silex: http://silex.sensiolabs.org/ .. _Composer: http://packagist.org/about-composer .. _download and install: https://getcomposer.org/doc/01-basic-usage.md