使用Symfony2的组件创建自己的PHP框架（第一部分：使用Composer初始化项目）

4条回复

英文原文地址：http://fabien.potencier.org/article/50/create-your-own-framework-on-top-of-the-symfony2-components-part-1

Sf2是一系列独立的，解耦的，可复用的，有粘着力的（cohesive）PHP组件库。可用来解决web开发中的常见问题。
您可以用完整的，使用这些组件的sf2框架来替代使用这些底层组件来做开发，亦或使用这些底层组件来创建你自己的框架。这篇文章要达成的是第二种目的。

为什么要创建自己的框架？

为什么要创建自己的框架呢？如果你跟周围的人讨论，每个人都会告诉你重复发明轮子是一件糟糕的事情，你最好选择一个已有的框架，忘掉“创建自己的框架”这种想法。大部分情况，他们是正确的，但是我想到了几个创建自己的框架的好处：

1. 了解更多框架的底层架构，特别是像sf2那种完整框架的内部架构
2. 创建一个能满足你特殊需求的框架（但首先要确定你的需求真的是很特别）
3. 因为乐趣而试着写一个框架（为了“学习然后抛弃”的目的）（译者注：躺枪）
4. 想利用新的开发技术以及最佳实践重构已经存在的项目
5. 向世界证明自己也是可以写出框架的（……但只需那么一点点付出）

我将一步步的，循序渐进的引导你创建一个框架。每一步你得到的都是一个完全能使用的框架。我们将从一个简单的框架开始，然后一点点的给它加功能。最后，你将能得到一个完整的web框架。

当然，每一步你都会学到一些新的sf2组件的用法。

如果你没有时间看完全文，或者你想快一点开始学习，你也可以看看*Silex*库——一个使用*sf2*组件写的微框架，它的代码简单，而且也体现出*sf2*组件的许多用法。

许多现代框架都称呼自己为“MVC框架”。在这里我们不讨论MVC，因为sf2组件可以用来创建任何类型的框架，而不仅仅是MVC架构的框架。总之，如果你看过MVC的定义，这个系列只讨论如何创建控制器部分。对

于数据模型部分以及视图部分，这得看你的个人爱好（译者注：意思是看你喜欢使用什么库或者什么方式来处理Model和View），对我而言，我会让你使用一些第三方的库文件（数据部分比如Doctrine，Propel或者直接使用PDO；视图部分比如Twig）。

当创建一个框架的时候，只是为遵循MVC模式并不是一个正确的目的。最主要的目的是要做到“分离关注点”（译者注：其实就是我经常说的框架的目的是为了分工，MVC就是一种分工的方式）。事实上这是我认为需要真正关心的，唯一的设计模式。sf2组件都是围绕实现http协议这个基本原则而创建的。同样，我们将要创建的框架应更准确的被描述为http框架或者说是“请求/响应框架”。

开始之前的准备

只学习如何创建框架是不够的，你最好按照我们写的例子自己实际操作一下。所以你需要一个较新的PHP（译者注：因为sf2是基于PHP5.3的所以最起码PHP的版本不能低于PHP5.3.0，最好是PHP5.3.8及其以上），一个web服务器比如Apache或者Nginx，对PHP有较好的掌握，并且对面向对象有所了解。

准备好了吗？那我们就开始吧！

引导程序（**Bootstrapping**）

在构思我们的框架之前，我们需要谈谈一些规定，比如我们在什么地方保存我们的代码，如何给类命名，如何引用对外部代码的依赖等等。

我们先建立一个目录来保存我们的代码：

```
1 $ mkdir framework
2 $ cd framework
3
```

代码规范

我们不讨论哪种代码规范更好，只要坚守同一种代码规范就行了，这个系列我们就使用Symfony2的编码规范。

安装组件

我们将使用Composer来安装我们需要的sf2组件，它是一个用PHP做的项目依赖管理器（译者注：类似Gentoo的emerge, Ubuntu的apt-get）。首先，将你要使用的组件用composer.json文件列出来：

```
1 {
2     "require": {
3         "symfony/class-loader": "2.1.*"
4     }
5 }
6
```

这里我们告诉Composer我们需要Symfony的class-loader组件，版本为2.1.0及其以上。要安装组件，我们需要下载composer二进制文件并且运行它（译者注：其实是一个php打包后的phar文件，直接用PHP来运行。关于phar文件可见我之前写的文章）：

```
1 $ wget http://getcomposer.org/composer.phar
2
```

或者

```
1 $ curl -O http://getcomposer.org/composer.phar
2
```

译者注：如果是*windows*，可先直接下载好再放进刚刚建立的目录里；如果用*cygwin*可以先安装*links*再使用*`links http://getcomposer.org/composer.phar`*下载

```
1 $ php composer.phar install
2
```

运行完install命令，你会看见目录下多了一个vendor目录，此目录包含了sf2的ClassLoader代码

虽然我们强烈建议你使用*Composer*来安装组件，但是你也可以直接使用*Git*来下载组件压缩包。这都您说了算。

译者注：如果你觉得*Composer*的速度太慢或者不稳定，可以尝试使用镜像，点此查看

命名规则和自动加载

我们将自动加载所有的类。如果不使用自动加载，你需要在使用某个类之前自己手工包含类定义文件。不过如果遵守了命名规则，我们可以让php来帮我们做这些痛苦的工作。

Sf2的类命名以及自动加载遵循de-facto标准，PSR-0。sf2提供了实现PSR-0标准的组件自动加载器（ClassLoader），大多数情况sf2自动加载器将能为你载入一个项目的所有类。

在autoload.php中建立一个自动加载器：

```
1  <?php
2
3  // framework/autoload.php
4  require_once __DIR__.'/vendor/symfony/class-loader/Symfony/Compone
5
6  use Symfony\Component\ClassLoader\UniversalClassLoader;
7
8  $loader = new UniversalClassLoader();
9  $loader->register();
10
```

现在你可以在命令行下运行autoload.php，它应该什么都不会做，但也不会报错

```
1  $ php autoload.php
2
```

*Sf*官网有关于自动加载器的更多的文档

Composer也为安装过的依赖组件创建自动载入器，你可以直接使用载入vendor/.composer/autoloader.php来替代sf的autoloader。

我们的项目

与其按自己想象瞎写一个框架，我们不如不断的改进我们的“应用程序”，每次改进都引入一个“概念”。让我们从最简单的一个程序开始：

```
1  <?php
2
3  $input = $_GET['name'];
```

```
4
5 printf('Hello %s', $_GET['name']);
6
```

这就是我们第一个系列的所有代码。下一章我们将介绍`HttpFoundation`组件，看它能给我们的开发带来什么改进。