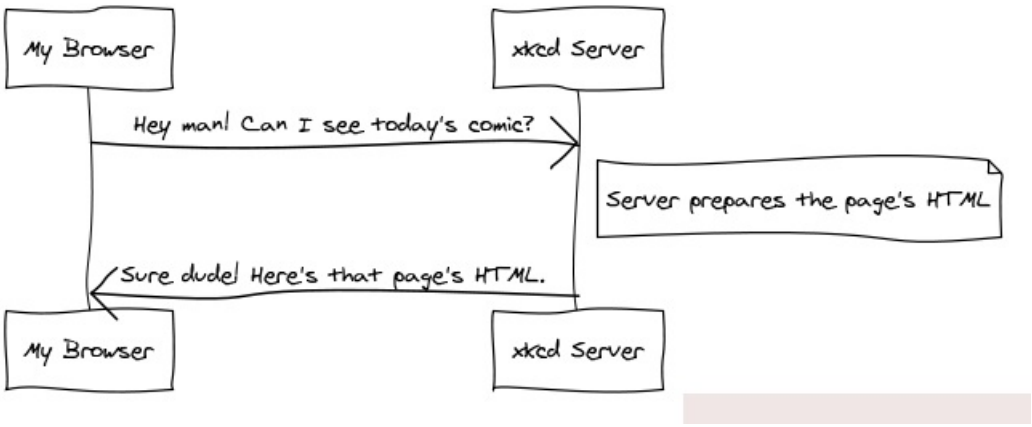
恭喜,通过前面几章对Symfony的学习,对于一名开发人员来说,你的开发方式已经变得更高效,更全面,更流行了,其实,你已经很优秀了!Symfony可以让你从最底层开始,开发更快,更强大的web应用程序,同时,你也可以保持自己的开发风格!Symfony的是建立在许多技术上的最好的框架,现在,你将要正式开始学习Symfony的核心内容了,这是多年来由成千上万人努力研发出的工具和概念，换一种说法，你不仅仅只是在学习“的Symfony”，你正在学习一些基本的网络知识，开发应用程序的实用技巧，以及如何使用许多建立Symfony在内部或独立的一些惊人的新的PHP类库的基本原理。所以，做好准备,让我们开始吧。

通过前面几章的学习,相信你已经完全入门了.现在正是该掌握Symfony原理的时候了.这一章将开始阐述Symfony用于web应用开发的基础概念HTTP.不管你有着如何深厚的编程背景或者你用什莫样的编程语言,这一章是你必须要看的!!

HTTP很简单

`HTTP`（`Hypertext Transfer Protocol to the geeks`超文本传输协议）是一种文本语言，允许两台机器间相互通信.例如，检查是否有最新的XKCD漫画的时候，下面的（近似）的对话将会发生：



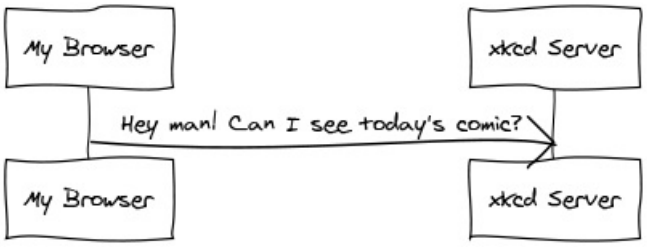
虽然实际编程操作的时候有些麻烦，但是它的协议相当简单。`HTTP`是用来描述这种简单的基于文本的语言术语。不管你在网络上如何开发，你的服务器的目标始终是理解简单的文本请求，并返回简单的文本响应。

Symfony的是来自世界各地的人们构建的。不管你是否意识到这一点，`HTTP`是你每天使用的东西。随着Symfony的学习，您将学习着去掌握`http`协议。

第一步：客户端发出请求

在网络上的每个对话都是以一个请求开始的。该请求是由客户端（例如浏览器，智能手机应用程序等）发起，用被称为`HTTP`一个特殊格式创建文本消息。客户端发送请求到服务器，然后再等待服务器响应。

看看一个浏览器和XKCD Web服务器之间的交互（请求）的第一步：



在HTTP会话中，客户机发送的HTTP请求实际上会是这个样子：

```
GET / HTTP/1.1
Host: xkcd.com
Accept: text/html
User-Agent: Mozilla/5.0 (Macintosh)
```

这个简单的信息传达了客户端的一些必要的 信息。一个HTTP请求的第一行是最重要的，他包含两件事情：所述 `URI` 和 `HTTP` 请求方法。

`URI` （例如 `/` ， `/contact` ，等等）是用来标识客户端请求的资源服务器的地址。 `HTTP` 方法（例如 `GET` ）定义你想要请求的资源。 `HTTP` 方法是一种请求动作.并且定义了一些常用方法，以下是http请求的一些常用方法：

`GET`	从服务器检索资源
`POST`	在服务器上创建一个资源
`PUT`	更新服务器上的资源
`DELETE`	删除服务器上的资源

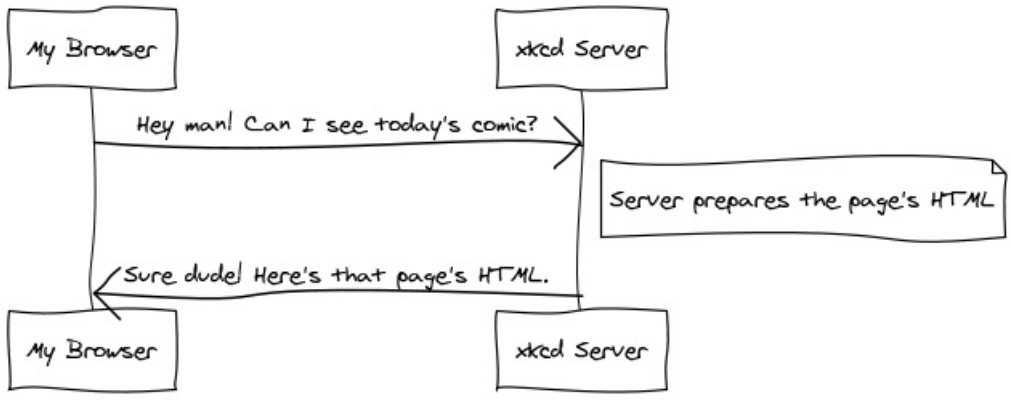
考虑到这一点，你可以想像一个 `HTTP` 请求可能看起来就像删除特定的博客条目一样，例如：

```
DELETE /blog/15 HTTP/1.1
```

除了第一行中，一个 `HTTP` 请求包含其他行被称为信息请求头部。头可以提供各种各样的信息，例如所请求的主机 `Host` ，客户机接受响应（ `Accept` ）的格式和应用程序客户端的信息(比如:用的什莫浏览器,是手机访问的还是电脑访问的)，也就是 `User-Agent` 。许多其他的头部已经存在，可以维基百科的文章List of HTTP header fields找到。

第二步：服务端返回响应

一旦服务器收到请求时，它通过 `URI` 知道到底是哪个客户端需要资源，这个客户机想要做什莫,请求的是哪一个资源（通过方法）。例如，在 `GET` 请求的情况下，服务器返回准备好的资源做HTTP响应。可考虑从XKCD Web服务器的响应：



翻译成 `HTTP` 的响应发送回浏览器时会是这个样子：

```
HTTP/1.1 200 OK
Date: Sat, 02 Apr 2011 21:05:05 GMT
Server: lighttpd/1.4.19
Content-Type: text/html

<html>
  <!-- ... HTML for the xkcd comic -->
</html>
```

关于http请求相关知识译者将不再累述,因为这是一套相当复杂,完善的一门技术,如果真要讲的话完全可以出一本书.在这里,译者向大家推荐一本书**《HTTP权威指南》**,在此给出该书的下载地址:

https://github.com/Pengfei-Gao/develop-reference-data/raw/master/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%BD%91%E7%BB%9C/%E3%80%8AHTTP%E6%9D%83%E5%A8%81%E6%8C%87%E5%8D%97%E3%80%8B%E4%B8%AD%E6%96%87%E7%89%88.pdf

PHP里的请求和响应

那莫如何使用php来处理和响应http请求呢?实际上,php已经将这些方法完全封装好了 .示例如下:

```php
$uri = $_SERVER['REQUEST_URI'];
$foo = $_GET['foo'];

header('Content-Type: text/html');
echo 'The URI requested is: '.$uri;
echo 'The value of the "foo" parameter is: '.$foo;
```

这乍看起来好像是相当古怪，这个小应用程序实际上是从HTTP请求中获取信息，并用它来创建一个HTTP响应。相反也能解析原生的 `HTTP` 请求，PHP准备了超全局准备变量，如 `$_SERVER` 和 `$_GET` 包含了来自请求中的所有信息。同样地,如果你不仅仅想返回HTTP格式的文本响应，你可以使用 `header()` 函数来创建HTTP响应头，将它打印出来。在上述的示例中, PHP将创建一个真正的HTTP响应，并将其返回给客户端：

```
HTTP/1.1 200 OK
Date: Sat, 03 Apr 2011 02:14:33 GMT
Server: Apache/2.2.17 (Unix)
Content-Type: text/html

The URI requested is: /testing?foo=symfony
The value of the "foo" parameter is: symfony
```

Symfony里的请求和响应

Symfony的提供了两个类，允许你使用更简单的方法来发送 `HTTP` 请求和响应请求,你完全可以用这两个类来替代原生的PHP的方法。 `Request` 类是HTTP请求消息的简单的面向对象的封装。有了它，你就有了所有的用户请求信息：

```php
use Symfony\Component\HttpFoundation\Request;

$request = Request::createFromGlobals();

// the URI being requested (e.g. /about) minus any query parameters
$request->getPathInfo();

// retrieve GET and POST variables respectively
$request->query->get('foo');
$request->request->get('bar', 'default value if bar does not exist');

// retrieve SERVER variables
$request->server->get('HTTP_HOST');

// retrieves an instance of UploadedFile identified by foo
$request->files->get('foo');

// retrieve a COOKIE value
$request->cookies->get('PHPSESSID');

// retrieve an HTTP request header, with normalized, lowercase keys
$request->headers->get('host');
$request->headers->get('content_type');

$request->getMethod();    // GET, POST, PUT, DELETE, HEAD
$request->getLanguages(); // an array of languages the client accepts
```

作为奖励， `Request` 类做了很多工作，你永远不必担心你是否有工作背景。例如， `isSecure()` 方法检查在PHP三个不同的值，可以表示用户是否通过安全连接（即，HTTPS）。

Symfony的还提供了一个 `Response` 类：一个HTTP响应消息的一个简单的PHP表示。这使你的应用程序使用面向对象的接口来构造需要被返回到客户端的响应信息：

```php
use Symfony\Component\HttpFoundation\Response;

$response = new Response();

$response->setContent('<html><body><h1>Hello world!</h1></body></html>');
$response->setStatusCode(Response::HTTP_OK);
$response->headers->set('Content-Type', 'text/html');

// prints the HTTP headers followed by the content
$response->send();
```

现在，你已经有了一个来方便地访问请求信息和创建响应的面向对象的接口工具包。即使你已经学习了Symfony了许多强大的功能，但是请记住，你的应用程序的目标始终是解释请求，并创建一个基于适当的应用程序的逻辑响应。

从Request到Response的旅程

HTTP的请求和响应是非常简单的。构建应用程序的困难的部分是写什莫，也就是软件功能。换句话说，真正的工作是解释请求信息，并创建响应代码。

您的应用程序可能要做很多事情，像发送电子邮件，处理表单提交，存储数据到数据库，呈现HTML页面,内容保护以及站点的安全性。如何管理这一切，仍然可以保持代码的组织性和可维护性？

Symfony的成立为了解决这些问题，这样你就不必为这些事忧心了。

前端控制器

传统上web应用程序的开发，一个站点的每个页面就是是它自己的物理文件：

```
index.php
contact.php
blog.php
```

通过这种方法,有几个问题就出现了，URL的灵活性相当差（你能只改变 `blog.php` 而不破坏你在 `news.php` 的所有链接吗？）,事实上，每个文件必须手动包含, 有一套核心文件使之变得更加安全，数据库连接和该网站看起来可以保持一致。

一个更好的解决方案是使用一个前端控制器：处理每一个请求进入您的应用程序一个PHP文件。例如：

`/index.php`	executes `index.php`
`/index.php/contact`	executes `index.php`
`/index.php/blog`	executes `index.php`

可以使用使用Apache的 `mod_rewrite` （或其他Web服务器），该网址可以很容易地映射到是 `/` ， `/contact` 和 `/blog` 。

现在，每个请求的处理方式完全相同。相反，可以使用单独的URL来执行不同的PHP文件，前端控制器总是在执行的，应用程序的不同部分的URL路由在内部完成。这就解决了原来做法的问题。几乎所有的现代Web应用程序做到了这一点 - 包括像 `WordPress` 应用程序。

保持条理清晰

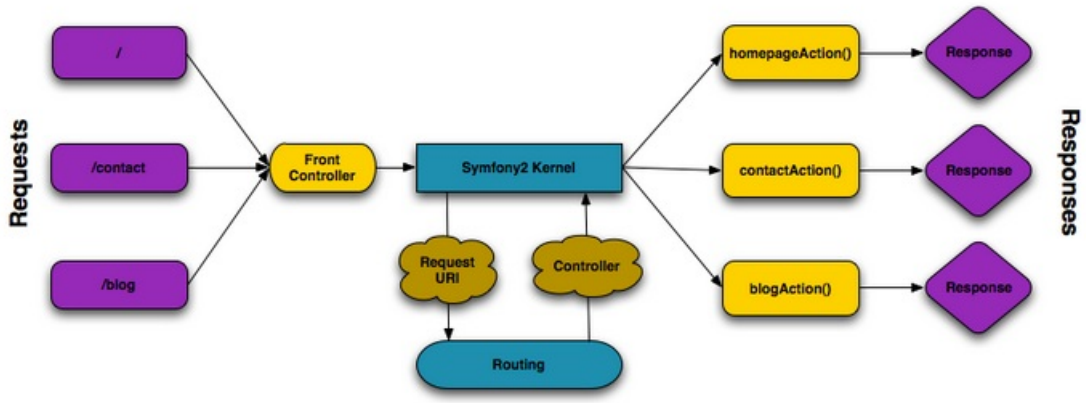
在你的前端控制器，你要弄清楚哪些代码应该被执行用来响应请求。为了弄清楚这一点，你需要检查传入URI以及执行取决于不同的值的　代码部分。这很快就让代码结构变得丑陋了：

```php
// index.php
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;

$request = Request::createFromGlobals();
$path = $request->getPathInfo(); // the URI path being requested

if (in_array($path, array('', '/'))) {
    $response = new Response('Welcome to the homepage.');
} elseif ('/contact' === $path) {
    $response = new Response('Contact us');
} else {
    $response = new Response('Page not found.', Response::HTTP_NOT_FOUND);
}
$response->send();
```

解决这个问题是很困难的。幸运的是这正是Symfony的特长。

Symfony应用的运行流程

当你让Symfony的处理每个请求时，程序流程要容易得多了。 Symfony为每个请求遵循相同的简单模式：



传入请求被由路由解释后,由路由传递到控制器,而后控制器返回 `Response` 对象。

你的网站的每一个页面被定义在路由配置文件里面,这样一来,就可以将不同的URL映射到不同的php函数去处理。这种PHP函数的工作，就被称为控制器，它使用的信息来自客户端的请求，可以用symfony的许多其他工具从请求得到的信息创建并返回一个Response对象。换句话说，控制器是你要编写的代码：你可以在这里解析客户端的请求,并按照你的理解创建响应。

就是这麽简单,回顾一下:

- 每个请求执行一个前端控制器文件;
- 路由系统决定哪一个php控制器函数会被执行,这个取决于客户端的请求以及你本地的路由配置
- 正确的PHP函数被执行时，你的代码将创建并返回相应的 `Response` 对象。

Symfony请求处理示例

无需深入到太多的细节，下面这是一个动作处理示例。假设你想在 `/contact` 页面添加到您的Symfony应用程序。首先，为 `/contact` 在路由配置文件中添加条目：

- *YAML*

```yaml
# app/config/routing.yml
contact:
    path:     /contact
    defaults: { _controller: AppBundle:Main:contact }
```

- *XML*

```xml
<!-- app/config/routing.xml -->
<?xml version="1.0" encoding="UTF-8" ?>
<routes xmlns="http://symfony.com/schema/routing"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://symfony.com/schema/routing
        http://symfony.com/schema/routing/routing-1.0.xsd">

    <route id="contact" path="/contact">
        <default key="_controller">AppBundle:Main:contact</default>
    </route>
</routes>
```

- *PHP*

```php
// app/config/routing.php
use Symfony\Component\Routing\Route;
use Symfony\Component\Routing\RouteCollection;

$collection = new RouteCollection();
$collection->add('contact', new Route('/contact', array(
    '_controller' => 'AppBundle:Main:contact',
)));

return $collection;
```

这时,当有人访问 `/contact` 页面时,这个路由将会被匹配到,一个在配置文件中对应的控制器将会被执行.当我们在后面几章专门讲解关于路由的相关知识时, `AppBundle:Main:contact` 这些字串是一种指向 `MainController` 类中的方法 `contactAction` 的缩写:

```php
// src/AppBundle/Controller/MainController.php
namespace AppBundle\Controller;

use Symfony\Component\HttpFoundation\Response;

class MainController
{
    public function contactAction()
    {
        return new Response('<h1>Contact us!</h1>');
    }
}
```

在这个非常简单的例子，控制器只需创建一个与HTML(`<h1>Contact us!</h1>`) `Response` 对象！在后面的控制器章节中，你将学会如何将你编写用于“表现(`presentation)` ”的代码（即任何实际写出HTML）存放在在一个单独的模板文件,并且使用控制器渲染模板。这样就减少了控制器的后顾之忧,剩下的就只有编些功能实现了：与数据库交互，处理提交的数据，或发送电子邮件。

Symfony：专注项目本身，而不是工具

现在你知道的任何应用程序的目标是解释每一个请求，并创建相应的响应。一个应用程序的功能会随着业务需求而不断的增长，它将变得更加难以保持代码的组织性和可维护性。不变的是，同 一个复杂的任务，不断地来了一遍又一遍：存储数据到数据库，渲染和重用模板，处理表单提交，发送电子邮件，验证用户输入和处理站点的安全性。

好消息是，这些问题并不是唯一的。 Symfony提供了一个完整的工具，使你可以构建应用程序，而不是你的工具框架。随着Symfony的，没有什么是强加给你：你可以自由使用完整 的Symfony框架，或者他的一部分功能。

独立的工具箱: Symfony 组件

那么，什么是Symfony的？首先，Symfony是可以在任何PHP项目中使用了二十独立库的集合。这些库，称为Symfony的组件，包含一些有用的东西，在任何情况下，不管如何你的项目开发。仅举几例：

- HttpFoundation
- Routing
- Form
- Validator
- Templating
- Security
- Translation

全方位解决方案: Symfony 框架

那么，是什么Symfony框架？ Symfony框架是一个PHP库，完成两个不同的任务：

1. 提供了一个选择的组件（即Symfony的组件）和第三方库（例如斯威夫特梅勒发送电子邮件）；
2. 提供合理的配置和“胶水”库来将所有这些组件结合在一起。

框架的目标是整合许多独立的工具，为开发者提供一致的体验。即使框架本身是一个Symfony的包（即一个插件），但他可以被配置或完全替换。

Symfony的提供了一套强大的工具，用于快速开发Web应用程序，而不强加给你的应用程序。普通用户可以通过使用Symfony的分布，它提供了一个 项目骨架合理的默认快速启动开发。对于更高级的用户，天空才是极限。

我的主页:http://aifei8.net:

我的博客:http://blog.aifei8.net

github主页:http://github.com/Pengfei-Gao