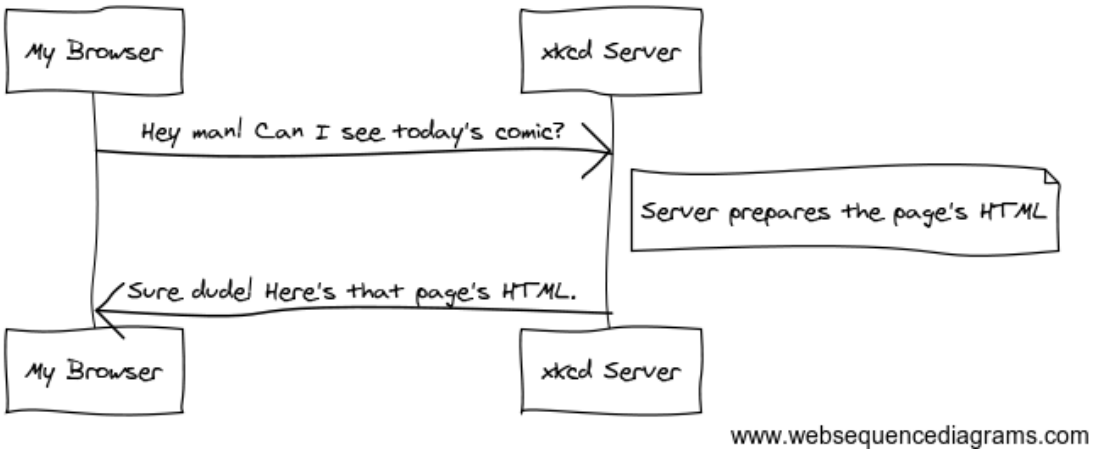
Symfony2 和 HTTP 基本原理

恭喜你！通过学习 Symfony2，你将成为一个更高效的、全面的和著名的 web 开发者（事实上，最后一条取决于你自己）。Symofny2 为了最基本的事情而存在：开发一些工具，让你以自己的方式开发出更快速、更健壮的应用。Symfony2 是在一些技术的最佳方式下构建起来的，你将学习到的工具和概念是成百上千的人们多年的经验成果。换句话说，你不只是在学习 Symfony，你在学习 web 的基础、开发的最佳实践以及如何在 Symfony 内或者之外，使用的很多令人兴趣的新的 PHP 类库。所以，开始吧。

为了了解 Symofny2 的本质，本章从解释 web 开发的基本概念 HTTP 协议开始。无论你有什么样的开发背景或者使用何种语言，这章对每个人来讲都是**必读**的。

HTTP 是如此简单

HTTP(超文本传输协议)是一种允许两台机器相互通讯的文本语言。就是这样。举个例子，当你去 xkcd 网查看最新的漫画时，下面这些对话将会发生：



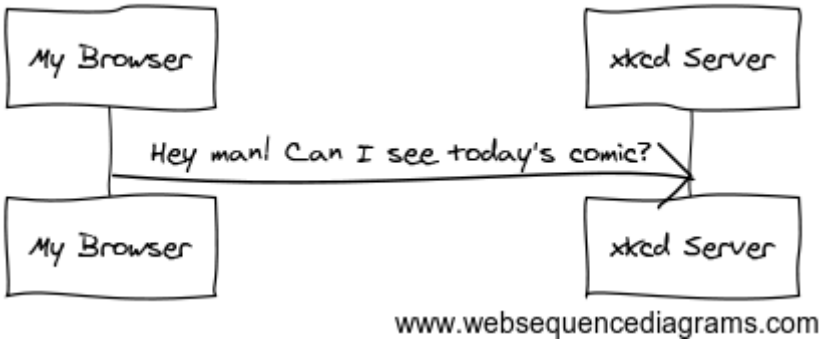
虽然上面的语言有点正式，但是仍然非常简单。HTTP 就是用来描述这个简单的基于文本的语言。不管你如何在 web 上开发，你的目标服务器总是能理解简单的文本请求，并且返回

一个简单的文本相应。

第一步，客户端发送请求

web 上的每个会话都已一个请求开始。请求就是一个客户端（比如浏览器、iphone app）创建的一种 HTTP 协议知道的文本消息。客户端发送这个请求到服务器，然后等待服务的响应。

看看浏览器和 xkcd web 服务器交互的第一部分：



在 HTTP 会话中，这个 HTTP 请求可以被看成这样：

```
1. GET / HTTP/1.1
2. Host: xkcd.com
3. Accept: text/html
4. User-Agent: Mozilla/5.0
```

这个简单的消息传递了关于客户端请求哪个资源的必备条件。HTTP 请求中的第一行是最重要的，他包含了两个信息：uri 和 http 方法。

上面例子中，uri(比如"/","/contact"等)是定义客户端需要的资源的唯一地址。其中，HTTP 方法（比如 get）定义了你想对该资源做什么。HTTP 方法是这个请求的动作，并且定义了你可以操作资源的几种常见的方式：

GET	从服务器检索资源
POST	在服务器上创建资源
PUT	更新服务器上的资源

DELETE	删除服务器上的资源

记住这些，你可以想象的到删除特定的某个博客的请求应该像这样：

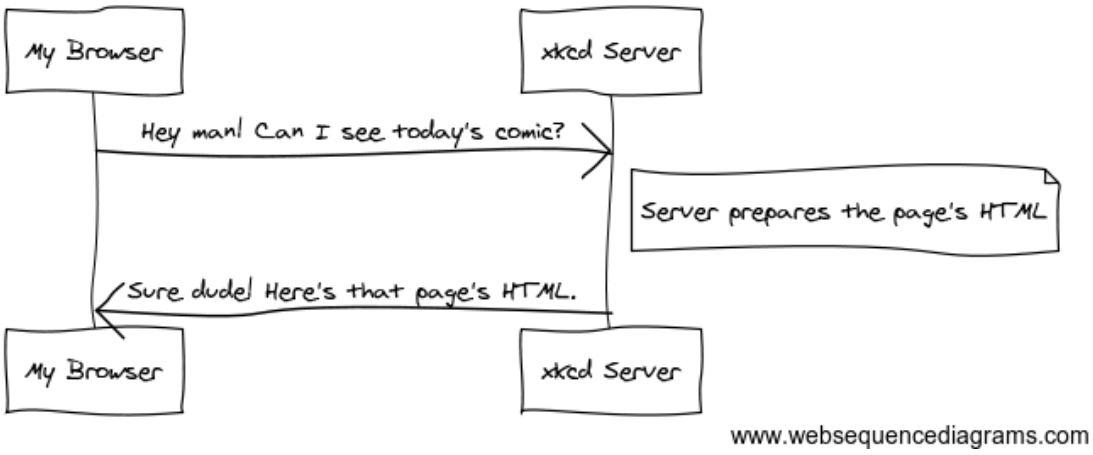
```
1.  DELETE  /BLOG/15  HTTP/1.1
```

事实上，HTTP 一共定义了 9 种方法，但是他们中多数不是被广泛使用或者被支持。事实上，很多现代的浏览器也不支持 PUT 和 DELETE 方法。

除了第一行之外，一个 HTTP 协议重视包含其他行的信息，被称之为请求头。请求头可以支持很多的信息，比如请求的 host，客户端接收的相应格式，以及客户端用来发送请求的应用程序（User-Agent）。一些其他的请求头可以在维基百科 List of HTTP header fields 上找到。

第二步，服务器返回响应

一旦服务器接收到请求，它就知道客户端需要哪个资源（通过 uri），以及客户端希望如何操作该资源（通过请求方法）。比如，在一个 get 方式的请求中，服务器准备资源，然后在一个 http 响应中返回它。想象下从 xkcd 服务器上来的响应：



转义成 HTTP，发送回浏览器的响应看起来像这样：

```
1.  HTTP/1.1   200   OK
2.  Date:   Sat, 02 Apr 2011 21:05:05 GMT
3.  Server:   lighttpd/1.4.19
4.  Content-Type:   text/html
5.
6.  <html>
7.      <!-- ... HTML for the xkcd comic -->
8.  </html>
```

http 响应中包含了请求的资源（该例子中就是 html 内容），同时还有其他信息。第一行是特别重要的，包含了 HTTP 响应的状态码（例子中为 200）。这个状态码表明，请求的全部结果返回到了客户端。那么，这个请求是否成功了？是否发生了错误？不同的状态码表示不同的结果，成功，错误，或者客户端需要做一些事（比如跳转到其他页面）。完整的列表可以在维基百科的文章中阅读 List of common media types。

还有很多其他的响应头存在，他们中有些能力非常大。比如说有些头可以用来构建一个强大的缓存系统。

请求、响应和 web 开发

请求-响应交互是驱动 web 上所有会话的基础过程。该过程是如此的重要和强大，但是缺非常简单。

最重要的事实就是：不管你用何种语言，你构建的应用程序的类型（web，手机，或者 json api），或者你遵循的开发理论，应用程序的最终目标是理解每个请求并且创建和返回适当的响应。

Symfony 一开始就是为了遵循这个事实而构建的。

想要了解更多关于 HTTP 规范，可以阅读最基本的 HTTP 1.1 RFC 或者 HTTP Bis，他们是阐述 HTTP 最基本概念的有效文档。一个用来在浏览器中检查请求和响应头的好工具是 Live HTTP Headers，这是一个火狐浏览器的插件。

PHP 中的请求和响应

你在 php 中是如何处理请求和创建一个响应的呢？事实上，PHP 封装了这整个过程：

```
1.  $uri  =  $_SERVER['REQUEST_URI'];
2.  $foo  =  $_GET['foo'];
3.
4.  header('Content-type: text/html');
5.  echo  'The URI requested is: '  .  $uri;
6.  echo  'The value of the "foo" parameter is: '  .  $foo;
```

这听起来有点奇怪，事实上这个小应用从 HTTP 请求中提取信息，然后使用它创建了一个 HTTP 响应。php 无需解析最原始的 HTTP 请求信息，它提供了一些超全局的变量比如：$_SERVER 和$_GET，他们包含了请求中的信息。同样的，你可以使用 header()函数创建一个响应头，并且输出响应中的一部分信息，而无需返回 HTTP 格式的文本响应。PHP 会创建真正的 HTTP 响应并且返回给客户端：

```
1.  HTTP/1.1  200  OK
2.  Date:  Sat, 03 Apr 2011 02:14:33 GMT
3.  Server:  Apache/2.2.17 (Unix)
4.  Content-Type:  text/html
5.
6.  The URI requested is:  /testing?foo=symfony
7.  The value of the "foo" parameter is:  symfony
```

Symfony 中的请求和响应

Symfony 提供了两个类用来处理请求和响应。Request 类是 HTTP 请求信息的面向对象的封装。通过它，你可以随意获取所有请求信息。

```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
```

好处就是 ,Request 类在底层做了大量的事情 ,而我们根本不需要去关心。比如说 ,isSecure() 方法通过检查 php 的 3 个变量去验证用户是否使用安全链接（比如 https ）。

ParameterBags 和 Request attributes

如上面所见，$_GET 和$_POST 变量分别可以通过 public 的 query 和 request 属性访问。他们都是一个 ParameterBag 对象，该对象有 get(),has(),All()等方法。事实上，上例中每个 public 的属性都是一个 ParameterBag 对象。

Request 类还有一个 public 的 attributes 属性，它保存了支撑应用内部运行的特殊数据。

对 Symfony2 框架来说，这个属性保存了所匹配的路由的信息，像_controller,id(如果有一个{id}通配符)，甚至所匹配的路由名（_route）。attributes 属性就是可以存储当前请求相关的特殊信息的地方。

Symfony 同时也提供了一个 Response 类，这是一个代表 HTTP 响应信息的简单的类。它允许你的应用使用面向对象的方式去构建返回给客户端的响应。

```
1.  use Symfony\Component\HttpFoundation\Response;
2.  $response = new Response();
3.
4.  $response->setContent('<html><body><h1>Hello
    world!</h1></body></html>');
5.  $response->setStatusCode(Response::HTTP_OK);
6.  $response->headers->set('Content-Type', 'text/html');
7.
8.  // prints the HTTP headers followed by the content
9.  $response->send();
```

v2.4 新功能：在 Symfony2.4 中添加了支持 HTTP 状态码的功能

如果除此之外 Symofny 没有提供其他工具，那么你已经拥有了一个可以简单的访问请求信息和用面向对象接口的方式创建响应的工具。哪怕之后你学习了更多的 Symfony 有用的特性，你始终要记住，你的应用的目标总是理解一个请求然后根据你的应用的逻辑创建一个适当的响应。

Request 和 Response 类是 Symofny 中的独立组件的一部分，我们称之为 HttpFoundation。这个组件可以独立于 Symfony 之外使用，并且它也提供了处理 session 和文件上传的类。

从请求到响应的旅程

就想 HTTP 协议本事，Request 和 Response 对象是非常简单的。构建应用程序的难点在于这两点之间的转换过程。换句话说，主要工作就是写解析请求信息然后创新响应这个过程的代码。

你的应用可能会做很多事，像发送邮件、处理表单提交、保存数据到数据库、渲染 html 页面和保护受保护的内容。你该如何处理任务的同时还要保证代码的组织性和可维护性呢？Symfony 就是为了解决这类问题而存在的。

前端控制器

传统中的应用创建中，我们尽量使每个页面逻辑在它自己的文件中：

1. index.php
2. contact.php
3. blog.php

这种方法有几个问题，包括没有弹性的 url（如果你想要把 blog.php 改名为 news.php，而又不想更改所有 url 要怎么办？）以及每个文件都要包含一些像安全、数据库连接和网站样式一致性的核心文件。

更好的解决办法是使用前端控制器，这是单独的一个文件却处理你的应用的所有请求，比如：

1. /index.php　executes index.php
2. /index.php/contact　executes index.php
3. /index.php/blog　executes index.php

使用 apache 的 mod_rewrite（或者其他 web 服务器的类型模块），这些 url 可以很简单的被转换成"/"，"/contact"，"/blog"。

现在每个请求都以同样的方式被处理。我们不再根据不同的 url 执行不同的文件，前端控制器总是被执行，不同 url 对应你的应用内部不同地方的路由映射在内部执行。这解决了前面说的俩个问题。绝大多数的 web 应用都采用这种方式，包括 wordpress。

代码的组织性

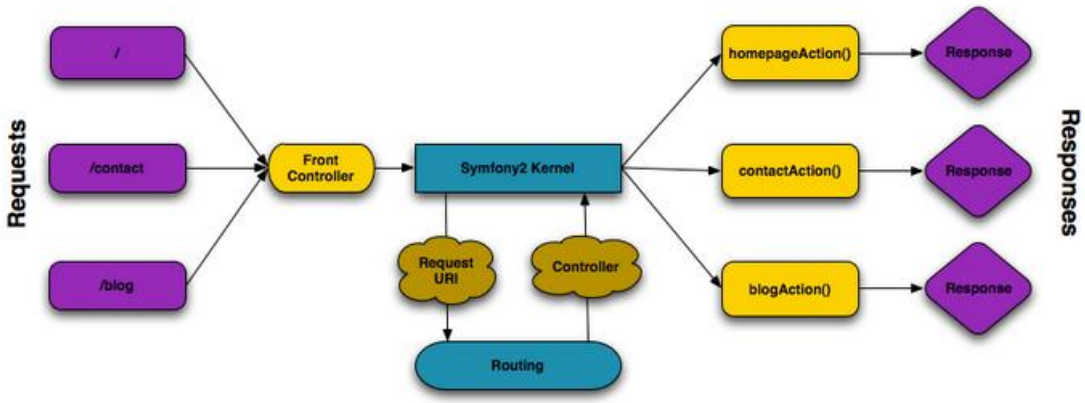
在前端控制器内部，你不得不指出哪些代码要被执行以及什么样的内容需要被返回。为了做到这步，你需要检查每个进来的 uri，然后根据它的值执行部同的代码。它可以简单粗暴的想这样：

```
1.  // index.php
2.  use Symfony\Component\HttpFoundation\Request;
3.  use Symfony\Component\HttpFoundation\Response;
4.
5.  $request = Request::createFromGlobals();
6.  $path = $request->getPathInfo(); // the URI path being requested
7.
8.  if (in_array($path, array('', '/'))) {
9.      $response = new Response('Welcome to the homepage.');
10. } elseif ($path == '/contact') {
11.     $response = new Response('Contact us');
12. else {
13.     $response = new Response('Page not found.',
    Response::HTTP_NOT_FOUND);
14. }
15. $response->send();
```

解决这个问题是很困难的。幸运的是，Symofny 就是被设计用来做这件事的。

Symfony 应用流程

当你让 Symfony 处理每个请求，整个生命期是很简单的。Symofny 按照下面的流程处理每个请求：



进入的请求被路由器拦截，然后传给控制器的方法，最终返回一个 Response 对象。

你网站中的每个页面都被定义在一个路由配置文件中，它负责映射不同的 url 到不同的 php 方法。这些 php 方法，我们称之为控制器，他们的作用是结合其他可用的 Symfony 工具使用从请求中获取的信息，创建和返回 response 对象。换句话说，控制器就是你写代码的地

方，你在这里获取请求、创建响应。

就是这么简单，总结一下：

- 每个请求都通过一个前端控制器文件执行
- 路由系统根据请求中的信息和你创建的路由配置决定哪个 php 方法被执行
- 正确的 php 方法被执行，在这个方法中写你的逻辑，然后返回适当的 response 对象

在动作中的 Symfony Request 对象

不要在意太多细节，下面是一个动作的基本过程。想象一下，你想要添加一个"/contact"

页面到你的 Symfony 项目中。首先，在你的路由配置中为"/contact"添加一个条目：

```
1.  # app/config/routing.yml
2.          contact:
3.                  path: /contact
4.                  defaults: { _controller: AcmeDemoBundle:Main:contact }
```

这个列子使用 YAML 文件定义路由配置。路由配置也可以用其他格式书写 ,比如 xml ,php。

当某人访问了"/contact"页面，这条路由被匹配，然后指定的控制器被执行。字符串

"AcmeDemoBundle:Main:contact"是一种简写的语法 ,它指向了 MainController 类中的

contactAction 这个方法，你可以通过 routing chapter 了解更多：

```
1.  // src/Acme/DemoBundle/Controller/MainController.php
2.  namespace Acme\DemoBundle\Controller;
3.
4.  use Symfony\Component\HttpFoundation\Response;
5.
6.  class MainController
7.  {
8.      public function contactAction()
9.      {
10.             return new Response('<h1>Contact us!</h1>');
11.     }
12. }
```

在这个简单的例子中，控制器创建了一个包含了`<h1>Contact us!</h1>`字符串的

Response 对象。在 controller chapter 你可以学到控制器可以渲染模板，允许表现层代码（任何可以输出 html 的）写在单独的模板文件中。不需要控制器去关心一些负责的任务，比如操作数据库、处理提交的数据或者发送邮件。

Symofny2：创建你的 app，而不是你的工具

现在你已经知道，任何 app 的目标都是处理每个进来的请求，然后创建一个恰当的响应。但是随着应用的增长，报纸代码的组织性和可维护性变的越来越难。一些类型的复杂任务总是再三的出现：持久化数据到数据库、渲染和重复利用模板、处理表单提交、发送邮件、验证用户输入和安全处理。

好消息是这些问题不是独特的。Symfony 提供了框架级的工具，这使的你可以构建你的应用程序，而不是你的工具。通过使用 Symfony2，没有任何东西是强加给你的，你可以自由的使用整个框架，或者使用其中的一部分。

独立的工具：Symofny2 组件

那么 Symfony2 是什么？首先，它是超过 20 个独立的、可用在任何 php 项目中的类库的集合。这些类库，叫做 Symfony2 组件，几乎适用于任何场景，而且不管你的项目是如何开发的。下面是他们中的一些：

- HttpFoundation - 包含了 Request 和 Response 类，同时还有处理 session 和文件上传的类
- Routing - 强大、快速的路由系统，允许你映射一个特定的 uri 到该请求该如何处理
- Form -一个功能全面、灵活的框架，用来创建表单和处理表单
- Validator - 一个对数据创建规则，然后验证用户提交的数据是否符合这些规则的

系统

- ClassLoad - 一个让 php 类不再需要手动 require 文件的自动载入类的库
- Templating - 一个渲染模板、处理模板继承和其他常见模板任务的工具箱
- Security - 一个强大的，用来处理应用内部所有安全类型问题的类库
- Translation - 在你的应用中翻译字符串的框架

这些组件中每一个都是解耦的，可以用在任何 php 项目中，不管你是否使用 Symfony2 框架。它们的每一个既可以在需要时使用，也可以在必要时被替换。

完整的解决方案：Symfony2 框架

那么，Symfony2 框架是什么？Symfony2 框架是一个完成下面两个不同任务的 php 库：

1. 提供选择性的组件和第三方的库
2. 提供简单的配置和黏性的库使得所有部分整合起来

框架的目标是为了给开发者提供相同的经验而整合很多独立的工具。即使框架本事就是一个 Symfony2 的 bundle，它任然可配置并且可以完全被替换掉。

Symfony2 提供了大量工具去加快 web 应用的开发，而不使你的应用臃肿。一般用户可以通过使用 Symfony2 发行版快速上手，它提供了一个项目的骨架和一些简单的默认配置。对于高级用户，天空才是极限。