# 理解REST架构

【参考：<http://www.ruanyifeng.com/blog/2011/09/restful.html】>

【参考：<https://www.zhihu.com/question/27785028/answer/48096396】>

【参考：<https://blog.csdn.net/qq_21383435/article/details/80032375】>

**开门见山：**

|  |
| --- |
| **URI定位资源，用HTTP动词（GET/POST/DELETE/PUT）描述操作。**  **URI是面向资源的，是名词，并不包含动词** |

## 前言：

越来越多的人意识到，**网站即软件**，这是一种新型的软件。

这种“互联网软件”采用“客户端/服务器”模式，建立在分布式体系下，通过互联网通信，具有高延时、高并发等特点。

网站开发，完全可以采用软件开发的模式。但是传统上，软件和网络是两个不同的领域，很少有交集：软件开发主要针对软件设计的分类、设计方法的演化；网络研究主要关注系统之间通信行为的细节、如何改进特定通信机制的表现。

互联网的兴起，使得这两个领域开始融合，现在我们必须考虑：**如何开发在互联网环境中使用的软件。**

**RESTful架构，就是目前最流行的一种互联网软件架构。**

它结构清晰、符合标准、易于理解、拓展方便，所以得到越来越多网站的采用。

下面先来了解一下REST的具体含义。

## 二、REST起源：

REST这个词，是 [Roy Thomas Fielding](http://en.wikipedia.org/wiki/Roy_Fielding" \t "http://www.ruanyifeng.com/blog/2011/09/_blank) 在他2000年的博士论文中提出来的。

Fielding将他对互联网软件的架构原则，定名为REST，即：**Representational State Transfer**。

我对这个词组的翻译是“表现层状态转移”。

**如果一个架构符合REST原则，就称它为RESTful架构。**

要理解RESTful架构，最好的方法就是去理解Representational State Transfer 这个词组到底是什么意思。如果你把这个名称搞懂了，也就不难体会REST是一种什么样的设计。

## 三、资源（Resources）：

REST的名称“变现层状态转移”中，省略了主语，“表现层”其实指的是“资源”的变现层。

**所谓“资源”，就是网络上的一个实体，或者说是网络上的一个具体信息。**

它可以是一段文本、一张图片、一首歌曲、一种服务，总之就是一个具体的实在。你可以用一个URI（统一资源标识符）指向它，每个资源对应一个特定的URI。要获取这个资源，访问它的URI就可以，因此URI就成了每一个资源的地址或第一无二的标识符。

所谓“上网”，就是通过URI找到互联网上一系列的“资源”，并与之互动。

## 表现层（Representational）：

“资源”是一种信息实体，它可以有多种外在表现形式。

|  |
| --- |
| **我们把“资源”具体呈现出来的形式，叫做它的“表现层（Representational）”** |

比如，文本可以用txt格式表现，也可以用HTML格式、XML格式、JSON格式表现，甚至可以采用二进制格式；图片可以用JPG格式表现，也可以采用PNG格式表现。

|  |
| --- |
| **URI只表示资源的实体，不代表它的形式** |

严格的说，有些网址最后的“.html”后缀名是不需要的，因为这个后缀名属于格式，属于表现层范畴，而URI应该只代表“资源”的位置。它的具体表现形式，应该在HTTP请求的头信息中用Accept和Content-Type字段指定，这两个字段才是对“表现层”的描述。

## 状态转化（State Transfer）：

访问一个网站，就代表客户端和服务器的一个互动过程。在这个过程中，势必涉及到数据和状态的变化。

|  |
| --- |
| **互联网协议HTTP协议，是一个无状态协议。** |

这意味着，所有的状态都保存在服务器端。因此，如果客户端想要操作服务器，必须通过某种手段，让服务器发生“状态转化（State Transfer）”。而这种转化是建立在表现层之上的，所以就是“变现层状态转化”。

客户端用到的手段，只能是HTTP协议。具体来说，就是HTTP协议里面，四个表示操作方式的动词：

1. GET：获取资源
2. POST：新建资源（也可以用于更新资源）
3. PUT：更新资源
4. DELETE：删除资源

## 综述：

综合上面的解释，我们总结一下什么是RESTful架构：

1. 每一个URI代表一个资源；
2. 客户端和服务器之间，传递这种资源的某种表现层；
3. 客户端通过四个HTTP动词，对服务器端的资源进行操作，实现“表现层状态转化”。

## 误区：

RESTful架构有一些典型的设计误区

### 误区一：URI包含动词：

最常见的一种设计错误，就是URI包含动词。因为“资源”表示一种实体，所以应该是名词，URI不应该有动词，动词应该放在HTTP协议里。

举例来说，某个URI是 /posts/show/1，其中show是动词，这个URI就设计错了，正确的写法应该是 /posts/1，然后用GET表示show。

如果某些动作是HTTP动词表示不了的，你就可以把动作做成一种资源。比如网上汇款，从账户1向账户2汇款500元，错误的URI是： POST /accounts/1/transfer/500/to/2

正确的写法是把动词transfer改成名词transaction，资源不能是动词，但是可以是一种服务。

|  |
| --- |
| POST /transaction HTTP/1.1 Host: 127.0.0.1 　　 from=1&to=2&amount=500.00 |

### 误区二：URI中加入版本号：

|  |
| --- |
| http://www.example.com/app/1.0/foo  　　http://www.example.com/app/1.1/foo  　　http://www.example.com/app/2.0/foo |

因为不同的版本可以理解为同一种资源的不同表现形式，所以应该采用同一个URI。版本号可以在HTTP请求头信息中的Accept字段中进行区分

|  |
| --- |
| Accept: vnd.example-com.foo+json; version=1.0  　　Accept: vnd.example-com.foo+json; version=1.1  　　Accept: vnd.example-com.foo+json; version=2.0 |

## 八、注意点：

### 1、HTTP是根据REST原则设计出的架构：

不要以为是由了HTTP才有了REST，其实是：REST只是一种规范、一种原则，R.T.D根据REST原则设计出来HTTP。