



分类：[开发者手册](#)

上一篇：[Michael S.](#)

下一篇：[学习C语言的教材](#)

理解RESTful架构

作者：阮一峰

分享

日期：2011年9月12日

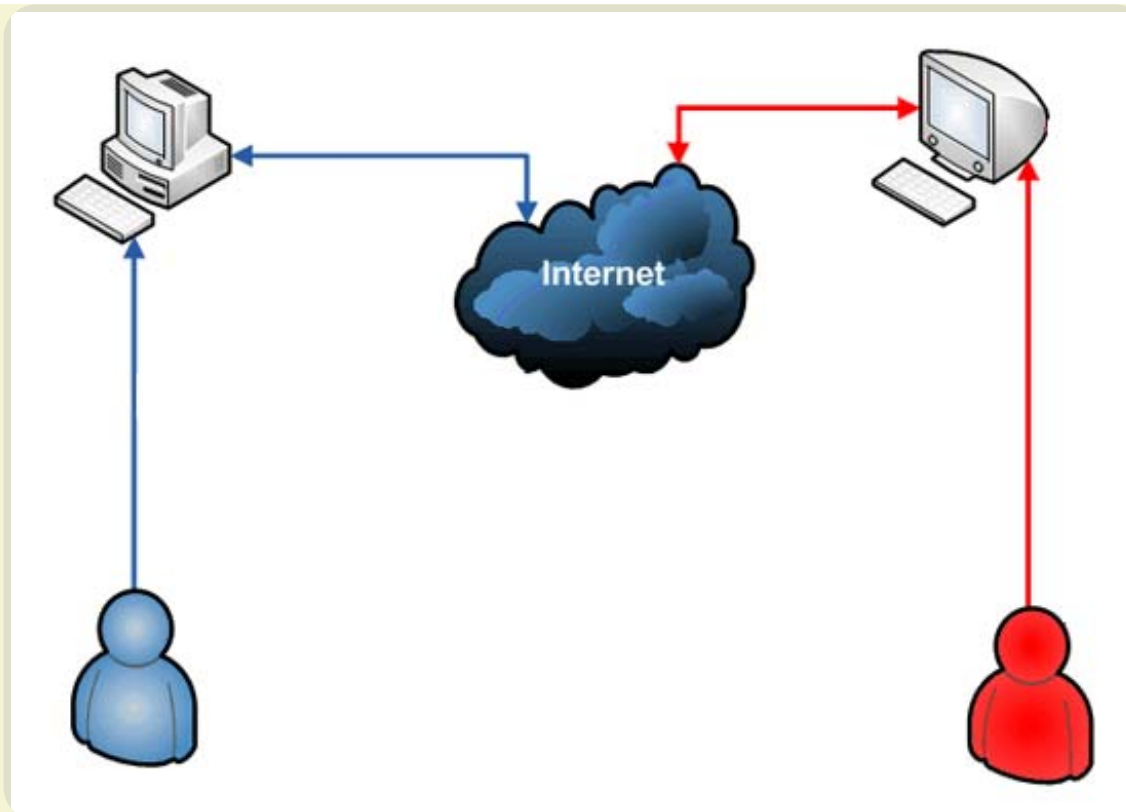


本站由 珠峰培训（专业前端培训）独家赞助

越来越多的人开始意识到，**网站即软件**，而且是一种新型的软件。

这种“互联网软件”采用客户端/服务器模式，建立在分布式体系上，通过互联网通信，具有高延时（high latency）、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；软件开发主要针对单机环境，网络则主要研究系统之间的通信。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、扩展方便，所以正得到越来越多网站的采用。

但是，到底什么是RESTful架构，并不是一个容易说清楚的问题。下面，我就谈谈我理解的RESTful架构。

一、起源

REST这个词，是[Roy Thomas Fielding](#)在他2000年的[博士论文](#)中提出的。



Fielding是一个非常重要的人，他是HTTP协议（1.0版和1.1版）的主要设计者、Apache服务器软件的作者之一、Apache基金会的第一任主席。所以，他的这篇论文一经发表，就引起了关注，并且立即对互联网开发产生了深远的影响。

他这样介绍论文的写作目的：

"本文研究计算机科学两大前沿----软件和网络----的交叉点。长期以来，软件研究主要关注软件设计的分类、设计方法的演化，很少客观地评估不同的设计选择对系统行为的影响。而相反地，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，常常忽视了一个事实，那就是改变应用程序的互动风格比改变互动协议，对整体表现有更大的影响。我这篇文章的写作目的，就是想在符合架构原理的前提下，理解和评估以网络为基础的应用软件的架构设计，得到一个功能强、性能好、适宜通信的架构。"

(This dissertation explores a junction on the frontiers of two research disciplines in computer science: software and networking. Software research has long been concerned with the categorization of software designs and the development of design methodologies, but has rarely been able to objectively evaluate the impact of various design choices on system behavior. Networking research, in contrast, is focused on the details of generic communication behavior between systems and improving the performance of particular communication techniques, often ignoring the fact that changing the interaction style of an application can have more impact on performance than the communication protocols used for that interaction. My work is motivated by the desire to understand and evaluate the architectural design of network-based application software through

principled use of architectural constraints, thereby obtaining the functional, performance, and social properties desired of an architecture.)

二、名称

Fielding将对互联网软件的架构原则，定名为REST，即Representational State Transfer的缩写。我对这个词组的翻译是"表现层状态转化"。

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，它的每一个词代表了什么涵义。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"表现层状态转化"中，省略了主语。"表现层"其实指的是"资源" (Resources)的"表现层"。

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（统一资源定位符）指向它，每种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独一无二的识别符。

所谓"上网"，就是与互联网上一系列的"资源"互动，调用它的URI。

四、表现层 (Representation)

"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" (**Representation**) 。

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，也可以用PNG格式表现。

URI只代表资源的实体，不代表它的形式。严格地说，有些网址最后的".html"后缀名是不必要的，因为这个后缀名表示格式，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。

五、状态转化 (**State Transfer**)

访问一个网站，就代表了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。

互联网通信协议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，让服务器端发生"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"表现层状态转化"。

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：GET、POST、PUT、DELETE。它们分别对应四种基本操作：**GET**用来获取资源，**POST**用来新建资源（也可以用于更新资源），**PUT**用来更新资源，**DELETE**用来删除资源。

六、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代表一种资源；

(2) 客户端和服务端之间，传递这种资源的某种表现层；

(3) 客户端通过四个HTTP动词，对服务器端资源进行操作，实现"表现层状态转化"。

七、误区

RESTful架构有一些典型的设计误区。

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协议中。

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方法表示show。

如果某些动作是HTTP动词表示不了的，你就应该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
```

另一个设计误区，就是在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，可以理解成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请求头信息的Accept字段中进行区分（参见[Versioning REST Services](#)）：



```
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
```

(完)

文档信息

- 版权声明：自由转载-非商用-非衍生-保持署名（创意共享3.0许可证）
- 发表日期：2011年9月12日
- 更多内容：档案 » 开发者手册
- 文集：《前方的路》，《未来世界的幸存者》
- 社交媒体： twitter,  weibo



育精英前端 冲月薪3万



QCon [北京站·2018]
全球软件开发大会

跻身顶级程序员所需要的全局视野

100+海内外实践先驱

20+领域的关键落地经验

QCon

北京·国际会议中心
会议：04月20-22日 / 培训：04月23-24日

立即了解

相关文章

■ **2018.03.30:** [Systemd 定时器教程](#)

Systemd 作为 Linux 的系统启动器，功能强大。

■ **2018.03.05:** [HTTP/2 服务器推送 \(Server Push\) 教程](#)

HTTP/2 协议的主要目的是提高网页性能。

■ **2018.02.27:** [Nginx 容器教程](#)

春节前，我看到 Nginx 加入了 HTTP/2 的 server push 功能，就很想试一下。

■ **2018.02.13:** [Docker 微服务教程](#)

Docker 是一个容器工具，提供虚拟环境。很多人认为，它改变了我们对软件的认识。

广告（购买广告位）



CODING
CLOUD DEVELOPMENT

体验极速
代码托管服务

—
3月31日前，点击注册并激活
立赠30天付费会员



优达学城
UDACITY

× 微信 腾讯微信官方合作课程

微信小程序开发

4 周入门实战

免费试听 >

留言（141条）

徐杰 说：

关注博主的博客有一段时间了。我想问下博主这个博文网站用什么技术实现的。最近在学网站开发语言，博主能否说的详细点，感激不尽！

2011年9月12日 16:49 | <#> | [引用](#)

darasion 说：

设计误区里边列出的，很多都这么设计啊。有什么不好吗？

2011年9月12日 17:20 | <#> | [引用](#)

lanisle 说：

简洁有力，表述连贯且清晰。谢谢阮兄。
祝中秋快乐！

2011年9月12日 17:36 | <#> | [引用](#)

崔大鹏 说：

网站即软件，我也早意识到这个问题了。阮兄中秋快乐！

2011年9月12日 21:19 | <#> | [引用](#)

ssddi456 说：

按我的理解：

这种交互模式相当于设计一个函数，函数内部（服务器）过程对客户端不可见，客户端只传递参数接受结果。

URI的设计就好象设计参数形式，函数的功能已经包含在网站本身不言自明，不需要再列出来。

2011年9月12日 22:53 | <#> | [引用](#)

风海迷沙 说：

asp.net mvc程序员路过，area/controller/view/pram 谁用谁知道。

2011年9月12日 23:16 | <#> | [引用](#)

张翼 说：

一直都有听说rest这个东西，但是没有认真研究过。这篇文章给了一个不错的入门介绍。谢谢！

能够再列出一些你找到的关于rest的网上的资源？我想再详细了解一下。

中秋快乐！

2011年9月13日 04:46 | <#> | [引用](#)

liws 说：

引用风海迷沙的发言：

asp.net mvc程序员路过，area/controller/view/pram 谁用谁知道。

这种URL是最不友好的，全是程序员按照自己的编程思路定制的URL,很多都不符合产品要求

2011年9月13日 09:29 | <#> | [引用](#)

乱发吹风 说：

用".html", ".json", ".xml"来表示表述类型，以及在URI上加入版本号，在实际使用中会更简单直观一些。

因为网站管理员检查HTTP日志时，大多数(一般为GET)情况下直接看URI就可以了解问题所在。

而这对于RESTful接口的应用开发者，也会更方便一些

你的博客内容感觉比较偏科普一些，其实可以写得再深入一些，或者可以把你的资料来源也列出来，方便

有兴趣进一步学习的读者去查阅.

2011年9月13日 09:53 | # | 引用

ls zhao 说 :

网站, 基于网络的应用软件, 基于PC的本地应用软件...

那整个互联网可以看成是一个软件, 分为不同的部分

2011年9月13日 12:39 | # | 引用

limu 说 :

感谢分享. 有两个问题,

1. post和put的区别, 何时用put? 我看文章里说更新用哪个都可以, 新增用post. 为什么?

2. 关于最后版本处理那块, JavaScript操作http的accept值还是有些困难, 有没有替代方案, 放在URL参数中么? 另外httpserver支持通过accept版本决定投递什么资源么? 浏览器支持按不同的accept分别缓存资源么?

2011年9月13日 13:27 | # | 引用

linyang 说 :

楼主的理解非常的好, 有一些我想补充, 顺别回答其它一些朋友的问题.

根据理查德森模型 (<http://martinfowler.com/articles/richardsonMaturityModel.html>), REST架构的成熟度有3个等级:

Level 0 POX (这个就不算REST了)

Level 1 Resources

Level 2 Http verbs

Level 3 Hypermedia Controls

楼主所表述的模型在level 2上, 这也是目前大多数RESTful的应用所在的成熟度.

Level 0 POX

这类应用只有一个URI上的上帝接口, 根据交换的XML内容操作所有的资源. 往往导致上帝接口越来越复杂, 越来越难以维护.

Level 1 Resources

这一级别主要解决了上帝接口的问题, 使得各种资源有了自己相应的URI, 虽然仍然是POX的交互方式, 但是每一个接口都更加紧凑和内聚, 相应的容易维护起来.

这里的主要问题是URI templating和URI tunneling.

URI templating带来的结果是服务器端和客户端的紧耦合, 任何时候服务器段想改变自身的URL schema的时候, 都要break已经存在的客户端应用.

URI tunneling带来的问题包含URI templating, 而且放弃了使用http协议标准带来的任何好处, level 2中详述.

早期的rails routes就是url templating/tunneling. Rails3中已经更加靠近level 2了.

2011年9月13日 14:07 | <#> | [引用](#)

linyang 说 :

Level 2 Http verbs

这一级别使用http verbs来对各种资源进行crud操作, 使得应用程序的接口更加的统一, 语义更加明确. 同时, 因为遵照http的标准进行交互, 很多http提供的好处几乎可以免费的得到.

1. Cache

按照HTTP协议, GET操作是安全的, 幂等(Idempotent)的. 任意多次对同一资源的GET操作, 都不会导致资源的状态变化. 所以GET的结果是可以安全的cache. 所有http提供的cache facilities 都可以被利用起来, 大幅度提高应用程序的性能. 甚至你仅仅只在response里加上cache directives就可以免费获得网络上各级的缓存服务器, 代理服务器, 以及用户客户端的缓存支持. 互联网上几乎所有的应用你都可以粗略统计得到Get VS Non-Get的请求比例约为 4:1. 如果你能为GET操作加上缓存, 那将极大提供你的程序的性能.

2. Robust

在HTTP常用的几个动词里, HEAD, GET, PUT, DELETE 是安全的,幂等的. 因为对同一资源的任意多次请求, 永远代表同一个语义. 所以任何时候客户端发出去这些动词的时候, 如果服务器没有响应, 或者

返回错误代码, 客户端都可以非常安全的再次执行同一操作而不用担心重复操作带来不同的语义及最终结果. POST, PATCH操作就不是安全的, 因为当客户端向服务器端发出请求后, 服务器没有响应或者返回错误代码, 客户端是不能安全的重复操作的. 一定只能重新与服务器确认现在的资源状态才能决定下一步的操作.

绝大部分的RESTful应用就停在这里了, 当然也满足绝大多数的需求.

2011年9月13日 14:07 | <#> | [引用](#)

linyang 说 :

Level 3

RESTful的架构本意是"在符合架构原理的前提下, 理解和评估以网络为基础的应用软件的架构设计, 得到一个功能强、性能好、适宜通信的架构。"

这个世界上规模最大的, 耦合度最低, 最稳定的, 性能最好的分布式网络应用是什么? 就是WEB本身. 规模,稳定,性能都不用说了. 为什么说耦合度低呢? 想一想每个人上网的经历, 你几乎不需要任何培训就可以上一个新的网络购物平台挑选商品,用信用卡付款,邮寄到自己家里.

把网站的程序想像成一个状态机, 用户在一系列状态转换中完成自己的目标. 这中间的每一步, 应用程序都告诉你当前的状态和可能的下一步操作, 最终引导用户从挑选商品,挑选更多商品,到支付页面,到输入信用卡信息,最终完成付费,到达状态机的终点.

这种service discoverability和self-documenting就是level 3想解决的问题

在这里面, 告诉用户当前状态以及各种下一步操作的东西, 比如链接, 按钮等等, 就是Hypermedia Controls. Hypermedia Controls 就是这个状态机的引擎.

Level 3的REST架构就是希望能够统一这一类的Hypermedia Controls, 赋予他们标准的, 高度可扩展的标准语义及表现形式, 使得甚至无人工干预的机器与机器间的通用交互协议边的可能. 比如你可以告诉一个通用的购物客户端, "给我买个最便宜的xbox", 客户端自动连上google进行搜索, 自动在前10个购物网站进行搜索, 进行价格排序, 然后自动挑选最便宜的网站, 进行一系列操作最终完成用信用卡付费, 填写个人收件地址然后邮寄.

这些都依赖于Hypermedia Controls带来的这种service discoverability和self-documenting

更多的关于REST的细节及其应用和实现, 请参考Rest in Practice. 非常非常棒的一本书, 把REST讲的非常透彻.

个人的理解难免有所偏颇, 还请大家批评指正

2011年9月13日 14:08 | <#> | [引用](#)

XXXX 说 :

还不知道Rails的你就OUT 了 , 那真是谁用谁知道!

2011年9月13日 18:49 | <#> | [引用](#)

阮一峰 说 :

引用limu的发言 :

1.post和put的区别,何时用put?我看文章里说更新用哪个都可以,新增用post.为什么?

PUT是幂等方法, POST不是。所以PUT用于更新、POST用于新增比较合适。

引用limu的发言 :

2.关于最后版本处理那块,JavaScript操作http的accept值还是有些困难,有没有替代方案,放在URL参数中么?

XHR对象的setRequestHeader方法可以指定头信息啊, 指定accept好像没有什么困难吧。

2011年9月13日 20:22 | <#> | [引用](#)

limu 说 :

引用阮一峰的发言 :

嗯 没错 谢谢阮老师.

不过一说到版本号就联想到静态资源以及他们的缓存策略,比如js,css,图片啥的.呵呵.除了XHR以外的浏览器发起Http无法指定request header.

而通常XHR获取的是动态数据吧,一般要实时最新的,REST应用中有什么具体场景会应用到带版本号的XHR数据么?

2011年9月14日 09:29 | # | 引用

CFC4N 说：

博主写的非常好，但如果是生硬的理解，可能很难理解透彻，或者说很容易忘记。当经过一次设计之后，之前对REST理解不透彻的同学，可能会回头来再次阅读。作为读者，感谢博主了。

PS：“有些网址最后的”.html”后缀名是不必要的”对于浏览器(解析执行渲染)来说，确实不必要，但对于服务器来说，需要完整的文件名用来定位请求的资源所在的真实路径。我觉得这条不适合作为论证“表现层”的一个理由。

引用linyang的发言：

楼主的理解非常的好，有一些我想补充，顺便回答其它一些朋友的问题

这位仁兄补充的也非常好。

2011年9月14日 14:08 | # | 引用

kedron 说：

为什么阮这么一个非专业的人员比很多所谓的程序员理解的还要深入些？只是因为他英语好么？期待阮翻译更多的好书。

netwix 说：

引用kedron的发言：

为什么阮这么一个非专业的人员比很多所谓的程序员理解的还要深入些？只是因为他英语好么？期待阮翻译更多的好书。

他根本就不是非专业人员 或者根本不是你想象的仅仅是有翻译书的经历

2011年9月17日 22:26 | # | 引用

Cat 说：

这样的话 Google APIs 和 Twitter APIs 在 URL 中加入了版本，是不正确的 Rest API

2011年9月18日 00:35 | # | 引用

Ethan 说：

@linyang：

博主的文章结合你的评论，真是受益匪浅。准备拜读你推荐的Rest in Practice，谢谢。

2011年9月18日 22:45 | # | 引用

noodles 说：

这篇文章所涉及到的内容banq在jdon论坛里讨论过，具体的链接：

<http://www.jdon.com/jivejdon/thread/41716>

2011年9月19日 01:09 | # | 引用

28ding 说：

考虑到uri的使用者很大一部分都是浏览器，而在浏览器向服务器提交请求的时候是不能任意增加http头的（除非把浏览器的浏览功能全部用javascript实现，XHR可以指定，但这不太实现），许多信息都得放在uri地址中。

所以uri中出现动作、版本等内容都是正常的

2011年9月20日 09:25 | # | 引用

jacklondon 说：

如果是B/S系统的话，只能是用 URL 来体现了：

<http://mysystem/system/user?action=add>

<http://mysystem/system/user?action=delete>

<http://mysystem/system/user?action=update>

<http://mysystem/system/user?action=query>

我自认为也符合 RESTful .

难道你能在页面中放入一个链接，对应 HTTP DELETE?

同样的道理，在网址中放入版本号，我也认为没有问题。

"REST是设计风格而不是标准"，在 B/S 架构中，考虑到页面链接(也就是 HTML 标签)的实际情况，用网址体现操作，可以理解。

---欢迎大家试用我们的折桂单点登录系统 <http://zhegisoft.com>

2011年9月23日 11:02 | # | 引用

夜猫子 说：

引用阮一峰的发言：

PUT是幂等方法，POST不是。所以PUT用于更新、POST用于新增比较合适。

具体的程序逻辑里面，完全可以把POST动作实现为幂等操作，比如POST /user 创建新用户，POST /user/1 修改id为1的用户信息，无论POST /user/1多少次，都是对id为1的用户进行修改（幂等）。

真正的理由是，REST仅仅是一种架构风格的指导规范，并非具体技术实现。因此用POST新建，PUT修改仅仅是具体实施过程中的约定。就像代码需要合理的缩进一样，虽然不缩进，程序也能执行（python除外），但安装这个约定去做，事情可以做得更漂亮点。

我也看到过其它的REST文章建议说POST修改，PUT新建，我更倾向于POST新建，PUT修改。

2011年9月27日 14:03 | # | 引用

Linyang 说：

引用夜猫子的发言：

具体的程序逻辑里面，完全可以把POST动作实现为幂等操作，比如POST /user 创建新用户，POST /user/1 修改id为1的用户信息，无论POST /user/1多少次，都是对id为1的用户进行修改（幂等）。

真正的理由是，REST仅仅是一种架构风格的指导规范，并非具体技术实现。因此用POST新建，PUT修改仅仅是具体实施过程中的约定。就像代码需要合理的缩进一样，虽然不缩进，程序也能执行（python除外），但安装这个约定去做，事情可以做得更漂亮点。

我也看到过其它的REST文章建议说POST修改，PUT新建，我更倾向于POST新建，PUT修改。

没错，你是可以把post /users/1 实现为幂等的。但是通用客户端，比如浏览器，没有这样的预先假设。当post 操作异常时，比如超时，服务器503等等，浏览器是不能自动重试post操作的。这样你的幂等的 post /users/1 并没有给你带来太多的好处，除非你的客户端都是定制客户端

REST确实不是技术标准，但是HTTP是标准。

2011年9月27日 23:16 | # | 引用

夜猫子 说：

引用Linyang的发言：

没错，你是可以把post /users/1 实现为幂等的。但是通用客户端，比如浏览器，没有这样的预先假设。当post 操作异常时，比如超时，服务器503等等，浏览器是不能自动重试post操作的。这样

你的幂等的 `post /users/1` 并没有给你带来太多的好处，除非你的客户端都是定制客户端

REST确实不是技术标准，但是HTTP是标准。

HTTP标准没有真正得到落实

现在其实处于一个“尴尬”的状况，一方面REST已经从理论上提供了解决方向，另外一方面是实际环境其实还没有提供足够多的REST支持。

比如通过ajax，大多数(?)浏览器可以发出原生的PUT/DELETE方法，不使用ajax，就只能GET/POST。

实际开发中，没有谁敢保证浏览器可以发出真的PUT，也没有谁敢保证web服务器可以理解真正的PUT，即使web服务器可以理解，也不能保证服务器端开发语言能够理解（比如php有 `$_GET`，`$_POST`，但是没有`$_PUT`）

所以PUT/DELETE方法好些时候其实是通过POST模拟的，常见的方法有两种

```
<form method="post">
<input type="hidden" name="_method" value="put"/>
</form>
```

或者request message包括自定义的http header，一般是X-HTTP-METHOD-OVERRIDE: PUT

POST/PUT是否幂等，只能由你的程序来保证，并没有实际的支撑，具体选择POST还是PUT，其实就是一种约定或者规范。

PS：写到后面我有点觉得自己在说废话.....

2011年9月28日 15:30 | 丑 | 引用

linyang 说：

引用夜猫子的发言：

HTTP标准没有真正得到落实 现在其实处于一个“尴尬”的状况，一方面REST已经从理论上提供了解决方向，另外一方面是实际环境其实还没有提供足够多的REST支持。

确实，很遗憾，设计良好的HTTP标准没有得到大范围的遵守。

HTTP协议被设计为通用的应用程序协议，语义明确，高度可扩展。可惜大部分的应用仅仅把它当作传输协议。

特别是基于HTML的应用，HTML 4.0 的表单只支持GET和POST。所以像Rails这样的框架只能是使用其它方式模拟PUT和POST，在Server端Dispatch时还原。至少Server端还是可以把这些HTML 表单的局限隔离开的。（很遗憾，HTML5貌似也把PUT和DELETE的支持取消了）

除去HTML应用，广大的基于HTTP的Web Services还是可以，也应该充分利用HTTP协议实现RESTful Services的。

2011年9月29日 06:02 | # | 引用

abloz.com 说：

我前些天想到在网页上实现表现和数据的分离。同样的表现层只下载一次，而数据可以替换。是不是很符合rest的思想呢？

2011年10月25日 14:53 | # | 引用

cailei 说：

我觉得这个理论概念太重

- 1.对开发人员来说，uri这一层理解上存在的问题，需要改进
- 2.对使用者来说，URI除了知道你首页后，其他全是用户引导，用户是不会去看更不会去记忆你的uri

2011年11月 2日 12:01 | # | 引用

Dozer 说：

引用liws的发言：

这种URL是最不友好的，全是程序员按照自己的编程思路定制的URL,很多都不符合产品要求

这只是 MVC 里的几个概念，最终的 URL 经过 Routing 后并不是这个格式。

2012年3月28日 09:46 | # | 引用

小池 说：

真实醍醐灌顶。评论部分也有牛人。

2012年6月19日 14:14 | # | 引用

逆流的鱼 说：

一石激起千层浪，呵呵，受教了，继续逆流而上

2012年6月26日 01:43 | # | 引用

插件吧 说：

谢谢分享，受益颇多~~~

2012年7月15日 00:53 | # | 引用

bangyue 说：

很受用，也觉得restful具有一定的应用前景，全新网站软件的设计可以采用这种设计架构。我觉得目前web服务器端各种架构设计皆有自己的优点

2012年8月 8日 14:37 | # | 引用

圣僧 说：

又一年中秋，看了评论才知道你是2011中秋写的，2012的中秋节也祝你中秋快乐，很凑巧今天是中秋正好看到你的文章。你的文章很有水准，以后会来常看看的。

2012年9月30日 01:30 | <#> | [引用](#)

zhan0903 说：

收益颇多，谢谢

2012年10月18日 09:34 | <#> | [引用](#)

十二能 说：

最近使用spring3.0+ 使用RESTful,看了文章受教了

2012年11月 3日 13:30 | <#> | [引用](#)

haitao 说：

一直反感rest把动作放在method，觉得作为第一个参数更好：act=insert&arg1=a&arg2=b&...
考虑到缓存，放在method的确方便

2012年11月13日 10:59 | <#> | [引用](#)

DolphinBoy 说：

感觉没怎么懂，RESTful到底是什么，只是一种规范吗，还是一种技术？只是POST和PUT的区别吗，貌似Nodejs里面POST,PUT,GET,UPDATE有区分

2013年1月10日 09:42 | <#> | [引用](#)

程序员 说：

真正使用中很少绝对RESTful的，大都有文中所述的错误做法，但是照样很有效。

2013年6月25日 10:09 | <#> | [引用](#)

菜鸟 说：

有没有一些具体的实例啊！！！！

2013年8月12日 09:04 | <#> | [引用](#)

丕子 说：

又是一年中秋节，看了原文和评论才收获全面了

2013年9月24日 20:19 | <#> | [引用](#)

Rex wong 说：

关于版本号的问题，版本加到url里面，假设我v1.1部署一套环境，v1.2部署一套环境，客户端就可以通过版本的不同选择调用不同版本的接口。但是写在header里面，如果在不更改后台代码的情况下，实现客户端不同版本的请求转发呢？谢谢

2013年9月30日 18:24 | <#> | [引用](#)

田原 说：

很简洁明了的介绍。

2013年10月28日 16:09 | <#> | [引用](#)

demo 说：

在Yahoo developer network中的我看到一些web service的restful api就是直接在URI里面包含版本号。

2013年11月 8日 13:13 | <#> | [引用](#)

Richard 说：

引用风海迷沙的发言：

asp.net mvc程序员路过，area/controller/view/pram 谁用谁知道。

你确定是area/controller/view/pram，而不是area/controller/action/pram？

另外，asp.net webapi 有研究过吗？

2014年1月 5日 22:11 | # | 引用

handerliu 说：

好文

2014年1月24日 09:42 | # | 引用

凌云 说：

请问这个地方

POST /transaction HTTP/1.1

Host: 127.0.0.1

from=1&to=2&amount=500.00

这三个参数也是资源，不是应该使用../1/2/500这样的url吗，个人愚见，望不吝赐教

2014年2月10日 17:19 | # | 引用

eric 说：

引用Cat的发言：

这样的话 Google APIs 和 Twitter APIs 在 URL 中加入了版本，是不正确的 Rest API

在URL中带有版本信息，并不是使用错误。这是一种正确的合法的，符合REST各种约束的使用方法（一些英文书籍中已经介绍过这种version方法）。REST进行版本管理方法不是只有一种，你说的

accept header只是其中一种而已

2014年3月10日 22:47 | # | 引用

chloe 说：

阮叔叔你好，我是一名学生，我想请教一个问题：CAD中为什么要提出“半宽”的定义，有什么意义吗？什么困惑，百度也没出来什么结果，这才来请教您。（因为我认为你很厉害）谢谢你~~o(^▽^)o

2014年3月10日 23:07 | # | 引用

morgan 说：

说实话看完博客对REST没有特别立体的认识，只是知道博主说的一些观念已经应用在一些领域，平时没有细想为什么这样，这方面受教不少。但是，看了博客和所有的评论之后，我感觉自己越来越糊涂了。为啥到后来都扯到HTTP里的方法里去了？

2014年5月12日 17:03 | # | 引用

twlkyao 说：

有长见识了。

2014年5月19日 18:54 | # | 引用

ajian 说：

引用凌云的发言：

请问这个地方

POST /transaction HTTP/1.1

Host: 127.0.0.1

from=1&to=2&amount=500.00

这三个参数也是资源，不是应该使用../1/2/500这样的url吗，个人愚见，望不吝赐教

同问

2014年6月25日 17:35 | <#> | [引用](#)

131rong 说：

写的很好 赞一个

2014年7月 2日 23:38 | <#> | [引用](#)

Anrin 说：

对比一下网上其他文章，博主的文章简洁且透彻。体现了博主的水平。

2014年7月11日 13:52 | <#> | [引用](#)

Ching 说：

学习Rails接触到REST,感谢阮老师分享，赞赞!

2014年7月29日 17:08 | <#> | [引用](#)

leeir 说：

博主您好 可否有什么resetful方面的好书推荐

2014年8月15日 14:45 | <#> | [引用](#)

java源代码 说：

目前感觉weibo的api设计的url还算符合restful的原则。

2014年8月20日 14:10 | <#> | [引用](#)

fighive 说：

看得不是很懂, 感觉貌似就是一种设计原则?

但是不理解为什么要遵循这种原则, 有什么好处

2014年9月12日 17:57 | <#> | [引用](#)

ox1D 说 :

请问老师你的流程图片是怎么做的？

2014年11月 3日 22:48 | <#> | [引用](#)

Leo、 说 :

我想说完全没看懂

2014年11月22日 16:38 | <#> | [引用](#)

一个在IT领域刚刚起步的年轻人 说 :

感觉博主这里是不是弄错了。

URI：统一资源标识符；

URL：统一资源定位符，相当于URI+Location（标识+定位）；

2014年11月29日 21:08 | <#> | [引用](#)

harrison 说 :

REST四个基本原则：

- 1.使用HTTP动词：GET POST PUT DELETE；
- 2.无状态连接，服务器端不应保存过多上下文状态，即每个请求都是独立的；
- 3.为每个资源设置URI；
- 4.通过XML JSON进行数据传递；

实现上述原则的架构即可称为RESTFul架构。

- 1.互联网环境下，任何应用的架构和API可以被快速理解；
- 2.分布式环境下，任何请求都可以被发送到任意服务器；

3.异构环境下，任何资源的访问和使用方式都统一；

2014年12月12日 09:13 | # | 引用

大地 说：

博主之后的《RESTful API 设计指南》一文，已经纠正此博文中“另一个设计误区，就是在URI中加入版本号”的观点，还请及时修改，以免误导大家吧。

2014年12月30日 11:42 | # | 引用

王军 说：

restful是一种具体框架，还是一种设计思想？我一直稀里糊涂的。

2015年1月23日 15:22 | # | 引用

JetAn 说：

引用王军的发言：

restful是一种具体框架，还是一种设计思想？我一直稀里糊涂的。

当然是一种设计思想了。框架可以实现为RESTful的。

2015年2月14日 14:11 | # | 引用

斗转星移 说：

实用就好

2015年3月 1日 18:12 | # | 引用

MilkMan 说：

引用乱发吹风的发言：

用".html", ".json", ".xml"来表示表述类型, 以及在URI上加入版本号, 在实际使用中会更简单直观一些.

因为网站管理员检查HTTP日志时, 大多数(一般为GET)情况下直接看URI就可以了解问题所在. 而这对于RESTful接口的应用开发者, 也会更方便一些

你的博客内容感觉比较偏科普一些, 其实可以写得再深入一些, 或者可以把你的资料来源也列出来, 方便有兴趣进一步学习的读者去查阅.

REST是一个指导你设计现代web架构的原则和约束, N年来N多经验教训的批判继承的方法论。你不遵循又不犯法：)。

2015年3月 5日 01:02 | <#> | [引用](#)

nyer 说：

restful是http接口的设计规范，像浏览器用户可以访问的一些页面，加个后缀名没啥不好的

2015年3月 6日 15:57 | <#> | [引用](#)

zhaob 说：

引用王军的发言：

restful是一种具体框架，还是一种设计思想？我一直稀里糊涂的。

不是框架，不是技术，不是设计思想，是一种混合式“架构风格”

2015年4月25日 22:55 | <#> | [引用](#)

wnow20 说：

通俗易懂~~~大赞

pjqqq 说：

RESTful根本没有必要,完全是给自己找罪受,不能说大牛说的每个概念都当圣经吧

2015年6月 7日 16:33 | # | 引用

shann 说：

在了解Angularjs和Backbonejs这些前端框架的时候，发现这些先进的前端框架都是支持REST风格的服务端API。目前本人只是对REST有个大概的了解，希望能够把新的比较先进的架构引入到手头上的项目，但是有一个疑问：REST风格能不能应对数据结构还不成熟，开发期需要频繁增加数据库字段的情况。

2015年6月 8日 18:04 | # | 引用

Gevin 说：

URI不是一个资源独一无二的标识符，一个资源可以对应多个URI，只不过很少这样做而已

2015年6月 8日 20:51 | # | 引用

业界大牛 说：

为什么总感觉看起来怪怪的

POST /accounts/1/transfer/500/to/2

POST /transaction HTTP/1.1

from=1&to=2&amount=500.00

说实在的,在我看来,这就是抽象的过程,你写着写着发现,这整个动作就是参数的改变,那么把参数抽出来就好了.

为什么给你这篇文章写起来就有点奇淫技巧的感觉

2015年6月22日 09:56 | # | 引用

npc 说：

引用ajian的发言：

同问

用过.net web api 说下看法

../1/2/500 这种方式不能一样看出每个参数的意思. 而且参数的位置必须固定.

from=1&to=2&amount=500.00 这种直观, 参数位置可以随意, 就比如

from=1&amount=500.00&to=2一样

2015年7月27日 16:55 | # | 引用

Frend 说：

还是阮老师写的博文有水平+易懂，终于算是领会restful api是个什么意思了

2015年7月29日 16:52 | # | 引用

郝伟 说：

引用shann的发言：

在了解Angularjs和Backbonejs这些前端框架的时候，发现这些先进的前端框架都是支持REST风格的服务端API。目前本人只是对REST有个大概的了解，希望能够把新的比较先进的架构引入到手头上的项目，但是有一个疑问：REST风格能不能应对数据结构还不成熟，开发期需要频繁增加数据库字段的情况。

你这个示例我觉得通过在url中增加版本号，应该是稍微好点的解决方案。 我比较认同在url中增加版本号，这样切换或者回退应该都比较方便，而且不管是客户端还是服务端，不同版本的代码隔离应该也比较好处理。

2015年7月30日 11:01 | # | 引用

vhuabigger 说：

引用郝伟的发言：

你这个示例我觉得通过在url中增加版本号，应该是稍微好点的解决方案。

我比较认同在url中增加版本号，这样切换或者回退应该都比较方便，而且不管是客户端还是服务端，不同版本的代码隔离应该也比较好处理。

个人觉得在业务本身并不完善的时候，不建议使用Rest风格的服务框架，虽然可以使用版本号来区分，但是业务本身的变化可能带来的是版本号的混乱，在我所接触过的将现有架构转向Rest的项目中也都属于业务流程已经基本完善，处于一个需要将具体的业务流程解耦、接口细化的过程，当然如果项目本身已经具有一个设计良好、粒度合适的业务架构，那么引入Rest也是可以的，个人因为接触的涉及Rest的项目也不算太多，不知道目前使用Rest的项目是否都是业务重构之后才引入的，一点愚见

2015年8月24日 09:44 | <#> | [引用](#)

智愚 说：

请问 DELETE 方法能不在请求中携带参数，常见的 DELETE 方法在URL中指定被删除的对象的ID，但是这种方法中这个ID的确定还需要另一次的请求，对于用户而言只会提供一些被删除对象的属性，请问能不能在请求中携带这些属性交给Server端标志要删除的对象？

2015年9月30日 11:28 | <#> | [引用](#)

wkl17 说：

Cool..很浅显的文字..不会像有些文章看完了都不知道在说什么的感觉..

2015年10月10日 16:51 | <#> | [引用](#)

捷克 说：

想问问作者， Restful构架， 资源权限有什么好的方法控制吗。比如说， 我有一个人员管理的系统， 不同的权限的人可以查询不同的人。 经理可以看见所有的人， 小组长只能看见这个小组的人。

2015年10月27日 05:12 | <#> | [引用](#)

silkmilk 说：

那对于我在网站购买了东西，在付款界面点击提交，这时网络不好或者其他情况，显示了一个错误代码，这种情况的话，是属于PUT还是POST呢

2015年11月20日 10:41 | <#> | [引用](#)

yking 说：

這篇是我看過少數介紹的很好的restful.

不過我個人不建議使用restful. 相信有不少人在實案中嘗試使用後, 最後會不想使用它. 但也有朋友愛用的要死, 因為他的案子80%都長的差不多.

2015年12月18日 17:50 | <#> | [引用](#)

M 说：

这种"互联网软件"采用客户端/服务器模式，建立在分布式体系上，通过互联网通信，具有高延时（high latency）、高并发等特点。

“高延时”怎么理解

2016年1月 7日 17:03 | <#> | [引用](#)

Sophia 说：

该更新一下了呢

2016年1月18日 16:49 | <#> | [引用](#)

zsf 说：

为何网站上大多数的图片资源的URI是直接加图片后缀的，这显然不满足REST原则，例如百度首页的图片：

https://sso.bdstatic.com/5aV1bjqh_Q23odCf/static/superman/img/logo/bd_logo1_31bdc765.png

2016年1月24日 22:16 | # | 引用

蒙娜丽莎 说：

写的挺好

2016年2月19日 09:12 | # | 引用

liangjixun 说：

你好，你的这篇文章说把版本号放入url中是设计误区，然后你的另一篇文章

http://www.ruanyifeng.com/blog/2014/05/restful_api.html

又说应该将API的版本号放入URL，到底哪个是对的

2016年3月 7日 14:47 | # | 引用

lion 说：

URI是统一资源标识符吧，不是定位符

2016年3月16日 11:14 | # | 引用

小三斯基 说：

老师您好，文章写的非常有启发性，我是初学者。纠正一下文章里的小错误吧：“你可以用一个URI（统一资源定位符）指向它”这一句有些不妥，应为“统一资源标识符”，定位符的话是指URL。根据文章的语义，老师您应当指的是更抽象的URI

2016年4月 5日 10:31 | # | 引用

linwaner118 说：

如果客户端想要操作服务器，必须通过某种手段，让服务器端发生"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"表现层状态转化"。-- 这里的"这种转化是建立在表现层之上的"，这句话不是很理解，上面说，表现不是指的是资源的呈现形式吗？那状态的转换 建立在表现层之上的，代表什么意思？还望博主解答。

2016年4月 6日 14:24 | <#> | [引用](#)

小小 说：

关注很久了，感谢博主的默默奉献

2016年4月16日 13:32 | <#> | [引用](#)

杨洋 说：

uri是统一资源标识符，url才是统一资源定位符

2016年4月18日 13:31 | <#> | [引用](#)

hui 说：

state 是状况，声明的意思；

status 是状态的意思；

you r welcome.

2016年5月20日 02:05 | <#> | [引用](#)

AudienL 说：

引用darasion的发言：

设计误区里边列出的，很多都这么设计啊。有什么不好吗？

最后那个版本号，感觉没什么不妥

2016年5月23日 14:42 | # | 引用

tonghu 说：

url如果不能带动词的话

如果是用户操作，登录，注册。应该怎么设计url呢

/v1/user/后面带什么呢

能否告知下思路。。

2016年6月14日 13:35 | # | 引用

bensonlin 说：

引用tonghu的发言：

url如果不能带动词的话

如果是用户操作，登录，注册。应该怎么设计url呢

/v1/user/后面带什么呢

能否告知下思路。。

根据文章的描述，应当将具体的操作放到参数中进行指明

2016年6月14日 22:19 | # | 引用

我是猪我怕谁 说：

盲目的把某一个人定出来的东西的适用场景扩大化到适用任何的场景，本来就是一个浮云。

把一种风格当成一种标准，从而脱离实际的项目也数不胜数，恐怕RESTful就是其中一个。

理想是美好的，现实是残酷的。

这篇文章也就第一段话是有点用的。

但是即便在看到这篇文章之前，我自己能随时敲得出这样的观点。

正因为如此，所以后边的文章和起源，让我看了就是觉得没有必要太过执着哪一种一定是对的。

如果第一段是为了后边的洗脑和盲目崇拜而设立的，就应该把这段话去掉。

再看看后边的内容，谁还会觉得这篇文章有水准？不信就试试看，把第一段话去掉，重新读读这篇文章看看。

本来设计风格，就可以是一种约定嘛。对吧，看得懂就行了。

2016年6月17日 13:16 | # | 引用

我是猪我怕谁 说：

补充两点：

像增删查改，状态机，表现层这种听起来高大上的名词。稍为做过几个项目的人都知道是什么了。给小白洗洗脑是可以的。

文中也大量用到“错了”“正确”这种绝对化的术语。

就这样的不严谨的用词水平，也能称作是博士写出来的东西？

不知道有一种说法叫做“不科学”与“科学”吗？

2016年6月17日 13:19 | # | 引用

varprob 说：

(1) 如果Fielding参与过几个大型的跨系统、跨平台的应用，我想他现在一定会为他的“REST”感到不满意。

(2) 如果阮一峰不把Fielding当做神来看待，那么也一定不会说出“RESTFul的HTTP协议中不能有动词”这样超乎寻常的论点来。

(3) 在小项目中，REST与否，不重要，在大项目中，REST并不重要。

(4) HTTP协议只是工具，不是目的，REST同样只是一个工具，绝非目的，我们决不能本末倒置。

(5) 使用工具的目的是为了解决问题，那么问题是什么？答：。。。。。

2016年7月 5日 10:22 | # | 引用

xxbaby 说：

自相矛盾啊，在你的帖子http://www.ruanyifeng.com/blog/2014/05/restful_api.html中，说需要将version放入到url中，本篇帖子又说这是错误的，到底应该信哪篇？

2016年7月 8日 08:29 | # | 引用

阿修 说：

@tonghu：

RESTful 推荐的是无状态的，也就是说，按照RESTful 风格，本不应该存在登录一说。因为涉及session就是有状态了，

http 协议中 请求报头 中应该添加Authorization 字段，用于证明客户端有权查看某个资源。当浏览器访问一个页面时，如果收到服务器的响应代码为401（未授权），可以发送一个包含Authorization请求报头域的请求，要求服务器对其进行验证。这里就应该携带帐号密码（采用ssh）去获取资源。

2016年7月18日 00:08 | # | 引用

刘杰 说：

有趣，请允许我搜藏到我的博客慢慢看。感觉发现个大神，收藏。

2016年8月26日 00:05 | # | 引用

业余草 说：

写的不错，一看就懂，大概意思明确！

2016年9月 4日 17:17 | # | 引用

提拉米苏 说：

作者写的总是很通俗易懂，不过URL是统一资源定位符，URI是统一资源标识符吧？

孙张飞 说：

好吧，我一个前端做安卓的看这个真是有点累，完全搞不懂在说些什么。。。但是最近要用rxjava+dagger2+retrofit真心看的头痛，求大神指导啊

2016年9月12日 11:05 | # | 引用

tsrot 说：

还是没看的太懂，比较模糊

2016年9月29日 09:56 | # | 引用

六且 说：

在学习ThinkPHP5.0看到Restful,然后百度过来的，这样对技术的介绍，浅显易懂。我这种菜鸟也看懂了个大概

2016年10月11日 11:05 | # | 引用

王龙 说：

厉害了我的哥！关注博客很久了，竟然没发现这篇文章，还是通过Github过来的！以前一直不理解Restful API，今天看了Rest才知道！解了心头疑惑，甚是舒畅，谢谢了！

2016年10月12日 09:49 | # | 引用

月下临风 说：

写的真好，一看就懂

2016年10月19日 11:14 | # | 引用

zuwen 说：

那么用来区分当前项目是否是restful架构搭建的key难道只有url的命名规范？

像下面两种url风格是否都可以用restful架构实现？

1 : http:localhost:8080/parking/list/1001

2: <http://localhost:8080/parking/list?id=1001>

实现restful架构的核心到底在前端还是在后端？

2016年11月 1日 17:46 | # | [引用](#)

leslie 说：

天了撸，我才看到中这篇文章！

2016年11月15日 21:39 | # | [引用](#)

besthyhy 说：

非常不错！但我不得不指出文中的错误。URI是资源标识符，URL才是统一资源定位符。二者区别很大。

2016年12月 9日 10:27 | # | [引用](#)

杨彬 说：

好看。理解深入！

2016年12月16日 19:39 | # | [引用](#)

zz 说：

2011年的东西，现在才了解到这块~厉害~

2016年12月20日 15:20 | # | [引用](#)

Canal 说：

引用besthyhy的发言：

非常不错！但我不得不指出文中的错误。URI是资源标识符，URL才是统一资源定位符。二者区别很大。

2017年1月 3日 14:30 | # | 引用

两科威 说：

hello

2017年1月10日 16:25 | # | 引用

pj_zhong@163.com 说：

多谢你帮理清了一些概念，谢谢！！

2017年1月16日 15:07 | # | 引用

达洧 说：

引用徐杰的发言：

关注博主的博客有一段时间了。我想问下博主这个博文网站用什么技术实现的。最近在学网站开发语言，博主能否说的详细点，感激不尽！

是MT博客程序吧

2017年1月20日 22:22 | # | 引用

萧然泪下 说：

阮老师辛苦了

下雨天了怎么办 说：

URI (统一资源定位符) 这个错了吧

URI 应该是“统一资源标识符”

URL 才是“统一资源定位符”

2017年2月20日 11:47 | <#> | [引用](#)

123 说：

说实话，写的易懂！

2017年2月20日 14:39 | <#> | [引用](#)

周睿 说：

可以这样理解吗？RESTful可以说是一种设计规范，根据这种规范设计的程序，可以说它是RESTful框架程序，或者是运用了RESTful思想

2017年2月23日 15:04 | <#> | [引用](#)

alan 说：

终于稍微清楚了一点了。博主写的很好！！还有各种例子。

2017年3月10日 18:02 | <#> | [引用](#)

wk 说：

阮老师好。get用来获取资源，但有一种资源的获取需要用户验证权限，但我又不想将用户验证信息（如：用户名，密码）暴露在get的url上所以我选择了用post方法，这样符合规范吗？有没有其它比较好的解决方法？

2017年3月15日 15:51 | <#> | [引用](#)

ww 说：

引用下雨天了怎么办的发言：

URI（统一资源定位符） 这个错了吧

URI 应该是“统一资源标识符”

URL 才是“统一资源定位符”

又开始拿先有鸡还是先有蛋这种话题扯皮，URI和URL本来就是历史BUG，无所谓的小事

2017年3月15日 16:37 | # | 引用

aaa 说：

关注很久了，感谢博主的默默奉献

2017年4月 6日 14:51 | # | 引用

no1xzy 说：

引用我是猪我怕谁的发言：

盲目的把某一个人定出来的东西的适用场景扩大化到适用任何的场景，本来就是一个浮云。

把一种风格当成一种标准，从而脱离实际的项目也数不胜数，恐怕RESTful就是其中一个。

理想是美好的，现实是残酷的。

这篇文章也就第一段话是有点用的。

但是即便在看到这篇文章之前，我自己能随时敲得出这样的观点。

正因为如此，所以后边的文章和起源，让我看了就是觉得没有必要太过执着哪一种一定是对的。

如果第一段是为了后边的洗脑和盲目崇拜而设立的，就应该把这段话去掉。

再看看后边的内容，谁还会觉得这篇文章有水准？不信就试试看，把第一段话去掉，重新读读这篇文章看看。

本来设计风格，就可以是一种约定嘛。对吧，看得懂就行了。

非然

我从开头就没看第一段话，并且第一段话反而没有任何用

2017年4月17日 03:13 | # | 引用

AnLuoRidge 说：

引用大地的发言：

博主之后的《RESTful API 设计指南》一文，已经纠正此博文中“另一个设计误区，就是在URI中加入版本号”的观点，还请及时修改，以免误导大家吧。

是啊，我去读链接里的那篇文章后发现，作者并没有指出在 URL 里写版本号是个误区。

2017年4月28日 17:55 | # | 引用

dotnba 说：

@我是猪我怕谁：

的确，有点过于强迫症或者处女座了

2017年6月 1日 11:09 | # | 引用

太名 说：

引用liangjixun的发言：

你好，你的这篇文章说把版本号放入url中是设计误区，然后你的另一篇文章
http://www.ruanyifeng.com/blog/2014/05/restful_api.html
又说应该将API的版本号放入URL，到底哪个是对的

明显你提到的链接是新文章，你说哪个是对？

2017年6月 8日 12:08 | # | 引用

果冻 说：

这篇说不把版本号放在uri里，下一篇又说要放在uri里。。。

http://www.ruanyifeng.com/blog/2014/05/restful_api.html

2017年7月19日 10:51 | # | 引用

肖宗杰 说：

Representational State Transfer 我的翻译：对资源的访问状态的变化通过url的变化表述出来，或者，语义化所访问资源的变化状态

2017年9月20日 11:28 | # | 引用

jian 说：

这个东西没有自己做过网站真的好难理解。。。

2017年10月27日 11:13 | # | 引用

Leniy 说：

@linyang：

其实，你们说的这些，还有别的因素，网络应用不仅仅是用户和服务器，需要考虑到ISP的支持。不说TCP/IP模型，单考虑OSI，你们只想到了5-7层，却忘记了3、4层的业务。

另外一点，RESTful是思路而不是标准，把语言和版本号写入uri中，也是这种思路的一种实现方式。强行要求用原生的uri，只是吹毛求疵罢了

2018年1月11日 14:08 | # | 引用

时过境迁 说：

写的很好，是在看restful api设计指南引导过来的，受益匪浅。

2018年1月24日 10:13 | # | 引用

王同学 说：

阮老师，这里转态转化中的“转态”是指什么？？面试的时候被问到，瞬间蒙了。

2018年3月 5日 19:10 | # | 引用

夏渝 说：

“互联网通信协议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，让服务器端发生"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"表现层状态转化"。”

这句话有误，应该是“这意味着，所有的状态都保存在客户端。”

其它部分楼主的阐述非常赞了。

2018年4月11日 15:29 | # | 引用

我要发表看法

您的留言（HTML标签部分可用）

您的大名：

«-必填

电子邮件：

«-必填，不公开

个人网址：

«-我信任你，不会填写广告链接

记住个人信息？ ☐

发表

«- 点击按钮