



维基百科
自由的百科全书

首页
分类索引
特色内容
新闻动态
最近更改
随机条目

帮助
帮助
维基社群
方针与指引
互助客栈
知识问答
字词转换
IRC即时聊天
联络我们
关于维基百科
资助维基百科

工具
链入页面
相关更改
上传文件
特殊页面
可打印版本
固定链接
页面信息
维基数据项
引用本页
左侧路顶连接

其他语言
Deutsch
English
Español
Français
Bahasa Indonesia
日本語
한국어
Português
Русский
还有15种语言
编辑链接

条目 讨论 大陆简体

阅读 编辑 查看历史 搜索维基百科

维基台北写作聚于每月第二个礼拜六举行，欢迎报名参与

[关闭]

REST [编辑]

维基百科，自由的百科全书

REST（英文：**R**epresentational **S**tate **T**ransfer，又称**具象状态传输**）是Roy Thomas Fielding博士于2000年在他的博士论文^[1]中提出出来的一种**万维网软件架构**风格，目的是便于不同软件/程序在网络（例如互联网）中互相传递信息。

目前在三种主流的**Web服务**实现方案中，因为REST模式与复杂的**SOAP**和**XML-RPC**相比更加简洁，越来越多的web服务开始采用REST风格设计和实现。例如，**Amazon.com**提供接近REST风格的Web服务执行图书查询；**雅虎**提供的Web服务也是REST风格的。

| |
|--|
| <div></div> <div><div> </div><div> 目录 [隐藏]</div><div><div> </div><div><div>1 要点及标准</div><div> </div><div><div> </div><div>1.1 REST的要求</div><div> </div><div>1.2 具体说明</div><div> </div><div>1.3 关于状态</div><div> </div><div>1.4 应用于Web服务</div></div></div><div>2 实现举例</div><div>3 REST的优点</div><div>4 实现</div><div>5 参考文献</div><div> </div><div>5.1 引用</div><div> </div><div>5.2 网页</div></div></div> |
|--|

要点及标准 [编辑]

需要注意的是，REST是设计风格而**不是**标准。REST通常基于使用**HTTP**、**URI**、和**XML**以及**HTML**这些现有的广泛流行的协议和标准。

- 资源是由URI来指定。
- 对资源的操作包括获取、创建、修改和删除资源，这些操作正好对应HTTP协议提供的GET、POST、PUT和DELETE方法。
- 通过操作资源的表现形式来操作资源。
- 资源的表现形式则是XML或者HTML，取决于读者是机器还是人，是消费web服务的客户软件还是web浏览器。当然也可以是任何其他的格式。

REST的要求 [编辑]

- 客户端和服务器结构
- 连接协议具有无状态性
- 能够利用Cache机制增进性能
- 一致性的操作界面
- 层次化的系统
- 所需代码 - Javascript（可选）

具体说明 [编辑]

REST架构风格最重要的架构约束有6个^[2]：

- 客户-服务器 (Client-Server)
 - 通信只能由客户端单方面发起，表现为请求-响应的形式。
- 无状态 (Stateless)
 - 通信的会话状态 (Session State) 应该全部由客户端负责维护。
- 缓存 (Cache)
 - 响应内容可以在通信链的某处被缓存，以改善网络效率。
- 统一接口 (Uniform Interface)
 - 通信链的组件之间通过统一的接口相互通信，以提高交互的可见性。
- 分层系统 (Layered System)
 - 通过限制组件的行为（即每个组件只能“看到”与其交互的紧邻层），将架构分解为若干等级的层。
- 按需代码 (Code-On-Demand, 可选)
 - 支持通过下载并执行一些代码（例如Java Applet、Flash或JavaScript），对客户端的功能进行扩展。

关于状态 [编辑]

应该注意区别应用的状态和连接协议的状态。HTTP连接是无状态的（也就是不记录每个连接的信息），而REST传输会包含应用的所有状态信息，因此可以大幅降低对HTTP连接的重复请求资源消耗。

应用于Web服务 [编辑]

符合REST设计风格的Web API称为**RESTful API**。它从以下三个方面资源进行定义：

- 直观简短的资源地址：URI，比如：http://example.com/resources/。
- 传输的资源：Web服务接受与返回的**互联网媒体类型**，比如：**JSON**、**XML**、**YAML**等。
- 对资源的操作：Web服务在该资源上所支持的一系列**请求方法**（比如：POST、GET、PUT或DELETE）。

下表列出了在实现RESTful API时HTTP请求方法的典型用途。

| HTTP请求方法在RESTful API中的典型应用 ^{[3]} | | | | |
|---|--|-----------------------------------|---|------------------|
| 资源 | GET | PUT | POST | DELETE |
| 一组资源的URI，比如 <div>http://example.com/resources/</div> | 列出 URI，以及该资源组中每个资源的详细信息(后者可选)。 | 使用给定的一组资源 替换 当前整组资源。 | 在本组资源中 创建/追加 一个新的资源。该操作往往返回新资源的URL。 | 删除 整组资源。 |
| 单个资源的URI，比如 <div>http://example.com/resources/142</div> | 获取 指定的资源的详细信息，格式可以自选一个合适的网络媒体类型（比如：XML、JSON等） | 替换/创建 指定的资源。并将其追加到相应的资源组中。 | 把指定的资源当做一个资源组，并在其下 创建/追加 一个新的元素，使其隶属于当前资源。 | 删除 指定的元素。 |

PUT和DELETE方法是**幂等方法**。GET方法是**安全方法**（不会对服务器端有修改，因此当然也是幂等的）。

不像基于**SOAP**的Web服务，RESTful Web服务并没有“正式”的标准^[4]。这是因为REST是一种架构，而SOAP只是一个协议。虽然REST不是一个标准，但大部分RESTful Web服务实现会使用**HTTP**、**URI**、**JSON**和**XML**等各种标准。

实现举例 [编辑]

例如，一个简单的网络商店应用，列举所有商品，

```
GET http://www.store.com/products
```

呈现某一件商品，

```
GET http://www.store.com/product/12345
```

下单购买，

```
POST http://www.store.com/order
<purchase-order>
  <item> ... </item>
</purchase-order>
```

REST的优点 [编辑]

- 可更高效利用缓存来提高响应速度
- 通讯本身的无状态性可以让不同的服务器的处理一系列请求中的不同请求，提高服务器的扩展性
- 浏览器即可作为客户端，简化软件需求
- 相对于其他叠加在**HTTP协议**之上的机制，REST的软件依赖性更小
- 不需要额外的资源发现机制
- 在软件技术演进中的长期的兼容性更好

实现 [编辑]

- Ruby on Rails**1.2以后的版本支持**REST model**。
- JBoss RESTEasy**JBoss的REST实现
- Node.js RESTful API**Node.js实现RESTful API

参考文献 [编辑]

引用 [编辑]

- ↑ Fielding, Roy Thomas. **Chapter 5: Representational State Transfer (REST)**. Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). University of California, Irvine. 2000. "This chapter introduced the Representational State Transfer (REST) architectural style for distributed hypermedia systems. REST provides a set of architectural constraints that, when applied as a whole, emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems."
- ↑ **理解RESTful**.
- ↑ Richardson, Leonard; *Ruby, Sam*, RESTful Web Services, O'Reilly, 2007 ((May 8, 2007)), ISBN 0596529260
- ↑ Elkstein, M. *What is REST?*. Retrieved on 2009-07-04.

网页 [编辑]

- http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm Roy Thomas Fielding的博士论文《Architectural Styles and the Design of Network-based Software Architectures》
- http://www.xml.com/pub/a/2002/02/06/rest.html 第二代web服务，Paul Prescod。

| <div>查 · 论 · 编</div> | 网页设计 | | [隐藏] |
|--|------|---|---------------------|
| 用户端 | 语言 | DHTML (HTML、CSS、JavaScript) · HTML5 · VBScript (IE专用) · Java applet · ActionScript (用于Flash) | |
| | 技术 | 网页亲和力(无障碍网页检测) · 无表格网页设计 · 响应式网页设计 · MVVM · WYSIWYG · 容错 (优雅降级) · CSS Sprite · DOM · WebGL · MathML | |
| 服务器端 | 语言 | ASP · ASP.NET (C#, VB.NET) · JSP (Java) · PHP · CGI · ColdFusion · Perl · Python · Ruby · Smalltalk · Node.js (JavaScript) | |
| | 技术 | Web应用框架 · AJAX · MVC · 单页应用 · CRUD · REST | |
| 应用 | | 网志 · 网络论坛 · 企业门户网站 · 电子商务 · 内容管理系统 · 电子出版 · 数位下载 · 网页游戏 | |
| 其他 | | 网页程式设计 · 网络应用程序 · 网页浏览器比较 · 搜索引擎最佳化 | |
| 规范控制 | | GND: 7592728-7 | |

分类：软件工程 | Web服务 | Web 2.0新词

本页面最后修订于2016年12月29日 (星期四) 08:29。

本站的全部文字在知识共享 署名-相同方式共享 3.0协议之条款下提供，附加条款亦可能应用（请参阅使用条款）。
Wikipedia®和维基百科标志是**维基媒体基金会**的注册商标；维基™是维基媒体基金会的商标。
维基媒体基金会是在美国佛罗里达州登记的501(c)(3)免税、非营利、慈善机构。

隐私政策 关于维基百科 免责声明 开发者 Cookie声明 手机版视图

