具象状态传输

维基百科，自由的百科全书

具象状态传输（REST，英文：**Representational State Transfer**）是Roy Thomas Fielding博士于2000年在他的博士论文[1]中提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服务执行图书查詢；雅虎提供的Web服务也是REST风格的。

目录

要点及标准

需要注意的是，具象状态传输是设计风格而**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。

具象状态传输的要求

- 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
- 所需代碼 - Javascript（可選）

具体说明

具象状态传输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

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。

應用於Web服務

符合具象状态传输設計風格的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 `http://example.com/resources/`	**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 `http://example.com/resources/142`	**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，因此當然也是幂等的）。

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

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
```

呈現某一件商品，

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

下单购买，

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

具象状态传输优点

- 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

- 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實現RESTful API

参考文献

引用

1. 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."
2. 理解RESTful.
3. Richardson, Leonard; Ruby, Sam, RESTful Web Services, O'Reilly, 2007 （May 8, 2007）, ISBN 0596529260
4. Elkstein, M. *What is REST?* (http://rest.elkstein.org/2008/02/what-is-rest.html) Retrieved on 2009-07-04.

网页

- 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》
- https://web.archive.org/web/20080808122838/http://www.xml.com/pub/a/2002/02/06/rest.html第二代web服务，Paul Prescod。
