# Restful接口简述

|  |  |  |  |
| --- | --- | --- | --- |
| **操作人** | **操作时间** | **操作内容** | **备注** |
| 胡贺东 | 20170329 | 创建 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 什么是Web Service

\* [中国天气网Web Service](http://www.weather.com.cn/data/sk/101010100.html)(免费)

\* [百度APIStore](http://apistore.baidu.com/)(收费)

## Web Service相关标准(specification)

# ### 1. SOAP

\*\*Simple Object Access Protocol\*\*，基于HTTP/SMTP/TCP/... + XML，1998年为 M$ 设计的。

不够Simple。不过反映了 Web Service 的发展进程。

# ### 2. WSDL

\*\*Web Services Description Language\*\*，用于描述接口，就像 JAVA 的方法签名(method signature)。

由 IBM/M$/Ariba 2000年开发出来描述其 SOAP toolkit，\*\*WSDL 2.0\*\*在2007年6月成为\*\*W3C\*\*推荐标准(recommendation)。

# ### 3. REST

\*\*REpresentational State Transfer\*\* 具象狀態傳輸/表述性状态转移/表现层状态转化

## 需要了解的Web standards

## ### 1. HTTP

[旧版RFC2616](https://tools.ietf.org/html/rfc2616) 1999/06

[新版RFC7230](https://tools.ietf.org/html/rfc7230) 2014/06

### 2. URI

[旧版RFC2396](https://tools.ietf.org/html/rfc2396) 1998/08

[新版RFC3986](https://tools.ietf.org/html/rfc3986) 2005/01

## ## REST解释

## ### 1. Resources - 资源

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"。

Resource举几个好栗子和坏栗子：

好的

```

users

orders

tags/1/skus

banners

```

坏的

```

/Credit/Status

/Credit/MemerInfo

/Credit/SaveMemerInfo

/Credit/JobInfo

/Credit/SaveJobInfo

/area/{id}&{level}

/coupon\_status/{status} 根据用户id获取该用户优惠券并分类（分类：0:未使用，1:已过期，2:已使用）

/coupon\_use/{pid}&{cateId} 根据商品 + 用户 获取该商品是否有可使用的优惠券

/promotionPrizeList/{id} 活动奖品列表

/getPrizeList/{id} 获奖名单及奖品列表

```

特殊的

```

/users/signup

/users/signout

/users/verify\_code

```

要点：

\* 英文名词复数，resource 和 数据库表名，前端 class 名等等，统一术语

\* 注意不要有动词，动词是由 GET/POST/PUT/DELETE/PATCH 完成的

\* resource id 一般放在 path

\* 可读性高(human-readable URIs)

### 2. Representation - 表现层

>"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。

我们用到了最基本的 \*\*Content-Type\*\* ，高端一点的 \*\*Accept\*\* 暂时不支持，如果要实现接口的版本，则应该在 Accept 里指定。

这两个概念都属于 HTTP 协议。做 Web应用 开发对 HTTP 1.1 协议不了解的话，还是改行做单机软件比较合适。

### 3. State Transfer - 状态转化

互联网通信协议HTTP协议，是一个无状态协议(stateless protocol: 每次请求作为不相关的独立事物 independent transaction)，很像概率学的\*\*独立不相关\*\*，如抛硬币，如抽奖。

大部分底层协议都是无状态的，例如\*\*IP\*\*协议。

HTTP协议无状态的有点：\*\*简单可靠\*\*，缺点：需要在每次请求中，增加额外的信息，比如请求头中的`Cookie: JSESSIONID=xxx`。

HTTP协议无状态，不是说我们的应用无状态，恰恰相反，大部分有用户的应用都是有状态的。所有主流HTTP服务器都内置实现了各种Session管理功能，从而在 HTTP 协议之上构建了一种有状态协议(Stateful protocol)。

既然我们的应用是有状态的，那就可以实现状态转化，REST通过充分利用HTTP协议的四个动词来实现 CRUD 操作：

**1. GET Read**

**2. POST Create**

**3. PUT Update**

**4. DELETE Delete**

GET 是 “幂等”（idempotent） 的，`Math.pow(1, 100) = ?`<br>

还有哪个是幂等的？

其实除了这几种请求方法(Method)外，HTTP 1.1 协议还定义了很多请求方法，如：

\* HEAD response只返回header`curl -I "http://www.baidu.com/"`

\* OPTIONS 跨域发送非 GET 请求时，需要先发送一个预请求(preflight)，看服务端是否允许客户端跨域请求此Service

\* CONNECT

\* TRACE

响应的返回码含义参考[rfc7231 6.1章节](https://tools.ietf.org/html/rfc7231#section-6.1)：

**\* 200 OK**

**\* 204 No Content 可用于OK，但是无数据的情况，比如一个用户没有任何订单**

**\* 400 Bad Request 比如缺少必须参数，或应该传浮点型的，包含了非数字的字符**

**\* 401 Unauthorized 未登录**

**\* 403 Forbidden 登录了，但无权限访问此资源**

**\* 405 Method Not Allowed**

**\* 500 Internal Server Error JAVA异常**

**\* 502 Bad Gateway 网关错误，比如 Nginx 反向代理到 Tomcat ，但是 Tomcat 未启动**

**\* 504 Gateway Timeout 网关超时，比如我们的服务需要请求微信服务器这种外部资源，但是网络特别卡，超过30秒或1分钟没有返回给 Nginx ，Nginx通常会报 504**

## 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，对服务器端资源进行操作，实现"表现层状态转化"。

## SprintMVC对RESTful的支持太友好了

```

@Controller

@RequestMapping(value = "/resources", produces = "application/json")

class ResourceController {

@RequestMapping(value = "/{id}", method = RequestMethod.GET)

Response getDetail(@PathVariable("id") Integer id);

@RequestMapping(method = RequestMethod.GET)

Response getAll();

@RequestMapping(method = RequestMethod.POST, consumes = "application/json")

Response post(@RequestBody ResourceEntity resourceEntity);

@RequestMapping(value = "/{id}", method = RequestMethod.PUT, consumes = "application/json")

Response put(@PathVariable("id") Integer id, @RequestBody ResourceVO resourceVO);

@RequestMapping(value = "/{id}", method = RequestMethod.DELETE)

Response delete(@PathVariable("id") Integer id);

}

```

## JS/jQuery/AngularJS对RESTful的支持太友好了

\* PUT/POST 采用JSON传递数据，而不是Form表单

\* 几乎所有响应都是返回JSON

\* 注意跨域: [XMLHttpRequest.withCredentials](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/withCredentials)

## 网络请求分析工具

1. Postman 0.8.4.17 - REST Client

2. [Chrome DevTools](https://developers.google.com/web/tools/chrome-devtools/)

3. Charles 3.11.4

4. JSONView 0.0.32.2

5. EditThisCookie 1.4.1

上述1、4、5扩展可在共享目录`//192.168.17.205/xinyuangong/Chrome Extensions`下载到，或者使用360浏览器也可以下载

# ## 工具

1. [swagger.io](http://swagger.io/) Swagger is a simple yet powerful representation of your RESTful API

2. [I/O Docs](https://github.com/mashery/iodocs)

# ## 历史

> The term REST was defined by Roy T. Fielding in his PhD thesis (you might actually want to follow that link — it’s quite readable, for a dissertation at least).

2000年提出的，论文单词能看懂，但比较抽象。Roy 是 HTTP 协议和 URI 协议的作者之一。

# ## 参考

1. [理解RESTful架构](http://www.ruanyifeng.com/blog/2011/09/restful) 阮一峰 2011-09-12

1. [A Brief Introduction to REST](https://www.infoq.com/articles/rest-introduction) Stefan Tilkov 2007-12-10

3. [SOAP](https://en.wikipedia.org/wiki/SOAP)

4. [Stateless protocol](https://en.wikipedia.org/wiki/Stateless\_protocol)