**Restful API**

参考的网址：

* RESTful API 最佳实践：<http://www.ruanyifeng.com/blog/2018/10/restful-api-best-practices.html>
* SpringMVC+RestFul详细示例实战教程（实现跨域访问）：<https://juejin.im/post/5b1ddbc96fb9a01e5e3d3f79>

0、RESTful是目前最流行的 API 设计规范，**用于 Web 数据接口的设计**。

1、URL设计

1.1 动词 + 宾语

**RESTful 的核心思想就是，客户端发出的数据操作指令都是"动词 + 宾语"的结构。**比如，GET /articles这个命令，GET是动词，/articles是宾语。

动词通常就是五种 HTTP 方法，对应 CRUD 操作。

**GET：读取（Read）**

**POST：新建（Create）**

**PUT：更新（Update）**

**PATCH：更新（Update），通常是部分更新**

**DELETE：删除（Delete）**

根据 HTTP 规范，动词一律大写。

1.2 动词的覆盖

有些客户端只能使用GET和POST这两种方法。**服务器必须接受POST模拟其他三个方法（PUT、PATCH、DELETE）**。

这时，客户端发出的 HTTP 请求，要加上X-HTTP-Method-Override属性，告诉服务器应该使用哪一个动词，覆盖POST方法。

**POST /api/Person/4 HTTP/1.1**

**X-HTTP-Method-Override: PUT**

上面代码中，X-HTTP-Method-Override指定本次请求的方法是PUT，而不是POST。

1.3 宾语必须是名词

宾语就是 API 的 URL，是 HTTP 动词作用的对象。它应该是名词，不能是动词。比如，/articles这个 URL 就是正确的，而下面的 URL 不是名词，所以都是错误的。

**/getAllCars**

**/createNewCar**

**/deleteAllRedCars**

1.4 复数 URL

既然 URL 是名词，那么应该使用复数，还是单数？

这没有统一的规定，但是常见的操作是读取一个集合，比如GET /articles（读取所有文章），这里明显应该是复数。

为了统一起见，建议都使用复数 URL，比如GET /articles/2要好于GET /article/2。

1.5 避免多级 URL

常见的情况是，资源需要多级分类，因此很容易写出多级的 URL，比如获取某个作者的某一类文章。

**GET /authors/12/categories/2**

这种 URL 不利于扩展，语义也不明确，往往要想一会，才能明白含义。

更好的做法是，除了第一级，其他级别都用查询字符串表达。

**GET /authors/12?categories=2**

下面是另一个例子，查询已发布的文章。你可能会设计成下面的 URL。

**GET /articles/published**

查询字符串的写法明显更好。

**GET /articles?published=true**

2、状态码

2.1 状态码必须精确

客户端的每一次请求，服务器都必须给出回应。**回应包括 HTTP 状态码和数据两部分**。

HTTP 状态码就是一个三位数，分成五个类别。

**1xx：相关信息**

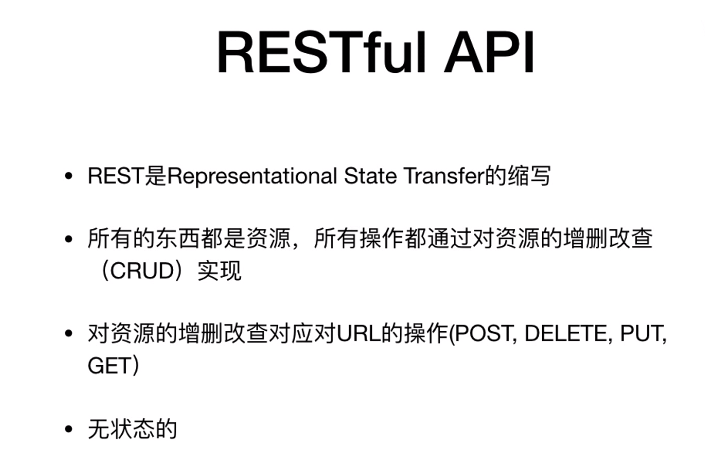
**2xx：操作成功**

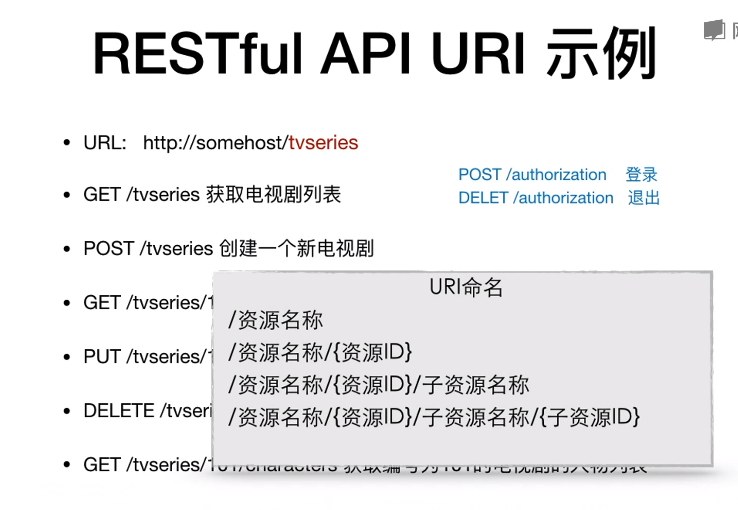
**3xx：重定向**

**4xx：客户端错误**

**5xx：服务器错误**

这五大类总共包含100多种状态码，覆盖了绝大部分可能遇到的情况。每一种状态码都有标准的（或者约定的）解释，客户端只需查看状态码，就可以判断出发生了什么情况，所以服务器应该返回尽可能精确的状态码。



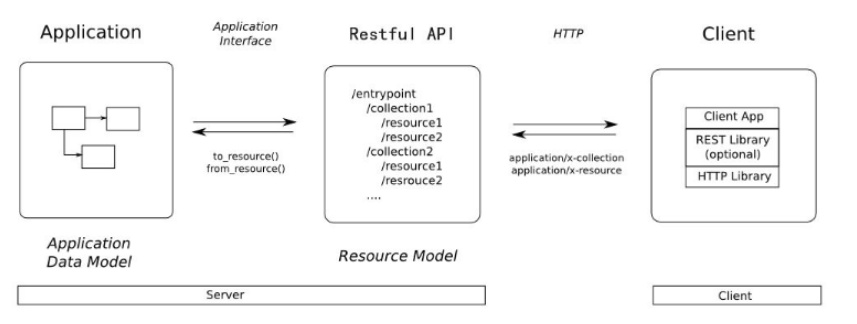


1、REST：Representation State Transfer的缩写【可重新表达的状态迁移—表示层状态迁移】

面向资源（一类数据）编程，（URL）地址即资源。

**对于资源我们的代码中可能的操作只有 增删改查**

**在HTTP协议中对每一个请求URL都会有不同的谓词GET（查）/POST（增）/PUT（改）/PATCH（改）/DELETE（删）**



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。

GET /tickets # 获取ticket列表

GET /tickets/12 # 查看某个具体的ticket

POST /tickets # 新建一个ticket

PUT /tickets/12 # 更新ticket 12.

DELETE /tickets/12 #删除ticekt 12

REST特点如下：

基于HTTP协议

是另一种服务架构

传递是JSON、POX（Plain Old XML）而不是SOAP格式的数据

充分利用HTTP谓词（Verb）

侧重数据的传输，业务逻辑交给客户端自行处理

2、REST 是面向资源的，强调描述应用程序的事物和名词。这样很容易导致的一个结果是我们在设计 RESTful API 时，在 URI 中使用动词。例如：GET /user/getUser/123。正确写法应该是 GET /user/123。