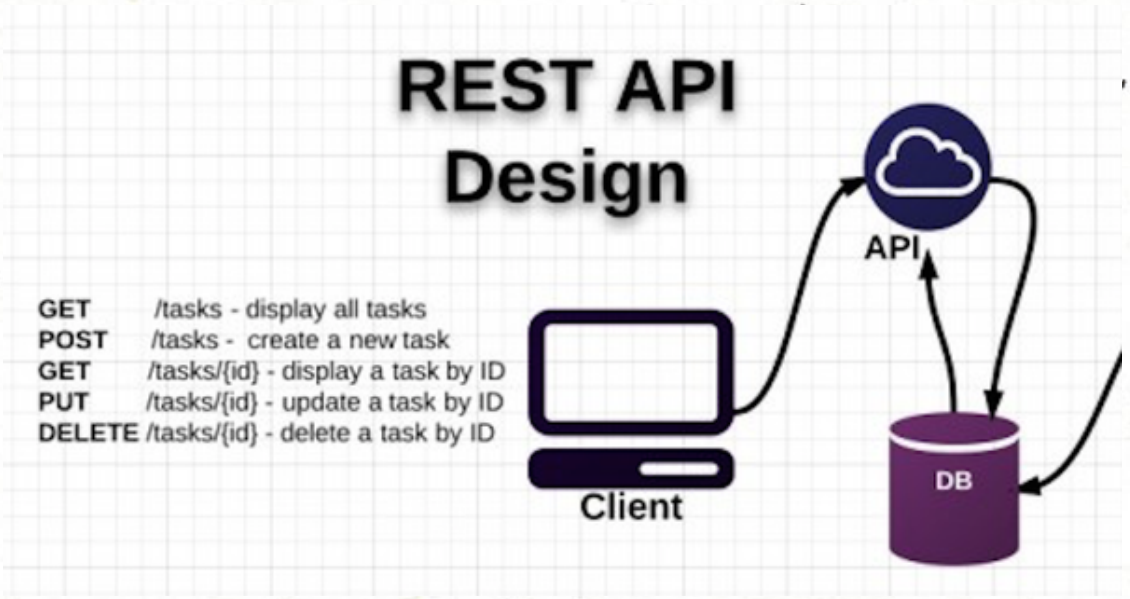
作者：覃超
链接：https://www.zhihu.com/question/27785028/answer/48096396
来源：知乎
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

简单理解:增删改查都是一个地址，具体靠http头部信息判断
星巴克例子:https://link.zhihu.com/?target=http%3A//www.infoq.com/cn/articles/webber-rest-workflow/

相同问题一起答了！
我觉得问题很好，我自己去年创业的时候去学习REST和尝试着设计RESTful API，一直觉得它的文档晦涩难懂，国内也没有找到太好文章。后来一年内反复琢磨了好几遍，和FB+Square的朋友讨论过好几次，有了一个比较清晰的总结。分享如下：
@Ivony
老师的一句话概括很精辟：
URL定位资源，用HTTP动词（GET,POST,DELETE,DETC）描述操作。
--- **简洁版** ---
0. REST不是"rest"这个单词，而是几个单词缩写。但即使那几个单词说出来，也无法理解在说什么 -_-!!（不是要贬低人，是我自己也理解困难）；
1. REST描述的是在网络中client和server的一种交互形式；REST本身不实用，实用的是如何设计 RESTful API（REST风格的网络接口）；
2. Server提供的RESTful API中，URL中只使用名词来指定资源，原则上不使用动词。“资源”是REST架构或者说整个网络处理的核心。比如：
    http://api.qc.com/v1/newsfeed: 获取某人的新鲜;
    http://api.qc.com/v1/friends: 获取某人的好友列表;
    http://api.qc.com/v1/profile: 获取某人的详细信息;
3. 用HTTP协议里的动词来实现资源的添加，修改，删除等操作。即通过HTTP动词来实现资源的状态扭转：
    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。
比如：
    DELETE http://api.qc.com/v1/friends: 删除某人的好友（在http parameter指定好友id）
    POST http://api.qc.com/v1/friends: 添加好友
    UPDATE http://api.qc.com/v1/profile: 更新个人资料

禁止使用： GET http://api.qc.com/v1/deleteFriend

图例：



4. Server和Client之间传递某资源的一个表现形式，比如用JSON，XML传输文本，或者用JPG，WebP传输图片等。当然还可以压缩HTTP传输时的数据（on-wire data compression）。

5. 用 HTTP Status Code传递Server的状态信息。比如最常用的 200 表示成功，500 表示Server内部错误等。

主要信息就这么点。最后是要解放思想，Web端不再用之前典型的PHP或JSP架构，而是改为前段渲染和附带处理简单的商务逻辑（比如AngularJS或者BackBone的一些样例）。Web端和Server只使用上述定义的API来传递数据和改变数据状态。格式一般是JSON。iOS和Android同理可得。由此可见，Web，iOS，Android和第三方开发者变为平等的角色通过一套API来共同消费Server提供的服务。

--- **详细版** ---

先说REST名称

REST：REpresentational State Transfer = 直接翻译：表现层状态转移。这个中文直译经常出现在很多博客中。尼玛谁听得懂“表现层状态转移”？这是人话吗？

首先，之所以晦涩是因为前面主语被去掉了，全称是 Resource Representational State Transfer：通俗来讲就是：资源在网络中以某种表现形式进行状态转移。分解开来：

Resource：资源，即数据（前面说过网络的核心）。比如 newsfeed，friends 等；

Representational：某种表现形式，比如用JSON，XML，JPEG等；

State Transfer：状态变化。通过HTTP动词实现。

REST的出处

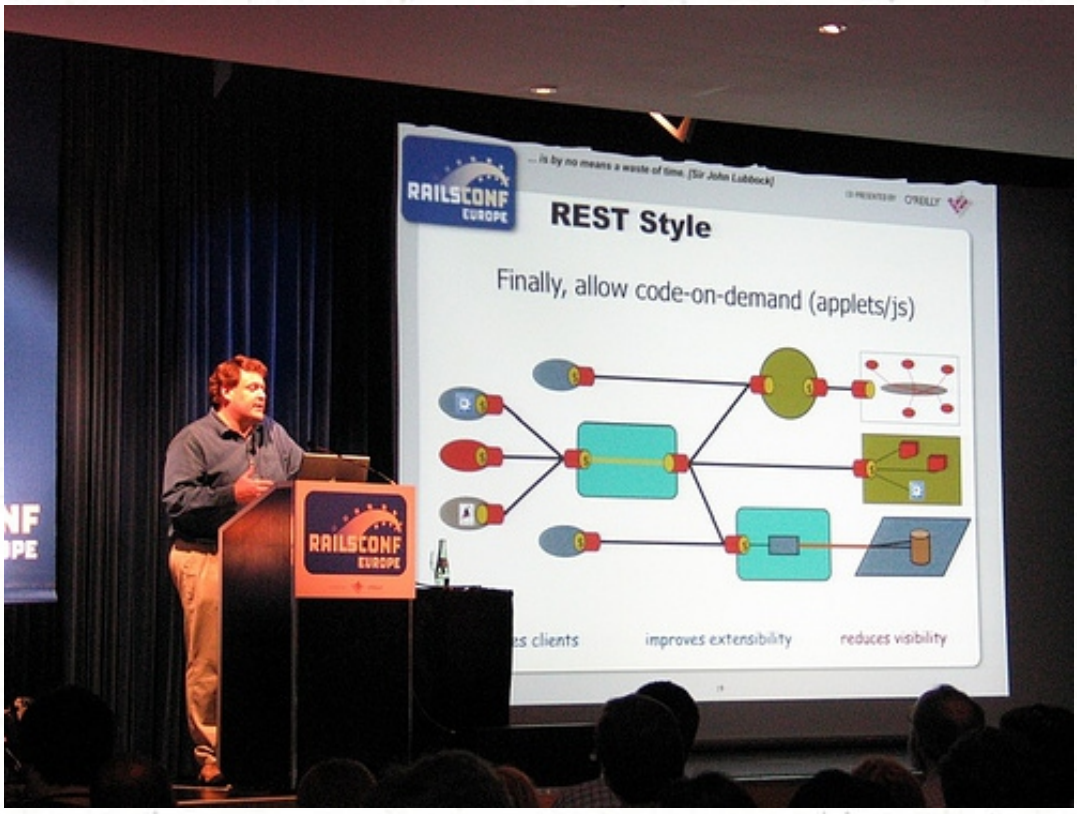
Roy Fielding的毕业论文。这哥们参与设计HTTP协议，也是Apache Web Server项目（可惜现在已经是 nginx 的天下）的co-founder。PhD的毕业学校是 UC Irvine，Irvine在加州，有着充裕的阳光和美丽的海滩，是著名的富人区。Oculus VR 的总部就坐落于此（虚拟现实眼镜，被FB收购，CTO为Quake和Doom的作者 John Carmack）。

众说周知，论文都是晦涩难懂的。当年在CMU读书的时候，很多课程都会安排每周两篇的Paper review。现在回想起来每次写Paper review都是我最为痛苦的时候。REST这篇博士论文毫无疑问更甚。

论文地址：[Architectural Styles and the Design of Network-based Software Architectures]

REST章节：[Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)]

REST那章我初读了，整个论文没有读完 =_=



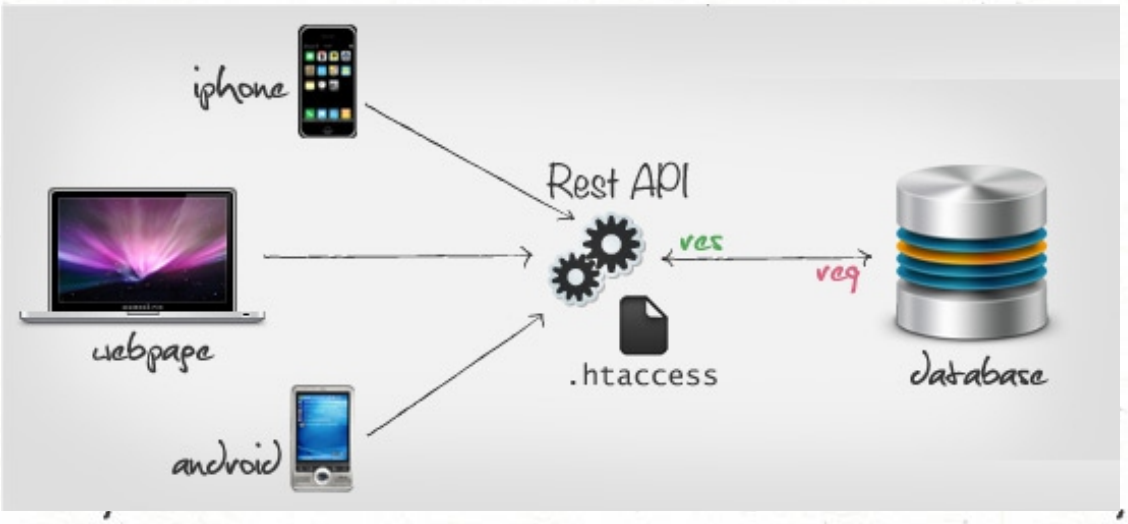
RESTful API

实用的是如何正确地理解 RESTful架构和设计好RESTful API。

首先为什么要用RESTful结构呢？

大家都知道"古代"网页都是前端后端融在一起的，比如之前的PHP，JSP等。在之前的桌面时代问题不大，但是近年来移动互联网的发展，各种类型的Client层出不穷，

RESTful可以通过一套统一的接口为 Web，iOS和Android提供服务。另外对于广大平台来说，比如Facebook platform，微博开放平台，微信公共平台等，它们不需要有显式的前端，只需要一套提供服务的接口，于是RESTful更是它们最好的选择。在RESTful架构下：



Server的API如何设计才满足RESTful要求?

首先是简洁版里面的那几点。外加一些附带的 best practices：

1. URL root:

    https://example.org/api/v1/*

    https://api.example.com/v1/*

2. API versioning:

可以放在URL里面，也可以用HTTP的header：

    /api/v1/

3. URI使用名词而不是动词，且推荐用复数。

BAD

- /getProducts
- /listOrders
- /retrieveClientByOrder?orderId=1

GOOD

- GET /products : will return the list of all products
- POST /products : will add a product to the collection
- GET /products/4 : will retrieve product #4
- PATCH/PUT /products/4 : will update product #4

4. 保证 HEAD 和 GET 方法是安全的，不会对资源状态有所改变（污染）。比如严格杜绝如下情况：

    GET /deleteProduct?id=1

5. 资源的地址推荐用嵌套结构。比如：

GET /friends/10375923/profile
UPDATE /profile/primaryAddress/city

6. 警惕返回结果的大小。如果过大，及时进行分页（pagination）或者加入限制（limit）。HTTP协议支持分页（Pagination）操作，在Header中使用 Link 即可。
7. 使用正确的HTTP Status Code表示访问状态：HTTP/1.1: Status Code Definitions
8. 在返回结果用明确易懂的文本（String。注意返回的错误是要给人看的，避免用 1001 这种错误信息），而且适当地加入注释。
9. 关于安全：自己的接口就用https，加上一个key做一次hash放在最后即可。考虑到国情，HTTPS在无线网络里不稳定，可以使用Application Level的加密手段把整个HTTP的payload加密。有兴趣的朋友可以用手机连上电脑的共享Wi-Fi，然后用Charles监听微信的网络请求（发照片或者刷朋友圈）。
如果是平台的API，可以用成熟但是复杂的OAuth2，新浪微博这篇：授权机制说明

各端的具体实现

如上面的图所示，Server统一提供一套RESTful API，web+ios+android作为同等公民调用API。各端发展到现在，都有一套比较成熟的框架来帮开发者事半功倍。

-- Server --
推荐： Spring MVC 或者 Jersey 或者 Play Framework
教程：
Getting Started · Building a RESTful Web Service
-- Android --
推荐： RetroFit（Retrofit）或者 Volley（mcxiaoke/android-volley · GitHub Google官方的被block，就不贴了）
教程：
Retrofit 1 Getting Started and Create an Android Client
快速Android开发系列网络篇之Retrofit
-- iOS --
推荐：RestKit（RestKit/RestKit · GitHub）
教程：
Developing RESTful iOS Apps with RestKit
-- Web --
推荐随便搞！可以用重量级的AngularJS，也可以用轻量级 Backbone + jQuery 等。
教程：http://blog.javachen.com/2015/01/06/build-app-with-spring-boot-and-gradle/

参考：

[1]: Some REST best practices
[2]: GitHub API v3
[3]: tlhunter/consumer-centric-api-design · GitHub

最后附带一个彩蛋：
Facebook台球表演: 台球1—在线播放