# 课程介绍

* 1. 什么是REST;（了解）
* 2. RESTful API的设计规范;（掌握）
* 3. HTTP BasicAuth基本原理;（掌握）

1. REST概述
   1. 什么是REST

* 一种软件架构风格、设计风格、而不是标准，只是提供了一组设计原则和约束条件。
* 它主要用于客户端和服务器交互类的软件。
* 基于这个风格设计的软件可以更简洁，更有层次，更易于实现缓存机制等。

REST概念首次在2000年Roy Thomas Fielding的博士论文中出。

REST的全称是：Representational State Transfer（表现层状态转移）。

Fielding是一个什么样的人那，他是HTTP协议（1.0版和1.1版）的主要设计者，Apache服务器软件的作者之一，Apache基金会的第一任主席。所以，他的这篇论文一经发表，就引起了广泛的关注。

本文研究计算机科学两大前沿----软件和网络----的交叉点。长期以来，软件研究主要关注软件设计的分类、设计方法的演化，很少客观地评估不同的设计选择对系统行为的影响。而相反地，网络研究主要关注系统之间通信行为的细节、如何改进特定通信机制的表现，常常忽视了一个事实，那就是改变应用程序的互动风格比改变互动协议，对整体表现有更大的影响。**我这篇文章的写作目的，就是想在符合架构原理的前提下，理解和评估以网络为基础的应用软件的架构设计，得到一个功能强、性能好、适宜通信的架构。**

* + 1. REST的爆发

在REST架构推出的十几年间，它并没有一路高歌的发展，真正的大范围推广是在2013年之后，伴随着移动端的飞速发展，越来越多人的开始意识到，网站即软件，而且是一种新型的软件。

这种"互联网软件采用"客户端/服务器"模式，也就是我们常说的C/S模式，这一切建立在分布式体系上，通过互联网通信，具有高延时，高并发等特点。

网站开发，完全采用软件开发的模式。但传统上，软件和网络是两个不同的领域，很少有交集，软件开发主要针对单机环境，网络则主要研究系统之间的通信。我们需要考虑的是如何开发在互联网环境中使用软件。

* 1. 理解REST

REST的字面意思非常难于理解，其实是少了一个关键的主语，Resource（资源）。

**资源表现层的状态转化标准**

* + 1. 资源（Resource）

所谓“资源”，就是网络上的一个实体，或者说是网络上的一个具体信息（数据）。

它可以是一段文本，一张图片，一首歌曲，一种服务，总之就是一个具体的实例。

我们可以使用一个URI（统一资源定位符）指向它，每种资源对应一个特定的URI。

要获取这个资源，访问它的URI就可以了，因此URI就成了每一个资源的地址或独一无二的识别符。所谓“上网”就是与互联网上一系列的“资源”互动，调用它们的URI。

* + 1. 表现层（Representation）

“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）。

URI只代表资源的抽象，不代表它的形式。严格地说，有些网站最后的”.html“后缀名是不必要的，因为这个后缀表示格式，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请求头的信息中使用Accept和Content-Type字段指定。

* + 1. 状态转换（State Transfer）

访问一个网站，就代表客户端和服务端的一个互动过程。在这个过程中，势必涉及到数据和状态的变化。

互联网通信协议HTTP协议，是一个无状态协议。这意味着，所有的状态都保存在服务端。因此，如果客户端想要操作服务器，就必须通过某种手段，让服务器端发生“状态转换（State Transfer）”。而这种转换是建立在表现层之上的，所以就是“表现层状态转化”。

客户端用到的手段，只能是HTTP协议。具体来说，就是HTTP协议中，四个表示操作方式的动词：GET，POST，PUT，DELETE。

它们分别对应四种基本操作：

* **GET用来获取资源**
* **POST用来新建资源（也可用于更新资源）**
* **PUT用来更新资源**
* **DELETE用来删除资源**
  + 1. 总结
* **到底什么是RESTful架构：**

每一个URI代表一种资源；

客户端和服务器之间，传递这种资源的某种表现层；

客户端通过四个HTTP动词，对服务端资源进行操作，实现“表现层状态转换”；

1. RESTful API设计
   1. 重要概念
      1. 协议

API与用户的通信协议，通常使用HTTP(S)协议。

* + 1. 域名

应该尽量将API部署在专用域名之下。

http://api.baidu.com

如果确定API很简单，不会有大规模扩充，可以考虑放在主域名之下。

http://www.baidu.com/api/

* + 1. 版本

应该将API的版本号放入URL。

http://api.baidu.com/v1/

也有做法是将版本号放在HTTP的头信息中，但不如放在URL中方便和直观。

* + 1. 路径（endpoint）

路径又称”终点“（endpoint），表示API的具体网址。

在RESTful架构中，每个网址代表一种资源（Resource），**所以网址中不能有动词，只能有名词**。

有些标准要求名词与数据库的表格名对应。一般来说，数据库中的表都是同种记录的”集合“（collection），所以API中的名词也应该使用复数。

* 1. HTTP的动词
     1. 常用动词

|  |  |
| --- | --- |
| **动词** | **说明** |
| **GET（SELECT）** | 从服务器取出资源 |
| **POST（CREATE or UPDATE）** | 向服务器创建资源或更新资源 |
| **PUT（UPDATE）** | 向服务器更新资源（客户端提供改变后的完整资源） |
| **PATCH（UPDATE）** | 向服务器更新资源（客户端提供改变的属性） |
| **DELETE（DELETE）** | 从服务器删除资源 |
| **HEAD** | 获取资源的元数据 |
| **OPTIONS** | 获取信息，关于资源的哪些属性是客户端可以改变的 |

对于资源的具体操作类型，由HTTP动词表示，示例：

|  |  |
| --- | --- |
| **HTTP请求头** | **状态说明** |
| **GET /books/** | 查看所有书籍信息； |
| **POST /books/** | 新建书籍 |
| **GET /books/id/** | 查看某一本书信息 |
| **PUT /books/id/** | 更新某本书的信息（需要提供书的全部的信息） |
| **PATCH /books/id/** | 更新某本书的信息（需要提供书变更部分的信息） |
| **DELETE /books/id/** | 删除某本书 |

* + 1. 过滤信息（Filtering）

如果记录数量过多，服务器不可能将它们返回给用户。API应该提供参数，过滤返回结果。

?limit=10

?offset=10

?page=2&per\_page=20

?sortby=name&order=desc

?student\_id=id

参数的设计允许存在冗余，即允许API路径和URL参数偶尔有重复，比如 GET /books/id/ 和 ？book\_id=id

* + 1. 状态码

服务器向用户返回的状态码和提示信息，常见的有：

|  |  |
| --- | --- |
| **响应状态码** | **说明** |
| **200 OK** | [GET]：服务器成功返回用户请求的数据 |
| **201 CREATED** | [POST/PUT/PATCH]：用户新建或修改数据成功 |
| **202 Accepted** | [\*] ：表示一个请求已经进入后台排队（异步任务） |
| **204 NO CONTENT** | [DELETE]：表示数据删除成功 |
| **400 INVALID REQUEST** | [POST/PUT/PATCH]：用户发出的请求有错误 |
| **401 Unauthorized** | [\*] ：表示用户没有权限（令牌，用户名，密码错误） |
| **403 Forbidden** | [\*]：表示用户得到授权，但是访问是被禁止的 |
| **404 NOT FOUND** | [\*]：用户发出的请求针对的是不存在的记录 |
| **406 Not Acceptable** | [\*]：用户请求格式不可得 |
| **410 Gone** | [GET] ：用户请求的资源被永久移除，且不会再得到的 |
| **422 Unprocesable entity** | [POST/PUT/PATCH]：当创建一个对象时，发生一个验证错误 |
| **500 INTERNAL SERVER EROR** | [\*] ：服务器内部发生错误 |

* 1. 错误信息的处理

错误可以分为两种类型：全局错误和本地错误。

全局错误包括：请求了一个不存在的API、无权请求这个API、数据库连接失败、或其他一些没有预期到的、会终止程序运行的服务端错误。这类错误应该由 Web 框架捕获，无需各个 API 处理。

本地错误则和API密切相关，例如表单验证、唯一性检查、或其他可预期的错误。我们需要编写特定代码来捕获这类错误，并抛出一个包含提示信息的全局异常，供 Web 框架捕获并返回给客户端。

在实际开发中，还有一种常见的设计规范：

不考虑改变HTTP返回的状态吗，而只是在返回结果中表明状态和错误信息，大致结构如下：

失败：

{"status":401,"message":"APP不存在，AK有误请检查再重试"}

成功：

{"status":0,"message":"……"}

1. HTTP BasicAuth

在HTTP中，基本认证（Basic access authentication）是一种用来允许网页浏览器或其他客户端程序在请求时提供用户名和口令形式的身份凭证的一种登录验证方式。

* 1. BasicAuth流程

HTTP Basic Auth使用两个HTTP Header实现，分别是WWW-Authenticate和Authorization。

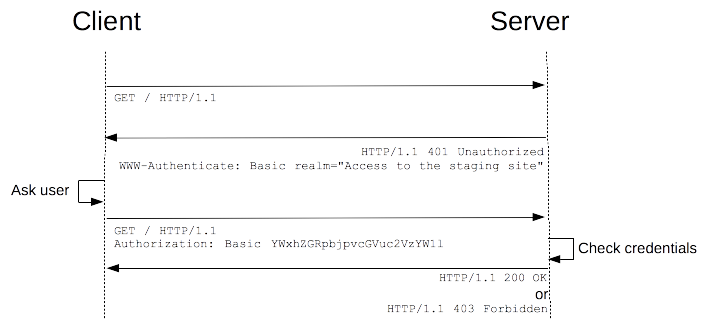
流程如下：

* + - 1. 客户端请求服务器页面，服务器返回401以及WWW-Authenticate: Basic realm="site"。
      2. 浏览器弹出对话框，提示用户输入用户名和密码。
      3. 浏览器再次请求页面，携带

Authorization: Basic <str>，其中，str=base64(username:password)。

4. 服务器返回正常页面。

base64只是一个编码过程，而不是加密过程，因此，HTTP Basic Auth是在明文传输用户名和密码，中间设备很容易通过检查数据包获取用户名和密码。



* 1. 注意点

HTTP Basic Auth是HTTP提供的一种验证方式，因为明文传输用户名和密码，非HTTPS环境下很不安全，一般用的非常少。

但是在某些情况下用一用还是非常方便的，比如，一些静态站点例如文档系统可以使用HTTP Basic Auth进行简单地权限验证。

1. 带登陆验证的设计
2. 课程总结
   1. 重点
      * 1. RESTful API的规范
        2. HTTP认证
   2. 难点
      * 1. HTTP认证
3. 课后练习
   * + 1. 上课演示代码
       2. todo功能的API设计
4. 面试题
   * + 1. 谈一谈你对RESTful API的理解
5. 扩展知识或课外阅读推荐（可选）