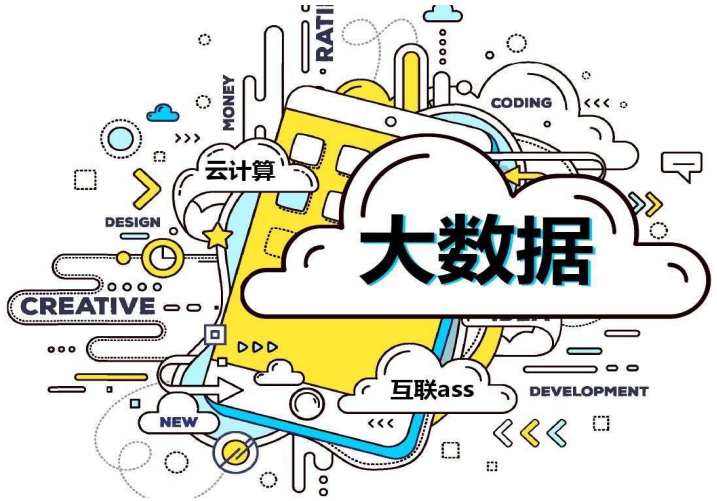
《位置服务网络与大数据》

第五讲-移动通信技术-Restful API

2022-3-14 · 桂林

本讲内容



- RESTful API简介
- Hello World
- 工具调试
- 注意事项

01

RESTful API简介

为什么会有RESTful API?

- 前端设备层出不穷（手机、平板、桌面电脑、其他专用设备……），促使前后端分离，方便不同的前端设备与后端进行通信，导致API架构的流行。
- RESTful架构，因结构清晰、符合标准、易于理解、扩展方便，是目前流行的一种互联网软件架构
- RESTful API是目前比较成熟的一套互联网应用程序的API设计理论

什么是REST?

- 由Roy Thomas Fielding在他2000年的博士论文中提出:
- 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。
- 他对互联网软件的架构原则，定名为REST。

- Roy Thomas Fielding
- 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

如何理解REST?

REST= Representation + State + Transfer 表现层状态转化

- 资源在网络中以某种表现形式进行状态转移：

  Resource：资源

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

  State Transfer：状态转化，通过HTTP动词实现

什么是RESTful架构？

如果一个架构符合REST原则，就称它为RESTful架构。

RESTful架构，是面向资源的架构：

- 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HTTP动词，对服务器端资源进行操作，实现"表现层状态转化"

什么是RESTful API?

符合REST架构设计的API，就是RESTful API

特征:

URL用于定位资源，用HTTP动词描述操作

常见举例:

查询编号为1的图书

[GET] http://api.test.com/v1/books/1

删除编号为1的图书:

[DELETE] http://api.test.com/v1/books/1

URI简介

- URI(Uniform Resource Identifier)：通用资源标识符，它被设计充当可用位置和持久名称。
- 语法规则：大致指向一个层次空间，协议是树根，从左往右每部分是前部分的分支。

例：http:// example.net /site/page ？name=zdp # photo

方案 域名 路径 查询 片段

- 路径：并非一定要采用层次结构，可根据应用程序模型定制路径结构。

资源

- URI规范(RFC 2396)指出：

  “资源可以是任何有标示的东西”。

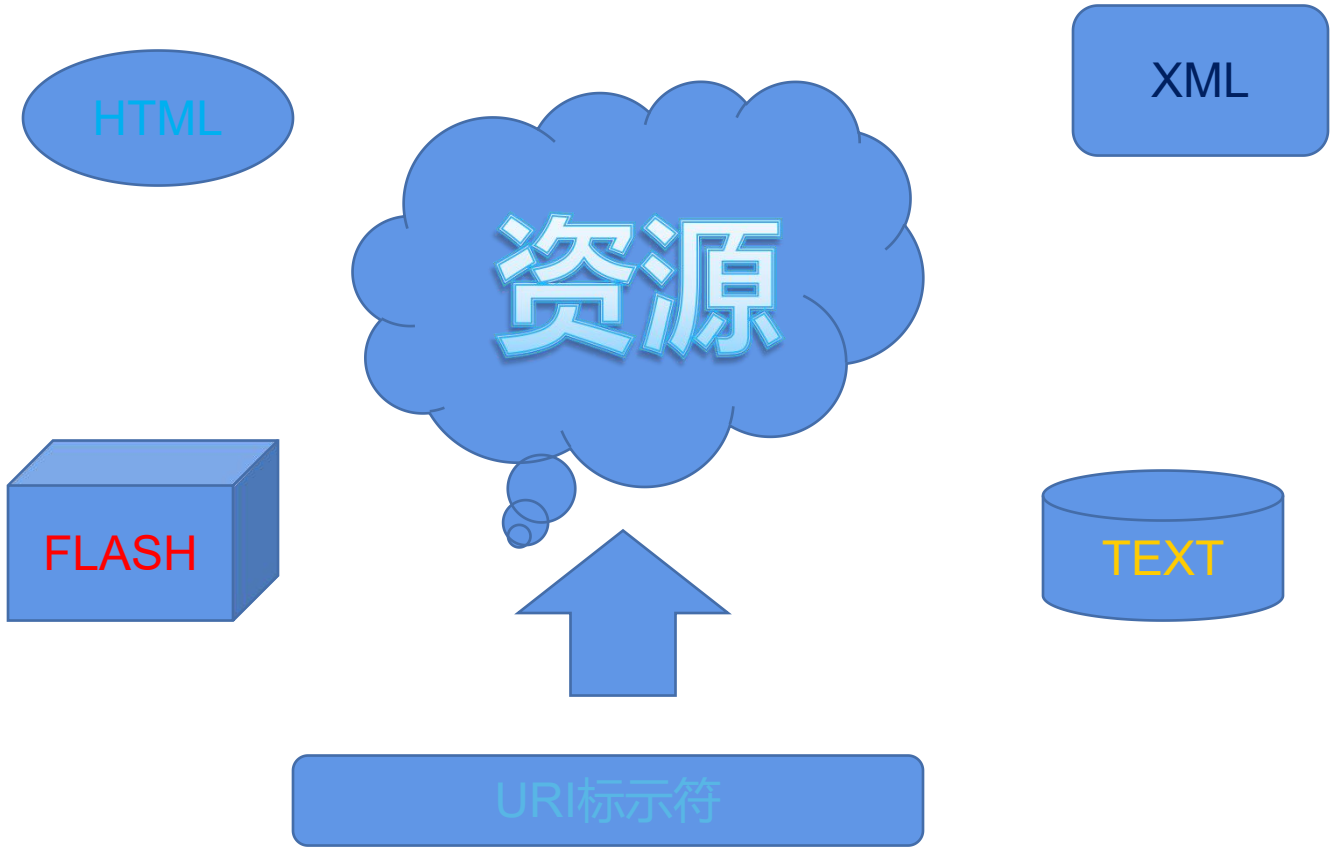
  “并非所有的资源都是通过网络能够获取的”。

- 任何事物，只要有被引用的必要，就是一个资源(resource)。它可以是一个实物，也可以是一个抽象的概念。

- 通常一个资源是某个可以存放在计算机上并体现为比特流的事物。在Web中，可以这样认为——资源是URI标示的东西。

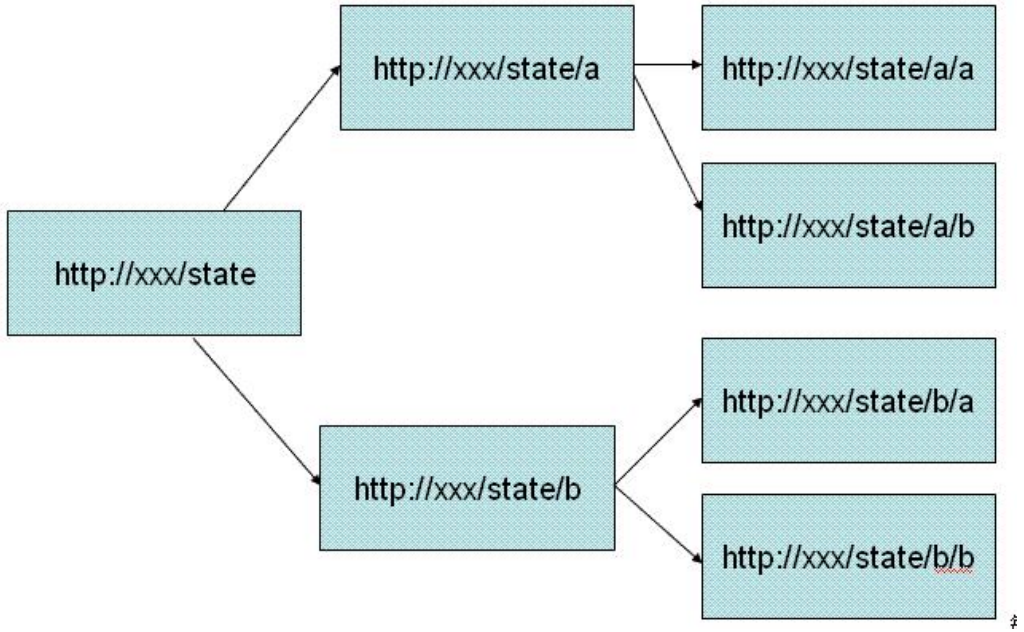
■ 表示

- 资源和表示不是一码事。Web上获取的不是资源，而是资源的表示。
- 对于给定的资源，可以有很多不同的表示。



■ 状态

- 首先要理解资源状态和应用状态
- 在客户-服务端模式下，让客户端维护应用状态，并确保客户端向服务器发出的请求都包含理解请求所需的全部信息，而服务器不应该维护该状态。
- REST式解决方案是使用URI。每个概念上独立的资源都可使用单个URI，不希望通过Cookie或隐藏在有效负载的参数来提供额外信息。



URI设计规则

规则1. 资源表示一种实体，URI使用名词表示，不应该包含动词；一般来说，数据库中的表都是同种记录的"集合"（collection），所以URI中的名词应该使用复数。

正确举例：

GET /zoos：列出所有动物园

POST /zoos：新建一个动物园

GET /zoos/ID：获取某个指定动物园的信息

错误举例：

GET /posts/show/1

正确写法：

GET /posts/1

规则2.如果某些动作是HTTP动词表示不了的，应该把动作做成一种资源。

错误举例：网上汇款，从账户1向账户2汇款500元：

POST /accounts/1/transfer/500/to/2

正确的写法：把动词transfer改成名词transaction，资源不能是动词，但是可以是一种服务：

POST /transaction

from=1&to=2&amount=500.00

比如登录、退出：

POST /sessions　　DELETE /sessions/{id}

规则3.参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 /zoos/ID/animals 与 GET /animals?zoo_id=ID 的含义是相同的。

注意：

对于/zoos/ID/animals/ID/方式，注意关联层次不要太深，可以将关系通过参数方式表现：/animals?zoo_id=ID

规则4.一些常见的参数：

?limit=10：指定返回记录的数量

?offset=10：指定返回记录的开始位置

?page=2&per_page=10：指定第几页，以及每页的记录数

?sortby=name&order=asc：指定返回结果按照哪个属性排序，以及排序顺序

HTTP动词

7个HTTP动词：GET/POST/PUT/DELETE/PATCH/HEAD/OPTIONS

表示对资源的一组操作：

GET（SELECT）：从服务器取出资源（一项或多项）

POST（CREATE）：在服务器新建一个资源

PUT（UPDATE）：在服务器更新资源（客户端提供改变后的完整资源）

PATCH（UPDATE）：在服务器更新资源（客户端提供改变的属性）

DELETE（DELETE）：从服务器删除资源

HEAD：用于获取某个资源的元数据（metadata）

OPTIONS：用于获取某个资源所支持的Request类型

ContentType

用于指定请求和响应的HTTP内容类型。下面是几个常见的Content-Type:

1.text/html

2.text/plain

3.text/css

4.text/javascript

5.application/x-www-form-urlencoded

6.multipart/form-data

7.application/json

8.application/xml

前面几个是html，css，javascript的文件类型，后面四个是POST的发包方式。

如果未指定 ContentType，GET默认为text/html。

■ 状态码

常见状态码：

2xx范围的状态码是保留给成功消息使用的。

3xx范围的状态码是保留给重定向用的。

4xx范围的状态码是保留给客户端错误用的。例如，客户端提供了一些错误的数据或请求了不存在的内容。这些请求应该是幂等的，不会改变任何服务器的状态。

5xx范围的状态码是保留给服务器端错误用的。这些错误常常是从底层的函数抛出来的，并且开发人员也通常没法处理。发送这类状态码的目的是确保客户端能得到一些响应。

■ 版本化

一是在URL中包含版本信息：

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

二是在请求头里面保持版本信息。

Accept: vnd.example-com.foo+json; version=1.0

相对来说，在请求头里面包含版本信息远没有放在URL里面来的容易。

返回结果

当使用不同的HTTP动词向服务器请求时，客户端需要在返回结果里面拿到一系列的信息。比如，当一个客户端创建一个资源时，客户端常常不知道新建资源的ID（也许还有其他的属性，如创建和修改的时间戳等）。这些属性需要在随后的请求中返回，并且作为刚才POST请求的一个响应结果。

下面是非常经典的RESTful API定义：

GET /resources: 返回一系列资源对象

GET /resources/ID: 返回单独的资源对象

POST /resources: 返回新创建的资源对象

PUT /resources: 返回完整的资源对象

PATCH /resources: 返回完整的资源对象

DELETE /resources: 返回一个空文档

认证

JWT（JSON Web Token）提供了一个轻量级的解决方案。JWT 的原理是，服务器认证以后，生成一个 JSON 对象，发回给用户。

```
{
  "姓名": "张三",
  "角色": "管理员",
  "到期时间": "2018年7月1日0点0分"
}
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Header

Header 部分是一个 JSON 对象，描述 JWT 的元数据。

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Payload

Payload 部分也是一个 JSON 对象，用来存放实际需要传递的数据。JWT 规定了7个官方字段，供选用。

iss (issuer)：签发人

exp (expiration time)：过期时间

sub (subject)：主题

aud (audience)：受众

nbf (Not Before)：生效时间

iat (Issued At)：签发时间

jti (JWT ID)：编号

■ Signature

Signature 部分是对前两部分的签名，防止数据篡改。

■ JWT 的使用方式

客户端收到服务器返回的 JWT，可以储存在 Cookie 里面，也可以储存在 localStorage。

此后，客户端每次与服务器通信，都要带上这个 JWT。你可以把它放在 Cookie 里面自动发送，但是这样不能跨域，所以更好的做法是放在 HTTP 请求的头信息Authorization字段里面。

■ JWT 的几个特点

（1）JWT 默认是不加密，但也是可以加密的。生成原始 Token 以后，可以用密钥再加密一次。

（2）JWT 不加密的情况下，不能将秘密数据写入 JWT。

（3）JWT 不仅可以用于认证，也可以用于交换信息。有效使用 JWT，可以降低服务器查询数据库的次数。

（4）JWT 的最大缺点是，由于服务器不保存 session 状态，因此无法在使用过程中废止某个 token，或者更改 token 的权限。也就是说，一旦 JWT 签发了，在到期之前就会始终有效，除非服务器部署额外的逻辑。

（5）JWT 本身包含了认证信息，一旦泄露，任何人都可以获得该令牌的所有权限。为了减少盗用，JWT 的有效期应该设置得比较短。对于一些比较重要的权限，使用时应该再次对用户进行认证。

（6）为了减少盗用，JWT 不应该使用 HTTP 协议明码传输，要使用 HTTPS 协议传输。

文档

必须要为API准备文档，否则没有人知道怎么使用它。

不要截断文档示例中请求与响应的内容，要展示完整的东西。

文档化每一个端点所预期的响应代码和可能的错误消息，和在什么情况下会产生这些的错误消息。

02

Hello World

SpringMVC的RESTful实现

常用注解:

@RequestMapping

@PathVariable

@RequestParam

@RequestBody

@ResponseBody

@RequestMapping

是一个用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径。

该注解有六个属性，分成三类：

1、 value， method

value： 指定请求的实际地址

method： 指定请求的method类型， GET、POST、PUT、DELETE等

2、consumes，produces

consumes： 指定处理请求的提交内容类型（Content-Type），例如application/json, text/html

produces: 指定返回的内容类型，仅当request请求头中的(Accept)类型中包含该指定类型才返回

3、params，headers

params： 指定request中必须包含某些参数值时，才让该方法处理请求

headers：指定request中必须包含某些指定的header值，才能让该方法处理请求

示例代码:

```
@Controller
public class TestController {
    @RequestMapping(value="/users/{userId}", method = RequestMethod.GET)
    public String getUser(@PathVariable("userId") String userId){
        System.out.println("User Id : " + userId);
    }
}
```

@PathVariable

用于将请求URL中的变量映射到功能处理方法的参数上。

示例代码:

```
@Controller
public class TestController {
    @RequestMapping(value="/users/{userId}", method = RequestMethod.GET)
    public String getUser(@PathVariable("userId") String userId){
        System.out.println("User Id : " + userId);
    }
}
```

@RequestParam

1.常用来处理简单类型的绑定，通过request.getParameter() 获取的String可直接转换为简单类型的情况。可以处理get 方式中queryString的值，也可以处理post方式中 body data的值；

2.用来处理Content-Type：为 application/x-www-form-urlencoded编码的内容，提交方式GET、POST；

3.有三个常用参数：
value：值表示接受的传入的参数类型
defaultValue：表示设置默认值
required ： 是否是必须要传入的参数

示例代码:

```
@Controller
@RequestMapping("/pets")
public class PetsController {
    @RequestMapping(value="",method = RequestMethod.GET)
    public String setupForm(@RequestParam("petId") int petId) {
        // ...
        return "petForm";
    }
}
```

@RequestBody

该注解常用来处理Content-Type不是application/x-www-form-urlencoded编码的内容，如application/json等。它将处理完的结果绑定到相应的实体或Map上。

示例代码：

```
@RequestMapping(value = "/users", method = RequestMethod.POST)
public void handle(@RequestBody User bean) throws IOException {
    //…
}
```

@ResponseBody

使用场景：

返回的数据不是html页面，而是其他某种格式的数据时（如json、xml等）

示例代码：

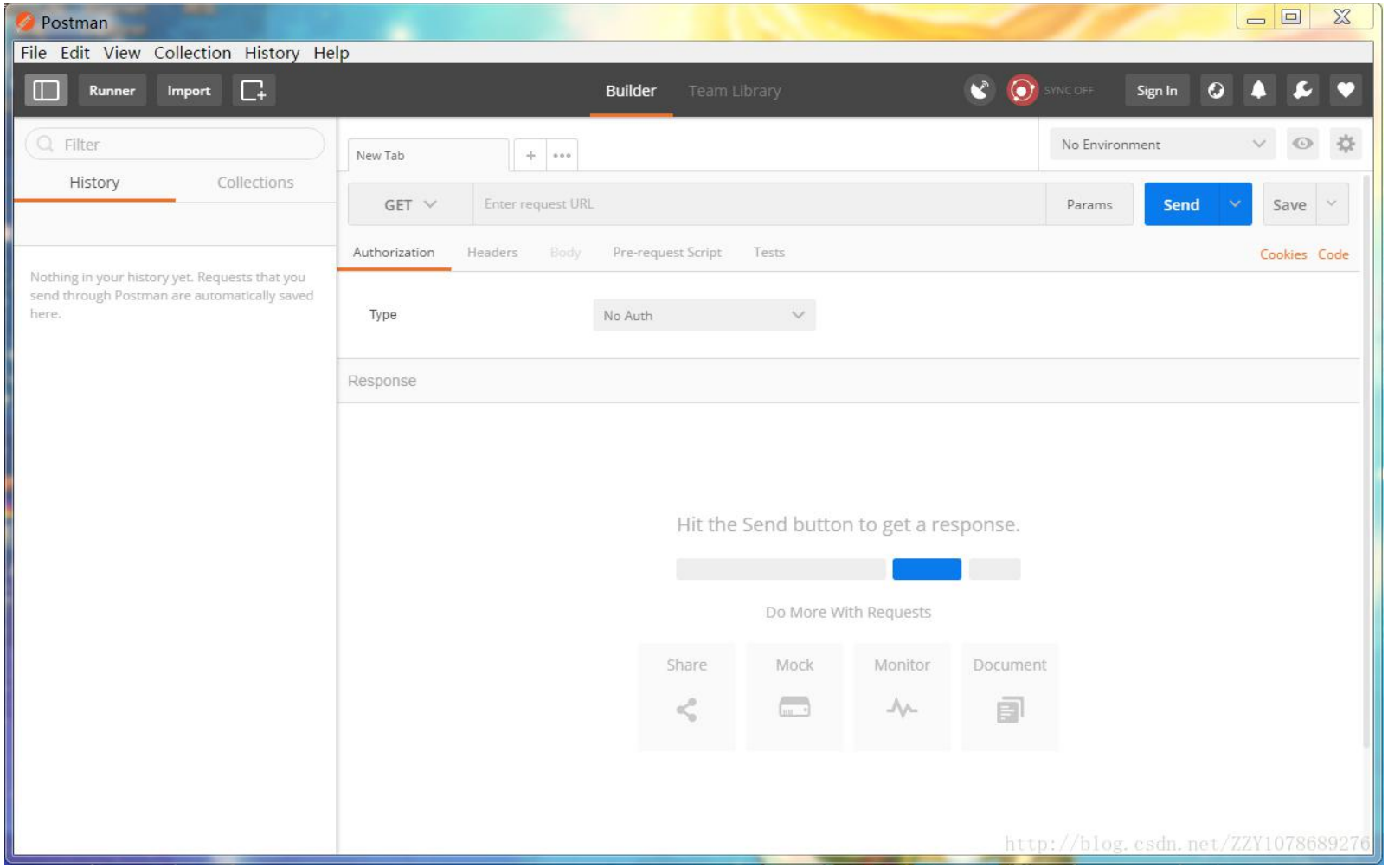
```
@RequestMapping("/users/{userId}")
@ResponseBody
public Person getUserInfo(@PathVariable(“userId”)String userId) {
    Person p = userService.getUserById(userId);
    return p;
}
```

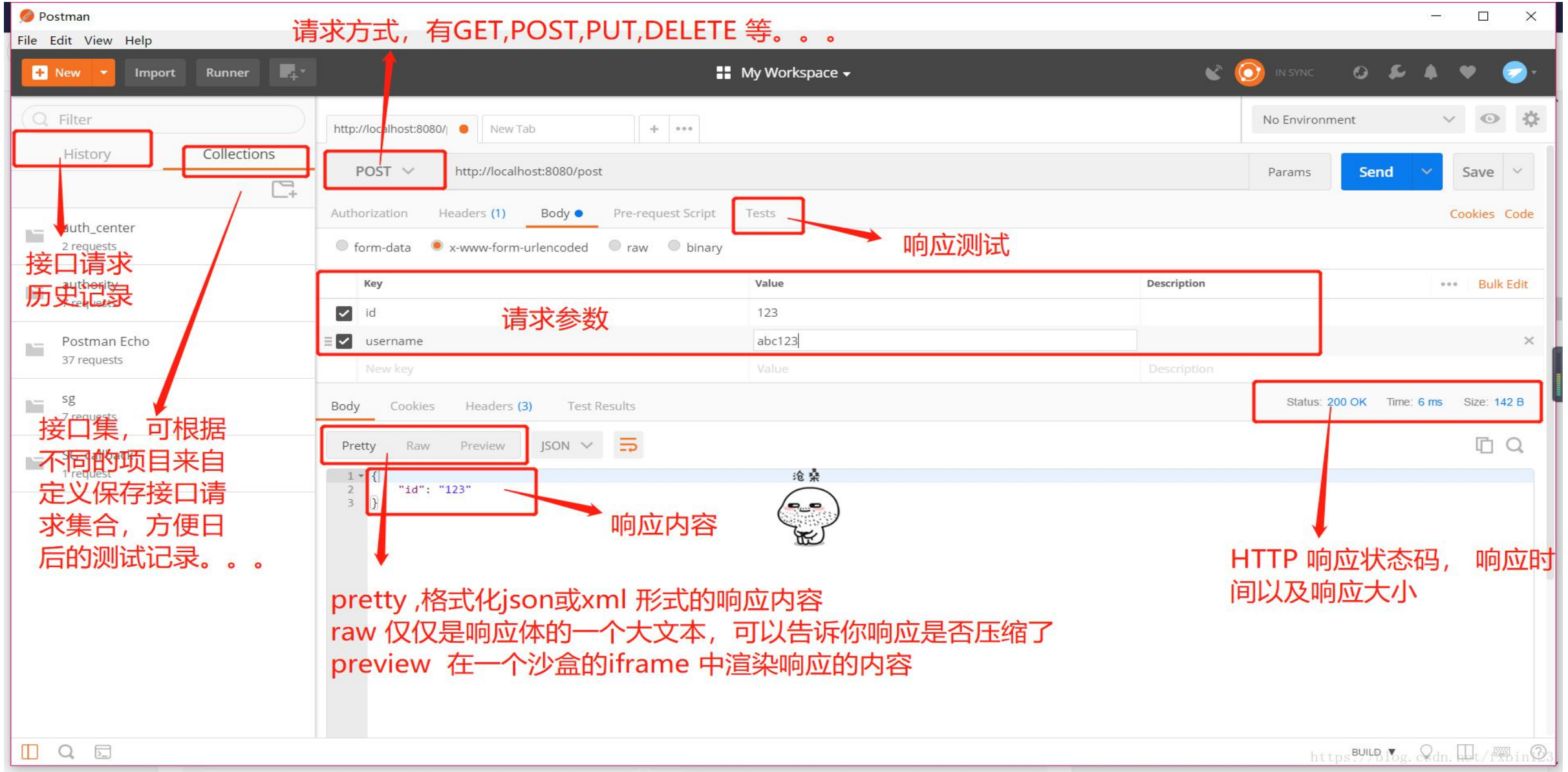
03

Postman工具测试

Postman是一个强大的HTTP调试和请求工具。

其他类似流行的工具有：火狐浏览器插件-RESTClient等。




Postman
File Edit View Help
New Import Runner
My Workspace
IN SYNC
Filter
History
Collections
auth_center
2 requests
Postman Echo
37 requests
sg
7 requests
No Environment
http://localhost:8080/
New Tab
POST
http://localhost:8080/post
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Cookies
Code
form-data
x-www-form-urlencoded
raw
binary
Key
Value
Description
Bulk Edit
id
123
username
abc123
New key
Value
Description
Body
Cookies
Headers (3)
Test Results
Status: 200 OK
Time: 6 ms
Size: 142 B
Pretty
Raw
Preview
JSON
{
"id": "123"
}
BUILD
请求方式，有GET,POST,PUT,DELETE 等。。。
接口请求历史记录
接口集，可根据不同的项目来自定义保存接口请求集合，方便日后的测试记录。。。
请求参数
响应测试
响应内容
HTTP 响应状态码， 响应时间以及响应大小
pretty ,格式化json或xml 形式的响应内容
raw 仅仅是响应体的一个大文本，可以告诉你响应是否压缩了
preview 在一个沙盒的iframe 中渲染响应的内容

Windows 64位下载地址：
https://app.getpostman.com/app/download/win64?_ga=2.206991635.1254191070.1514361597-261809244.1514361597

Windows 32位下载地址：
https://app.getpostman.com/app/download/win32?_ga=2.206991635.1254191070.1514361597-261809244.1514361597

04

注意事项

“REST难道就没有任何缺点了吗？”当然不是，评价一种软件架构的优劣，不能脱离开软件的具体运行环境。永远不存在适用于任何运行环境的、包治百病的银弹式架构。

REST是一种为运行在互联网环境中的Web应用量身定制的架构风格。REST在互联网这个运行环境之中已经占据了统治地位，然而，在企业内网运行环境之中，REST还会面临DO、RPC的巨大挑战。特别是一些对实时性要求很高的应用，REST的表现不如DO和RPC。所以需要针对具体的运行环境来具体问题具体分析。

1.避免不经意间泄露的业务信息

会说话的ID

返回多余的数据

2.避免遗漏对资源从属关系的检查

一个典型的RESTful的URL会用资源名加上资源的ID编号来标识某个唯一的资源

例如：

/users/100

带有从属关系的RESTful：

/ResourceA/<ResourceA Id>/ResourceB/<ResourceBId>/ResourceC/<ResourceCId>

例如：

/users/100/orders/280010

在对资源进行操作之前，就得先检查这些资源之间的从属关系，以确保当前请求具有相关的访问、操作权限。

3. API速率限制的保护

发送短信的API

用户登录的API

某些大量消耗服务器资源的API

解决办法:

图片验证码

网络防火墙

总结

1. RESTful API简介
2. Hello World
3. 工具调试
4. 注意事项

谢谢聆听

Email： fbx2020@163.com
WebSite： http://www.coohelp.vip/