RestAssured接口测试

Rest和Restful基础知识

- REST-Representation State Transfer

一种软件架构风格，可以降低开发的复杂性，提高系统的可伸缩性。

- REST or RESTFUL 区别：

RESTFUL 是REST的形容词形式

RESTFUL API指的是REST风格的接口

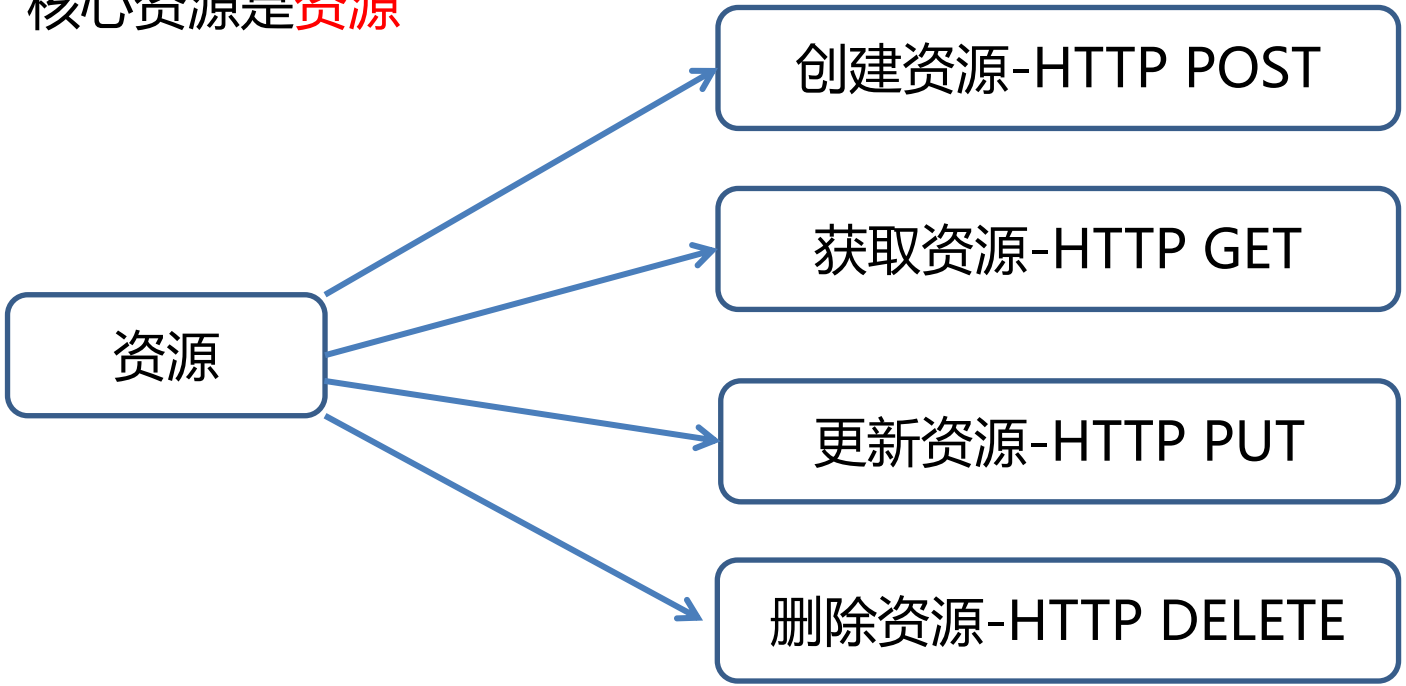
一般来说REST等于RESTFUL ，区别是一个是名词一个是形容词

REST API

- 出现：REST 是由Roy Fielding博士在2000年他的博士论文中提出来的
- 定义：简单来说REST 是一种系统架构设计风格（而非标准），一种分布式系统的应用层解决方案
- 目的：Client和Server进一步解耦
- 应用：最为经典的莫过于github API

REST API

核心资源是资源



REST API

- REST 支持的方法：

Verd	描述
HEAD（Select）	获取某个资源的头部信息
GET（Select）	获取资源
POST（Create）	创建资源
PATCH（Update）	更新资源的部分属性（很少用，一般用POST替代）
PUT（Update）	更新资源，客户端需要提高新建资源的所有属性
DELETE（Delete）	删除资源

REST API接口规范

- 设计规范

协议：建议使用HTTPS协议，确保交互数据的传输安全。

域名：应该尽量将API部署在专用域名之下。

https://api.example.com

版本控制：将版本号放在URL或者Header中

REST API接口规范

返回结果设计：

通用错误码。具体产品由具体产品api文档给出。

```
{
    "message":"success",
    "code":200
 }
```

REST API实例

- GET /product：列出商品
- POST /product：新建一个商品
- GET /product/ID：获取某个指定商品的信息
- PUT /product/ID：更新某个指定商品的信息
- DELETE /product/ID：删除某个商品
- GET /product/ID/purchase：获取某个指定商品的所有

Rest-Assured框架

- Rest-Assured 是一套由 Java 实现的 REST API 测试框架，它是一个轻量级的 REST API 客户端，可以直接编写代码向服务器端发起 HTTP 请求，并验证返回结果。

Rest-Assured请求处理_Get

import static io.restassured.RestAssured.*;

```
// 参数直接跟在URL后面发起请求
@Test
public void getHttpTest1() {
    Response response = given().get("https://api.douban.com"
            + "/v2/book/search");
    response.print();
}
```

Rest-Assured请求处理_Get

```
//  把参数剥离出来，使用.params("key","value","key","value"....),
//  因为是https，需要加上ssl的配置，让所有请求支持所有的主机名:
    @Test
    public void getHttpsTest2() {

        // 配置SSL 让所有请求支持所有的主机名
        Response response = given().config((RestAssured.config().sslConfig(new SSLConfig().relaxedHTTPSValidation())))
                .params("q", "测试", "start", 0, "count", 2).get("https://api.douban.com/v2/book/search");

        response.print();
    }

    @Test
    public void getHttpsTest3() {

        // 配置SSL 让所有请求支持所有的主机名
        Response response = given().config((RestAssured.config().sslConfig(new SSLConfig().relaxedHTTPSValidation())))
                .params("q", "测试")
                .params("start", 0)
                .params("count", 2)
                .get("https://api.douban.com/v2/book/search");
        // 打印出 response 的body
        response.print();
    }
```

Rest-Assured请求处理_Post

```
@Test
public void posetTestMiniblog1(){
    // 设置request Content-Type
    Response response = given().contentType("application/x-www-form-urlencoded")
            .body("username=lihuanzhen&password=123456")
            // POST 请求
            .post("http://study-miniblog-new.qa.netease.com/ajax/user/login");
    response.print();
}
```

```
@Test
public void posetTestMiniblog2(){
    // 设置request Content-Type
    Response response = given().contentType("application/x-www-form-urlencoded")
            .param("username", "lihuanzhen")
            .param("password", "123456")
             // POST 请求
             .post("http://study-miniblog-new.qa.netease.com/ajax/user/login");
    response.print();
}
```

Rest-Assured请求处理_Post

```
@Test
public void posetTestQgLogin() {
    // 设置request Content-Type
    Response response = given().contentType("application/json")
            .body("{\"phoneArea\":\"86\",\"phoneNumber\":\"20000000000\",\"password\":\"
            // POST 请求
            .post("http://study-perf.qa.netease.com/common/fgadmin/login");
    response.print();
}
```

注意：如果请求Content-Type类型是：application/x-www-form-urlencoded，可以直接用param()或者params()管理，但是如果是application/json 则只能用body()管理参数

设置header，cookie

```
@Test
public void setCookie(){
      Response response = given()
              .cookie("cookie","value")
              .cookies("cookiename1", "value1", "cookiename2", "value2")
              .header("Accept-Encoding:", "gzip, deflate")
              .headers("header1","value1","header2","value2") .get("XXXXXXX");

}
```

获取具体的某个cookies：

response.getCookie("cookiesName");

获取所有的cookies，返回一个map:

Map cookies = response1.getCookies();

获取Response 状态码

- 获取Response 状态码，返回int类型：
  response.getStatusCode();
- 获取Response body

  response.getBody().asString()
- 获取指定header

Headers headers = response.getHeaders();
headers.hasHeaderWithName("XXX")

解析JSON

```java
@Test
public void skulitTest() {
    Response response = given().get("http://study-perf.qa.netease.com/common/skuList?goodsId=1");
    int code = response.jsonPath().getInt("code");
    System.out.println("code: " + code);
    // 获取所有的 subtitle
    ArrayList<String> subtitles = response.jsonPath().get("result.skuName");
    for (int i = 0; i < subtitles.size(); i++) {
        System.out.println(subtitles.get(i));
    }

}
```

详见：JSONDemo.java

Rest-Assured响应校验

```
    @Test
    public void testDemo1() {
        Response response = given()
                .expect()
                .body("result.skuName", hasItems("夜空黑","青果绿"))
                .body("code", is(200))
                .body("message", is("success"))
//              // 判断 code 是否等于 5
                .body("code", equalTo(200))
                .when().get("http://study-perf.qa.netease.com/common/skuList?goodsId=1");
    }
```

尽管Rest Assured 提供了校验方法，但是在实际API自动化测试过程中，因为往往需要校验的字段非常多，建议还是直接先把要校验的JSON字段解析出来，再通过TestNG提供的Assert类进行校验。