# 前后端分离之接口设计

## 1：什么是接口设计？

实际生活中**最常见的接口例子**：插座！

接口设计就是设计插座的过程，不一样的接口具有不同的功能

。

## 2：为什么要对接口进行设计？

没有一个后端想一次一次地向前端解释他在做什么

没有一个前端想向后端一遍一遍的问他在做什么

我们只需要事先定义好插座的**接口标准**，各大插座厂商只要按这个接口标准生产，管你什么牌子、内部什么电路结构，这些均和用户无关，用户拿来就可以用；而且即使插座坏了，只要换一个符合接口标准的新插座，一切照样工作！

我们在设计一个软件的代码架构时，我们都希望**事先约定**好各个功能的**接口**（即：约定好接口签名和方法），实际开发时我们只需要实现这个接口就能完成具体的功能！后续即使项目变化、功能升级，程序员只需要按照接口约定重新实现一下，就可以达到系统升级和扩展的目的！

对应于一个良好的接口应该是无论底层代码如何变化扩展，都不会影响到外部对其正常的调用。

## 3：如何设计一个良好的接口？

在设计接口时，有很多因素要考虑，如接口的业务定位，接口的安全性，接口的可扩展性、接口的稳定性、接口的跨域性、接口的协议规则、接口的路径规则、接口单一原则、

接口过滤和接口组合等诸多因素，本篇文章将简要分析这些因素。

#### 3.1规范性建议：

#### 3.1.1职责原则

在设计接口时，必须明确接口的职责，即接口类型，接口应解决什么业务问题等

#### 3.1.2单一性原则

在明确接口职责的条件下，尽量做到接口单一，即一个接口只做一件事，而非两件以上。

#### 3.1.3协议规范

在设计接口时，应明确接口协议，是采用HTTP协议,HTTPS协议还是FTP协议，要根据具体情况来定。

(1)FTP协议(File Transfer Protocol，简称FTP)，是一套标准的文件传输协议，用于传输文件，如.txt，.csv等，一般文件传输，采用FTP协议

(2)HTTP协议，适用一般对安全性要求比较低或没要求的业务情景

(3)HTTPS=HTTP+SSL,适用于对安全性要求较高的业务情景

#### 3.1.4路径规则

在设计接口时，应明确接口协议，是采用HTTP协议,HTTPS协议还是FTP协议，要根据具体情况来定。

(1)FTP协议(File Transfer Protocol，简称FTP)，是一套标准的文件传输协议，用于传输文件，如.txt，.csv等，一般文件传输，采用FTP协议

(2)HTTP协议，适用一般对安全性要求比较低或没要求的业务情景

(3)HTTPS=HTTP+SSL,适用于对安全性要求较高的业务情景

#### 3.1.5http请求方式

接口基本访问协议：get(获取)，post(新增)，put(修改)和delete(删除)

get     /users：列出所有用户

get    /users/id：根据id获取用户

post   /user：新增用户

put      /user/id：根据用户id更新用户

delete   /user/id：根据用户id删除用户

#### 3.1.6域名

一般地，域名分为主域名和专有域名，主域名适合api长期不变或变化较少的业务，专有域名是解决具体的专有业务的

以百度举例：

(1)主域名:www.baidu.com

(2)产品服务类

 百度文库：https://wenku.baidu.com/

 百度知道：https://zhidao.baidu.com/

 百度资讯： https://zhidao.baidu.com/

(3)市场活动类

百度公益：http://gongyi.baidu.com

百度logo：http://logo.baidu.com/

百度世界：https://baiduworld.baidu.com

#### 3.1.7跨域考虑

在明确域名的情况下，一定要考虑接口是否跨域，以及跨域应采用的技术手段等

#### 3.1.8api版本

对于接口的url，应加版本号http://api.demo.com/v{d}/，如 ，其中d表示版本号,如v1.0,v2.0

例子：获取产品号为2019,版本号为v1.0的版本号的产品信息

/api/v1.0/Pruducts/2019

#### 3.1.9适度过滤信息

当记录数比较多时(如 SELECT \* FROM TBName)，因适当添加一些条件对数据进行过滤，如TOP,分页,分组，排序和WHERE条件等

下面是一些常见的参数。

1. ?limit=100：返回100条数据
2. ?offset=101：从第101条数据开始返回
3. ?page=10：指第10页
4. per\_page=100：每页100条数据
5. ?sortby=name：排序字段
6. ?order=desc：降序
7. ？group=groupName:分组
8. ?producy\_type=1：筛选条件

#### 3.1.10返回数据格式

返回数据格式，一般包括三个字段：

1. 失败情况(状态码、错误码和错误描述)
2. {
4. “status”:0,//状态码 0-表示失败，1-表示成功
6. “error\_code”:”2003”,//错误码，一般在设计时定义
8. “error\_des”:”身份验证失败”//错误描述，一般在设计时定义
10. }
11. 成功情况(标识id,数据对象,状态码)
12. {
14. ”sid“:”sh20190111”,//token id
16. ”users“:{
18. ”id“:”al201901111341”,//用户id
20. “name”:”Alan\_beijing”,//用户名
22. “addr”:”用户地址”
24. },
26. “status”:1//状态码 0-表示失败，1-表示成功
28. }

#### 3.1.11安全性原则

接口暴露的考虑，接口并发量的考虑，接口防攻击的考虑，接口跨域的考虑等

#### 3.1.12可扩展性原则

在设计接口时，充分考虑接口的可扩展性

#### 3.1.13定义api界限

任何api，从权限上，可归结为匿名api和非匿名api，前者不需要验证，后者需要验证

#### 3.1.14定义api返回码

在api设计时，要定好api返回码，如

1 --授权过期

404--未找到资源

500--内部服务器错误

600--账号被锁

#### 3.2反规范性建议：

存在这样一种业务场景：某个接口需要返回多个api接口组合的结果 ，在类似的业务场景下，所设计的接口，具有一定的反规范性。

1. **Request**
2. data:[
3. {url:'api1',type:'get',data:{...}},
4. {url:'api2',type:'get',data:{...}},
5. ]
6. **Responce**
7. {
8. status:0,
9. msg:'',
10. data:[
11. {status:1,msg:'',data:[]},
12. {status:1,msg:'',data:{}}
13. ]
14. }

## 4：接口设计实例

假设存在这样一个一个业务：一个ERP系统，需要提供两个接口，一个是用户访问接口(需要验证)，另一个是用户注册接口(不需要验证)。

根据本篇文章3部分的建议，我们来设计满足该业务需求的接口

#### 4.1定义统一参数

#### 4.1.1定义输入参数

注：注册接口不需要

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 描述 |
| Uid | Int | Y | 用户id |
| Sid | String | Y | 令牌token |

#### 4.1.2定义输出参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 描述 |
| Status | Int | Y | 状态码 |
| Error\_code | Int | Y | 错误码 |
| Error\_desc | String | N | 错误信息 |

#### 4.1.3定义统一错误码

|  |  |
| --- | --- |
| Error\_code | Error\_desc |
| 0 | 失败 |
| 1 | 成功 |
| 500 | 内部服务器错误 |
| 600 | 账号被锁 |
| …… | …… |

#### 4.2定义接口授权类别

|  |  |  |
| --- | --- | --- |
| 参数名 | 是否需要登录 | 描述 |
| User/register | N | 用户注册 |
| User/signin | Y | 用户登录 |
| User/signout | Y | 用户退出 |
| User/info | Y | 获取用户信息 |
| User/changprofile | Y | 修改用户资料 |

#### 4.3用户接口

#### 4.3.1用户注册

|  |  |
| --- | --- |
| 类别 | 含义 |
| 接口 | User/register |
| 描述 | 用户注册 |
| 验证方式 | Session |
| http方式 | post |
|  |  |

#### 4.3.2Request

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 描述 |
| mobile | int | Y | 注册手机号 |
| Verify\_code | String | Y | 手机验证码 |
| password | String | Y | 用户密码 |

#### 4.3.3Responce

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 描述 |
| Sid | string | Y | SessionId(token) |
| uid | Int | Y | 用户id |
|  |  |  |  |

#### 4.3.4code示例

1. Request:
2. {
3. "mobile":13636595499,
4. "verify\_code":"987654",
5. "pwd":"123456"
6. }
8. Responce:
9. (1)error
10. {
11. "status":0,
12. "error\_code":1001，
13. "error\_desc":"手机验证码已失效"
14. }
15. (2)succed
16. {
17. "sid":"sh201901141529",
18. "uid":1,
19. "status":1
20. }

#### 4.4用户登录

#### 4.4.1登录接口概述

|  |  |
| --- | --- |
| 类别 | 含义 |
| 接口 | User/signin |
| 描述 | 用户登录 |
| 验证方式 | Session |
| http方式 | Post |
|  |  |

#### 4.4.2Request

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 描述 |
| Mobile | Int | N | 注册手机号 |
| Username | string | N | 用户名 |
| password | String | N | 用户密码 |
| type | int | Y | 用户类型  （1.QQ2.微信3.支付宝4.手机号） |

#### 4.4.3Responce

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必须 | 描述 |
| Sid | string | Y | seesionId |
| User | String | Y | 用户信息 |
| Uid | Int | Y | 用户id |
| Name | String | Y | 用户名 |
| Age | Int | Y | 年龄 |
| gender | Int | Y | 性别 |

#### 4.4.4code

1. Responce:
2. 1.error
3. {
4. "status":0,
5. "error\_code":1002,
6. "error\_desc":"密码错误"
7. }
8. 2.succeed
9. {
10. "sid":"sh201901141529",
11. "user":{
12. "id":1,
13. "username":"",
14. age:0,
15. gender:0
16. },
17. "status":1
18. }