**接口开发规范**

# 接口架构设计

## 使用HTTPS 保证数据安全

HTTPS在HTTP的基础上添加了SSL安全协议，自动对数据进行了加密，在一定程度可以防止监听、劫持、重发等，安全性可以提高很多。

不清楚Web访问时是否会出现跨域的问题

## URL格式定义

由于URL对大小写敏感，推荐使用Snake命名法：全部小写，单词间用 - 连接

例如： https://api.example.com/api/this-is-my-first-post

## 使用JSON作为数据传输格式

对JSON字符串执行格式化

## 使用POST方式

为方便开发和统一管理，所有的接口都采用POST方法提供服务

## Token

使用JWT Token

JWT设计的好处是不用在服务器保持授权信息，会话不需要服务器维护

缺陷： 1，Token一旦签发，在有效期内不受控制，即服务端修改Token有效期时，之前签发的Token在过期前还是有效的

2，Token在过期后需要重新签发

## 签名的设计

           原理：用户登录后向服务器提供用户认证信息（如账户和密码），服务器认证完后给客户端返回一个Token令牌，用户再次获取信息时，带上此令牌，如果令牌正取，则返回数据。对于获取Token信息后，访问用户相关接口，客户端请求的url需要带上如下参数：

         时间戳：timestamp

         Token令牌：token

         然后将所有用户请求的参数按照字母排序（包括timestamp，token），然后更具MD5加密（可以加点盐），全部大写，生成sign签名，这就是所说的url签名算法。然后登陆后每次调用用户信息时，带上sign，timestamp，token参数。

例如：原请求https://www.andy.cn/api/user/update/info.shtml?city=北京 （post和get都一样，对所有参数排序加密）

 加上时间戳和token

       https://www.andy.cn/api/user/update/info.shtml?city=北京&timestamp=12445323134&token=wefkfjdskfjewfjkjfdfnc

      然后更具url参数生成sign

      最终的请求如

         https://www.andy.cn/api/user/update/info.shtml?city=北京&timestamp=12445323134&token=wefkfjdskfjewfjkjfdfnc&sign=FDK2434JKJFD334FDF2

其最终的原理是减小明文的暴露次数；保证数据安全的访问。

具体实现如下：

           1. api请求客户端想服务器端一次发送用用户认证信息（用户名和密码），服务器端请求到改请求后，验证用户信息是否正确。

        如果正确：则返回一个唯一不重复的字符串（一般为UUID），然后在Redis（任意缓存服务器）中维护Token----Uid的用户信息关系，以便其他api对token的校验。

        如果错误：则返回错误码。

            2.服务器设计一个url请求拦截规则

               （1）判断是否包含timestamp，token，sign参数，如果不含有返回错误码。

               （2）判断服务器接到请求的时间和参数中的时间戳是否相差很长一段时间（时间自定义如半个小时），如果超过则说明该                         url已经过期（如果url被盗，他改变了时间戳，但是会导致sign签名不相等）。

               （3）判断token是否有效，根据请求过来的token，查询redis缓存中的uid，如果获取不到这说明该token已过期。

               （4）根据用户请求的url参数，服务器端按照同样的规则生成sign签名，对比签名看是否相等，相等则放行。（自然url签名                       也无法100%保证其安全，也可以通过公钥AES对数据和url加密，但这样如果无法确保公钥丢失，所以签名只是很大程                       度上保证安全）。

                （5）此url拦截只需对获取身份认证的url放行（如登陆url），剩余所有的url都需拦截。

            3.Token和Uid关系维护

               对于用户登录我们需要创建token--uid的关系，用户退出时需要需删除token--uid的关系。

## 使用gzip进行数据压缩

# 接口格式设计

## 通用的请求报文格式

Body：

{

"auth": {

"appKey": "android\_erp",

"appVersion": 1,

"appID": "value",

"userID": "value",

"companyID": "value"

},

"data": {

"obj1": {

"name": "Google",

"url": "http://www.google.com"

},

"obj2": {

"name": "Baidu",

"url": "http://www.baidu.com"

},

"obj3": {

"name": "SoSo",

"url": "http://www.SoSo.com"

}

}

}

Header:

Token : 服务器返回的，不用授权的接口传空字符串

Sign ：基于Body的签名

## 通用的返回报文格式

{

"code" : 1234,

"message" : "Something bad happened <img src="http://blog.jobbole.com/wp-includes/images/smilies/icon\_sad.gif" alt=":-(" class="wp-smiley"> ",

"description" : "More details about the error here"

}

## 详细接口格式

接口类型大概分为如下几类：

### 提交一个对象，获取一个对象

### 提交一个对象，获取一个列表

排序

过滤

state=open&sort=-priority

### 提交一个对象，获取多个对象和多个列表

### 提交一个列表

### 提交多个列表

# 特殊数据格式设计

Number：整数

String：字符串

Boolean：true 或 false

Array：数组包含着方括号[]中

Object：对象包含在大括号{}中

Null：空类型

时间： 统一使用 “*yyyy-MM-dd HH:mm:ss*”

数值： 客户端解析失败时应该取0