# Yii2 api设计 简单可测接口demo

（请看一下底部文字处，有说明）

## 1.1 注册接口

接口地址 ： post /register

**传入参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **元素名称** | **元素名称** | **是否必传** | **类型** | **取值说明** |
| pf | 系统标识 | 是 | int | 1、安卓 ： 2、ios |
| ver | App版本号 | 是 | string |  |
| time | 请求时间 | 是 | int | 时间戳 |
| sign | 验签 | 是 | string | 生成方式见底部 |
| username | 账号 | 是 | string |  |
| password | 密码 | 是 | string |  |

**返回参数**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **元素名称** | **父元素名称** | **约束** | **元素描述** | **类型** | **取值说明** |
| code |  |  | 状态码 | int | 默认0为成功 |
| msg |  |  | 错误信息 | string |  |
| data |  |  | 返回数据 | object |  |
| token | data |  | 登录状态授权码 | stting |  |
| key | data |  | 验签生成算法key | string | 此处使用3Des对称加密，客户端需要解密后加入验签算法 |

## 1.2 登陆接口 （用户每次登录，token、key都会更改）

接口地址 ：post /login

**传入参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **元素名称** | **元素名称** | **是否必传** | **类型** | **取值说明** |
| pf | 系统标识 | 是 | int | 1、安卓 ： 2、ios |
| ver | App版本号 | 是 | string |  |
| time | 请求时间 | 是 | int | 时间戳 |
| sign | 验签 | 是 | string | 生成方式见底部 |
| username | 账号 | 是 | string |  |
| password | 密码 | 是 | string |  |

**返回参数**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **元素名称** | **父元素名称** | **约束** | **元素描述** | **类型** | **取值说明** |
| code |  |  | 状态码 | int |  |
| msg |  |  | 错误信息 | string |  |
| data |  |  | 返回数据 | object |  |
| token | data |  | 登录状态授权码 | stting |  |
| key | data |  | 验签生成算法key | string | 此除使用3Des对称加密，客户端需要解密后加入验签算法 |

## 1.3用户信息获取接口

接口地址 ：get /user/get-info

**传入参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **元素名称** | **元素名称** | **是否必传** | **类型** | **取值说明** |
| pf | 系统标识 | 是 | int | 1、安卓 ： 2、ios |
| ver | App版本号 | 是 | string |  |
| time | 请求时间 | 是 | int | 时间戳 |
| sign | 验签 | 是 | string | 生成方式见底部 |
| token | 登录状态授权码 | 是 | string |  |

**返回参数**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **元素名称** | **父元素名称** | **约束** | **元素描述** | **类型** | **取值说明** |
| code |  |  | 状态码 | int |  |
| msg |  |  | 错误信息 | string |  |
| data |  |  | 返回数据 | object |  |
| id | data |  | 登录状态授权码 | int |  |
| username | data |  | 验签生成算法key | string |  |
| real\_name | data |  | 用户真实姓名 | string |  |
| mobile | data |  | 用户手机号 | string |  |
| email | data |  | 用户邮箱 | string |  |

## 1.4用户信息更新接口

接口地址 ：post /user/update

**传入参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **元素名称** | **元素名称** | **是否必传** | **类型** | **取值说明** |
| pf | 系统标识 | 是 | int | 1、安卓 ： 2、ios |
| ver | App版本号 | 是 | string |  |
| time | 请求时间 | 是 | int | 时间戳 |
| sign | 验签 | 是 | string | 生成方式见底部 |
| token | 登录状态授权码 | 是 | string |  |
| real\_name | 用户真实姓名 | 是 | string |  |
| mobile | 用户手机号 | 是 | string |  |
| email | 用户邮箱 | 是 | string |  |

**返回参数**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **元素名称** | **父元素名称** | **约束** | **元素描述** | **类型** | **取值说明** |
| code |  |  | 状态码 | int |  |
| msg |  |  | 错误信息 | string |  |
| data |  |  | 返回数据 | bool |  |

1. 验签生成规则 （可以自行将app\lib\helpers\ApiSign类接口方法修改为你合适的算法）

传递数据+key=value(密钥值)按照键升序排序拼接成string, 然后md5加密生成，例子如下：

Sign = Md5(a=Va&b=Vb&...&key=Vkey&....z=Vz )

A、客户端登录、注册密钥（保存在客户端本地，不会改变）



可以在后台获取密钥保存到客户端。初步设计为，每升级一个版本，在后台添加一个记录，添加记录时候系统会随机分配一个密钥。如果登录注册接口有升级，客户端需获取保存该密钥到本地以生成登录、注册验签。

B、用户个人密钥 （用于登录成功后的接口验签生成算法）

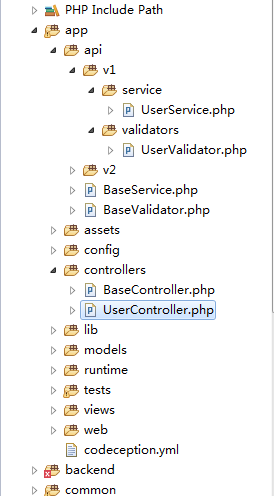
用户注册、登录信息成功都会返回，不过用户每次登录密钥信息都会改变，密钥数据为3Des（如果不合适，你可以自己设计合适的加密算法）对称加密数据，客户端需要解密后拼接到验签生成算法中。

1. 返回错误码和信息说明（具体错误码放到Error类，然后具体错误信息放到ErrorHandler 中）

我已经对ErrorHandler 组件进行了重写，我主要重写了渲染异常的接口方法，然后添加了一个接口异常类app\lib\helpers\ApiException和公共错误码类common\lib\helpers\Error，建议把错误码作为常量写到公共错误码类，然后通过错误类调常量方式使用错误码，以便系统错误码统一。当发生问题可以通过抛出ApiException 返回给客户端信息，你需要传递错误码给异常类，如果有自定义错误信息也可以传递到异常类中。如果没有，你可以定义错误码大概问题信息到错误组件，组件会匹配你该错误码信息。然后ErrorHandler 组件会自动转化数据为json格式数据传给客户端。

\*建议：不要对ErrorHandler做太多修改，只修改渲染方法，即可返回合适的json格式数据。如果需要修改，尽量保留日志记录逻辑，以便出现问题时候进行问题追溯。

1. 关于接口



目前很多有经验的人都会在mvc基础上引入多一层service层处理逻辑代码，减轻一般开发mc的开发冗余杂乱情况，同时提高代码复用。所以我引入了该理念，处理接口开发和很多web开发中。上图api相当service大层(放置每个版本接口的地方)，然后根据不同的接口版本，又分出多层对应不同接口版本的v1 ....... vn目录, 然后在不同版本下又分出小层validators（目录下放置参数验证类）和service（目录下放置逻辑数据处理提供服务类），这样将各接口版本数据验证逻辑和数据处理提供逻辑分离开，不用修改旧接口而提供给旧接口稳定性，同时保证升级（或者新）接口简单快捷开发，只要新接口新建V版本层,然后提供一样名字的service类实现一样的接口方法即可。

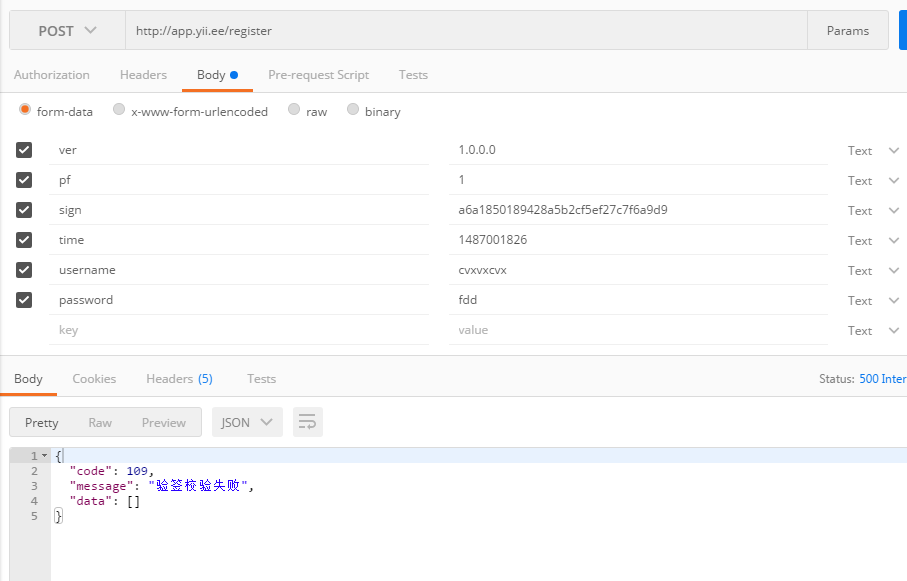
1. 具体验签校验、参数过滤、实例化服务类逻辑都放到app\lib\behaviors\RequestControl行为类中，可以到那查看，基本认为每个控制器对应一个相同名字service类。

5、 1.1、1.2、1.3为v1版本接口，1.4为v2版本接口。另外设计为客服端升级一个版本后台就要填相应记录和该版本对应接口版本，如图所示

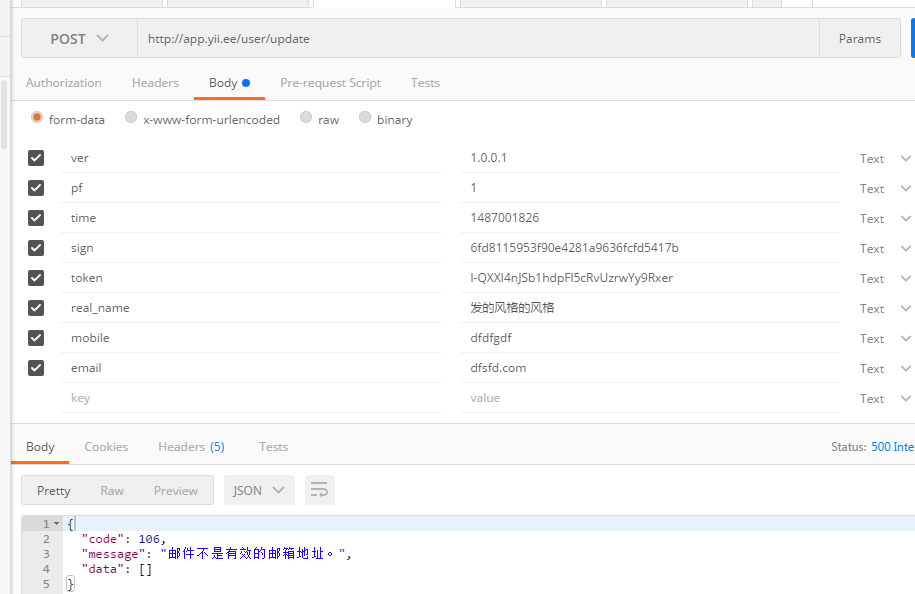


下面为请求例子图片

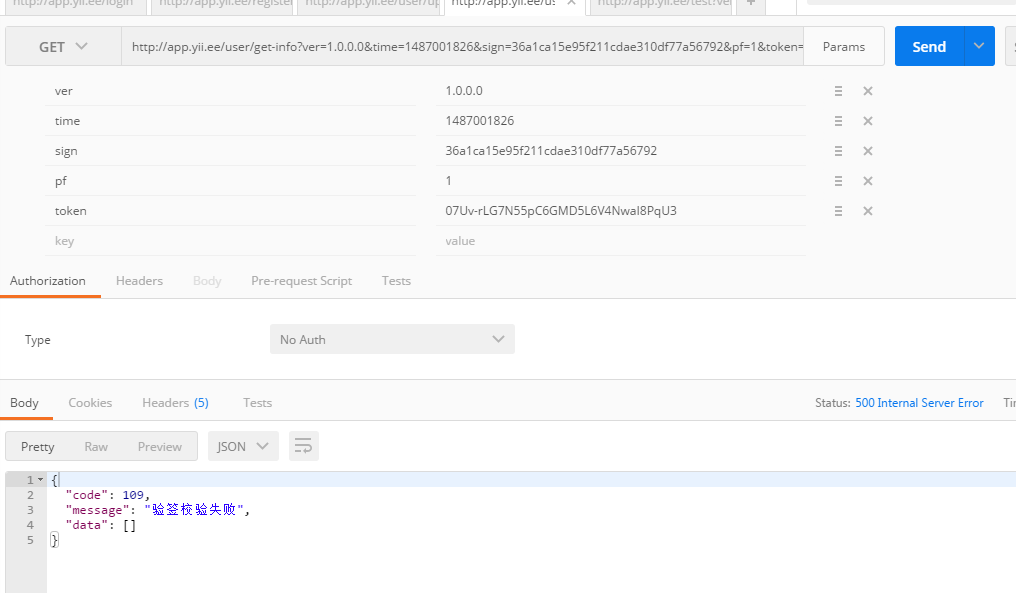
1.1 注册



1.2登录



1.3信息获取



1.4 信息修改

