HaoConnect

简介

HaoConnect 和 HaoResult ， 是HaoConnect框架中主要的两个类。

结构

```
HaoConnect
├── HaoConfig-example.php              配置文件范例（每次新项目，请重命名为HaoConfig.php并填写相关配置）
├── HaoConfig.php
├── HaoConnect.php                     请求接口基类，所有的Connect都是继承于此，也可以直接使用HaoConnect::loadJson的方法请求
├── HaoHttpClient.php                  数据请求公共库类，封装了curl的相关操作。
├── HaoResult.php                      接口返回的数据结果的封装对象，支持get、find、search等方法来取数据。
├── HaoUtility.php                     用到的些公共方法。
├── connects
│   └── UserConnect.php                拓展出来的Connect操作类。
├── readme.md
└── results
    └── UserResult.php                 拓展出来的Result实例对象。
```

HaoHttpClient.php

```php
<?php
    /*发送http请求，返回结果*/
    static function loadContent    ($pUrl, $pPostData=null, $pMethod = null, $pHeader=null, $pTimeout=30, $pResult='body
    /*发送http请求，返回Json数组*/
    static function loadJson       ($pUrl,  $pPostData=null,$pMethod=null, $pHeader=null, $pTimeout=30, $pResult='body')
    /*将头信息拆分成字典*/
    static function getHeaderList  ($pHeader=array())
    /*将参数信息拆分成字典*/
    static function getDataList    ($pData=array(),$pLink='')
?>
```

HaoConfig-example.php

在这里，由开发者设定对应的接口配置，甚至可以自行根据业务逻辑和开发环境，配置不同的信息。

```php
<?php
define('HAOCONNECT_CLIENTINFO',         'haoFrame-client');             //应用信息
define('HAOCONNECT_CLIENTVERSION',      '2');                           //使用本类所在客户端的版本号
define('HAOCONNECT_SECRET_HAX',         'secret=apio3i4089037arkefwap'); //加密秘钥，这里用的是2号设备类型的密钥
define('HAOCONNECT_APIHOST',            'api-haoframe.haoxitech.com');   //接口域名，建议根据正式服或测试服环境分别赋值
?>
```

HaoConnect.php

HaoConnect类中提供了 loadResult 的静态方法，用于发起请求到API服务器，并将返回结果转化成Result对象。

开发者也可以继承并拓展HaoConnect类，来定制更多方法，如UserConnect::login()。

```php
<?php
class HaoConnect {
    /* 保存用户信息到客户端 */
    static function setCurrentUserInfo($Userid='0',$Logintime='0',$Checkcode='0')
    /* 读取用户信息 */
    static function getCurrentUserInfo()
    /* 根据链接、参数、头信息，取得加密后的密钥 */
    static function getSignature($pLink='',$pPostData = array(),$pHeader = array())
    /* 请求API地址，获得字符串 */
```

```
    static function loadContent($pControllerAction, $pPostData = array(), $pMethod = 'get', $pHeader=array())
    /* 返回HaoResult */
    static function loadResult($pControllerAction,  $pPostData = array(),$pMethod = 'get', $pHeader=array())
}
?>
```

UserConnect.php

```
<?php
class UserConnect extends HaoConnect {
    /*
     * 联合登录
     * @param  array $pParms  参数
     *                        union_type       int          登录方式：2QQ 3微博 4微信
     *                        union_token      string       联合登录唯一识别码
     * @return UserResult
     */
    public static function actionUnionLogin($pParms=null)
    {
        return static::loadResult('user/union_login',$pParms,'post');
    }
}
?>
```

HaoResult.php

HaoResult是非常重要的模型类，所有的model数据都会被包装成对应的Result对象。其中Result里的errorCode是会被传递到自身包含的数据新转化的Result对象里的哦。

开发者也可以继承并拓展HaoResult类，如实现UserResult类等。

```
<?php
class HaoResult {
    /* 将数组初始化成对象 */
    static function instanceModel($results,$errorCode,$errorStr,$extraInfo,$resultCount)
    /* 更新 */
    function update($key,$value)
    /* 在结果中进行搜索，返回结果是数组（至少也是空数组） */
    function search($path)
    /* 根据路径取数据，默认是results中取，也可以指定extraInfo>路径下取数据。 */
    function find($path)
    /* 传入值如果是model，则以当前Result为框架构建新Result。如果是model组成的数组，则返回result组成的数组。否则直接返回。 */
    function value($value)
    /* 判断当前实例是否目标model */
    function isModelType($modelType)
    /* 判断是否等于目标ErroCode*/
    function isErrorCode($errorCode)
    /* 判断是否正确获得结果*/
    function isResultsOK()
}
?>
```

UserResult.php

继承之HaoResult，明文定义了其属性的get方法，方便开发时直接调用。

```
<?php
class UserResult extends HaoResult {
    public function findTelephone()
    {
        return $this->find('telephone');
    }
    public function updateTelephone($telephone)
    {
        return $this->update('telephone',$telephone);
    }
    ...
```

```
}
?>
```

HaoUtility.php

工具类，在HaoConnect开发过程中，会用到一些方法，就放在这里吧。

```
<?php
class HaoUtility {
    /* 判断目标变量是否某类型的model */
    function isModelTypeWithTarget($target,$modelType)
    /* 将数组里的key路径遍历取出 */
    function getKeyIndexArray($targetArray)
    /* 字符串转换。驼峰式字符串（首字母小写） */
    function camelCase($str)
    /* 字符串转换。驼峰转换成下划线的形式 */
    function under_score($str)
}
?>
```

用法举例

```
<?php
    /** @var UserResult [description] */
    $tmpResult = UserConnect::login(array('account'=>'13774298448','password'=>md5('123456')));

    $tmpResult = UserConnect::login(array('account'=>'13774298448','password'=>md5('123456')));

    if ($tmpResult->isResultsOK())
    {
        if (HaoUtility::isModelTypeWithTarget($tmpResult , 'user'))
        {
            //get属性
            print("\n\n\n");print('__get元素___');print("\n\n\n");
            printf("id               : %s \n\n\n",$tmpResult->getId());
            printf("telephone        : %s \n\n\n",$tmpResult->getTelephone());
            printf("telephoneLocal   : %s \n\n\n",$tmpResult->getTelephoneLocal());
            printf("username         : %s \n\n\n",$tmpResult->getUsername());
            printf("email            : %s \n\n\n",$tmpResult->getEmail());
            printf("level            : %s \n\n\n",$tmpResult->getLevel());

            //find元素
            print("\n\n\n");print('__find元素___');print("\n\n\n");
            printf("status           : %s \n\n\n",$tmpResult->find('status'));
            printf("lastLoginTime    : %s \n\n\n",$tmpResult->find('lastLoginTime'));
            printf("lastPasswordTime : %s \n\n\n",$tmpResult->find('lastPasswordTime'));
            printf("createTime       : %s \n\n\n",$tmpResult->find('createTime'));
            printf("modifyTime       : %s \n\n\n",$tmpResult->find('modifyTime'));
            printf("modelType        : %s \n\n\n",$tmpResult->find('modelType'));

            print("\n\n\n");print('__find整个results，注意，这里会被转换成对应的新的Result实例___');print("\n\n\n");
            var_export($tmpResult->find('results'));

            print("\n\n\n");print('__find整个附加数据___');print("\n\n\n");
            var_export($tmpResult->find('extraInfo'));


            print("\n\n\n");print('__find extraInfo 下的路径___');print("\n\n\n");
            printf("extraInfo>authInfo>Userid       : %s \n\n\n",$tmpResult->find('extraInfo>authInfo>Userid'));
            printf("extraInfo>authInfo>Logintime       : %s \n\n\n",$tmpResult->find('extraInfo>authInfo>Logintime'));
            printf("extraInfo>authInfo>Checkcode       : %s \n\n\n",$tmpResult->find('extraInfo>authInfo>Checkcode'));

            print("\n\n\n");print('__find results 下的路径___');print("\n\n\n");
            printf("results>lastPasswordTime       : %s \n\n\n",$tmpResult->find('results>lastPasswordTime'));

            print("\n\n\n");print('__find results 下的路径，可以不传results___');print("\n\n\n");
            printf("lastPasswordTime       : %s \n\n\n",$tmpResult->find('lastPasswordTime'));
```

```
            //search
            print("\n\n\n");print('__search，使用正则的方法去匹配路径，所以返回结果是个数组___');print("\n\n\n");
            print(".*?lastLoginTime   search    :  \n\n\n");
            var_export($tmpResult->search('.*?lastLoginTime'));
        }
    }
    else
    {
        print($tmpResult->errorStr);
    }
?>
```

TODO

- API缓存（使用文件存储）
- 请求内缓存 （使用内存）