## ZJApi\_ByCI

### 总体架构说明

增加Service层

* 由于CI没有Service层，一般都会把业务逻辑写到Controller里或者Model里，这样会造成业务层和控制层、业务层和持久层相互耦合，显然这是十分不合理的，**因此直接改写了CI框架代码增加了Service层**。
* Model(DAO):数据持久层的工作，封装对数据库的操作，可供一个或者多个Service一次或者多次调用。
* Service ： 业务逻辑层，负责业务模块的逻辑应用设计，controller中就可以调用service的接口实现业务逻辑处理，提高了通用的业务逻辑的复用性，设计到具体业务实现会调用Model的接口。**（PS：比如说用户下订单，这必然会有一系列的操作：更新购物车、添加订单记录、会员添加积分等等，且下订单的过程可能在多种场景出现，如果这样的代码放Controller中则很臃肿难以复用，如果放model会让持久层和业务层耦合。）**
* Controller ：控制层，负责具体业务流程控制，这里调用service层，将数据返回到视图。
* View ： 负责前端页面展示，与Controller紧密联系。

多环境兼容

* 多环境兼容：通过切换“**index.php**” 文件中的“**ENVIRONMENT**”常量进行处理
* development：开发环境；testing：测试环境；production：生产（发布）环境
* 要创建特定环境的配置文件，新建或复制一个配置文件到application/config/{ENVIRONMENT}/{FILENAME}.php。例如：application/config/production/config.php。当你将 ENVIRONMENT 常量设置为 ‘production’ 时，你新建的生产环境下的config.php 里的配置将会加载。（**CodeIgniter 总是先加载全局配置文件（例如， application/config/ 目录下的配置文件），然后再去尝试加载当前环境的配置文件。这意味着你没必要将所有的配置文件都放到特定环境的配置目录下，只需要放那些在每个环境下不一样的配置文件就可以了。**）

多平台兼容

* 多环境兼容（目前支持Local和SAE）：通过切换“**index.php**” 文件中的“**SERVER\_PLAT**”常量进行处理
* Local，兼容开发：
* SAE，**修改CI框架直接支持sae环境**：
* memcache兼容SAE平台的memcache：get、set（需要其他操作要再封装测试！）
* redis兼容SAE平台的KVDB：get、set（需要其他操作要再封装测试！）

memcache方面

* memcache调整：
* 不使用save方法，改为使用通用的set方法，如：

$this->cache->memcached->set('foo', 'bar', 10);

* memcache说明：
* 一般在构造方法中引入缓存驱动：

$this->load->driver('cache');（**只需引入一次，即可使用memcache和redis**）

redis方面

* redis调整：
* 不使用save方法，改为使用通用的set方法，如（**SAE平台使用KVDB代替redis，第三个参数“过期时间”无效**）：

$this->cache->redis->set('redisSet', 'redisValue', 10);

* redis说明：
* 一般在构造方法中引入缓存驱动：

$this->load->driver('cache');（**只需引入一次，即可使用memcache和redis**）

多应用部署

* 新建“template”目录，并把原来的“application”目录和“index.php”文件放入其中
* 修改此“index.php”文件，把“$system\_path”改为“../system”
* 修改“application/config/config.php”文件，把“$config['index\_page']”改为“'../template/index.php'”（每复制“template”目录增加一个新app时都需要进行此步骤，并其中template换成增加的app的名字，即“'../template/index.php'”，““'../app/index.php'””）

rewrite规则

* 各平台都需要增加rewrite规则
* sae上需要修改“config.yaml”文件，增加rewrite规则，注意顺序！例如：

- rewrite: if (!-f && %{REQUEST\_URI} ~ "template/(.\*)") goto "/template/index.php/%1"

- rewrite: if (!-f && %{REQUEST\_URI} ~ "app1/(.\*)") goto "/app1/index.php/%1"

* PS:如果不在sae上部署，只要增加.htacess文件即可

autoload自动加载类库

* 自动加载数据库：
* application/config/autoload.php 中的 library 数组里添加 :$autoload['libraries'] = array('database');
* $this->load->database();就都不用加了
* 自动加载my\_curl
* application/config/autoload.php 中的 library 数组里添加 :$autoload['libraries'] = array(my\_curl);
* 自动加载my\_des
* application/config/autoload.php 中的 library 数组里添加 :$autoload['libraries'] = array(my\_des);
* 自动加载my\_response
* application/config/autoload.php 中的 library 数组里添加 :$autoload['libraries'] = array(my\_response);
* 自动加载my\_config
* application/config/autoload.php 中的 library 数组里添加 :$autoload['config'] = array(my\_config);

autoload自动加载辅助函数

* 自动加载url、html辅助函数
* application/config/autoload.php

$autoload['helper'] = array('url', 'html');

增加自定义类库

* 增加CI\_My\_curl**（已配置自动加载）**：
* 支持$this->my\_curl->doCurlGetRequest($url)
* 支持$this->my\_curl->doCurlPostRequest($url)
* 增加CI\_My\_des（掌金des加解密库）**（已配置自动加载）**：
* 支持$this->my\_des->encode("WhJSN0B9CTSAZAG84XfFIJMkzlf5CzPZ", "sycs84ju");
* 支持$this->my\_des->decode($re, "sycs84ju");
* 增加CI\_My\_response**（已配置自动加载）**：
* 增加MyApi**（已配置自动加载）**：方便调用老的api

关于返回结果集

* 配置错误码：
* config/my\_config.php中配置错误码（已配置自动加载）
* 框架级错误码：1-401，例如： my\_error\_code401
* 应用级错误码：402-1000，例如：my\_error\_code501
* 配置正确返回码：1001-2000，例如：my\_success\_code1000

关于hooks

* hooks中会校验参数中是否带token和os，针对不需要hooks的Controller或者Action提供支持，在“template/application/config/my\_config.php”中的“$config['my\_hooks\_ignore']”进行配置

关于参数

* 入参：desencode(jsonencode(@p))
* @p->token:
* 接口需要加token接口过滤参数，见“总体技术规范”

http://192.168.10.243/zj\_Frameworks/PHP/ZJApi\_ByCI/template/Z\_test/testToken

* @p->os:
* 接口需要加os参数，见“总体技术规范”
* 取参数:My\_Base\_Controller→preDealPara($para, $arr = array())

forWeb前端的特殊处理

* 目录结构有所区别，见“2.开发规范及步骤——目录结构”
* autoload.php中不要配置自动加载“database”

### 开发规范及步骤

关于框架

* 总体要求见“总体技术规范→应用程序框架规范”，应用开发工程师不能调整以下文件
* system目录
* application→application→config→hooks.php
* application→application→controllers→My\_Base\_Controller.php
* ...等

新建应用开发

* 复制整个框架，把“template”重命名为本应用的名字（技术主管把握），后续参见“总体架构说明→多应用部署”

目录结构

* 单应用目录结构为（**一般为web目录**）
* system
* application

<script type="text/javascript" src='<?=base\_url().'assets/js/Ajax.js'?>'></script>

* assets
* index.php
* controllers下一般不再分文件夹
* My\_Base\_Controller extends CI\_Controller
* 所有自己写的Controller必须继承My\_Base\_Controller
* models下一般不再分文件夹
* services下一般不再分文件夹
* views下要求按照controller分文件夹