# Base Cms开发文档

## 基本规范

### 路由与命名

类名，控制器，模块首字母大写，如：

classKillPeople

{

}

函数名，变量名请使用驼峰法，如：

function addUserToDatabase();

returnData = [];

路由名称:

想给一个例子

Router::get("/admin\_sUserPowerGroup","Admin\PowerController@sUserPowerGroup");

Router::get("/模块\_函数名","模块\控制器@函数名");

在访问路径中使用驼峰命名，如admin只需要admin\_函数名，若模块叫normal user 那么就为normalUser\_函数名

其中约定几个简写select(s),update(u),delete(d),add(a)来标识一些常用操作

如果要访问一个路由，请使用/xxxxx的绝对路径，避免路由带来的调试测试困难

请尽量详细描述该路由的功能，即使它很长

一个页面应该隔开一行，一组功能应该靠近，如：

Route::get("/admin\_sUser","Admin\PowerController@sUser");//查看所有的用户

Route::post("/admin\_aUser","Admin\PowerController@aUser");//添加一个用户

Route::post("/admin\_dUser","Admin\PowerController@dUser");//删除某个用户

因为上面的页面可以进行两个操作：添加，删除，这两个功能只是在服务器处理并不需要页面，所以归为一组，不用空行断开。

同时，通过这种方法，可以确定一个页面的功能，起到项目文档注解的作用。

## 基础开发

为了使数据库开发独立于业务逻辑，所有的数据库相关的操作和复杂的逻辑操作都必须写到类库里面：当前我们开发的基础类库的位置在/gird\_plugins/Base/class/底下，每个人负责的模块都有相应一个类，如，开发权限的同学使用PowerFunc.php中的PowerFunc编写一些功能代码，如，你需要做一个检查权限的功能，就在里面写如下：

class PowerFunc

{

public function checkAdminPower($powerId)

{

}

}

在函数中命名注意遵循命名规范

然后由于laravel提供了自动加载的功能，你只需要简单的在控制器中调用：

public function powerTest(BaseFunc $power)//在这里要声明一下

{

$powe->checkAdminPower($power\_id);//调用

}

函数的命名尽量贴近其执行的动作。

## 团队合作基础

每个开发成员如果遇到需要调用数据库的操作，且该操作不归你负责，那么就需要去查询相关模块的，每个成员请把自己的接口文档写好，写在/Document/Base中，以你的模块名命名，如Power.doc,请先给出一些关键的接口，方便其他人调用此接口（即使你还没有实现）。

Document中给出一些基础方法的文档。

## 测试

每个成员请保证自己的类中所有方法经过了测试，且用户层交互界面无错误。

## 注释与文档写法：

我知道大家都不想写文档，但是注释还是要写的！

这里给出一个示范

对于扩展包中的函数，请使用phpdoc规范的注释方法，一般的控制器方法，请简单的说明即可。

/\*\*

\* 在提交接收页面以后，将错误/正确提示信息保存于session，并在下一页面调用showRedirectMessage()将信息显示在指定位置

\* @access public

\* @param bool $status 正确/错误

\* @param string $message 提示信息

\* @param string $plugin 需要额外加入的页面组件，如链接按钮等，显示在信息框底部

\* @param string $redirect 如果需要顺便跳转到某个页面，可以将其url填入，如果为空，则忽略不跳转

\* @return NULL/直接跳转

\*/

public function setRedirectMessage($status,$message,$plugin,$redirect=NULL)

{

//函数

}

每一个函数都请写好文档~！