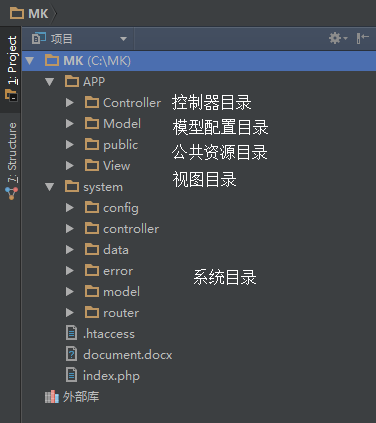
**MakePHP 1.0目录结构**



**控制器命名规则**：

1. 一个控制器对应唯一一个PHP文件。
2. 控制器的PHP文件名格式必须为 控制器名+Controller。
3. 控制器的类名必须和文件名完全一致。

**控制器类定义规则：**

1. 控制器类必须继承于controller。
2. 控制器类必须有一个Index方法。
3. 控制器类必须定义命名空间，命名空间的格式必须是 APP名称+‘\Controller’（即Controller文件相对根目录的路径）。

如： namespace APP\Controller

4.控制器类必须使用system\controller\controller命名空间，即controller类所在的命名空间。

**载入视图：**

可在控制器中调用display方法

如：

要载入View目录下的login.html或 login.php页面，可以这样写:

$this->display(‘login.html’);

$this->display(‘login.php’); (注意需要加上后缀名)

**向视图传递变量:**

如果在视图中需要引用控制器中的变量，必须在控制器调用display()方法之前调用assign()方法允许指定的变量在视图中使用。

函数原型：assign([变量名]<变量值>);

如：

控制器中有一个$username变量，您如果想在bg.html视图中使用该变量，应该首先在控制器中使用如下语句：

$this->assign(‘username’,$username);

$this->display(‘bg.html’);

然后在bg.html中，就可以使用$username变量了：

<php? echo $username; ?>

**给变量重命名：**

assign([变量名]<变量值>)的第一个参数为在视图中使用的变量名，第二个参数为该变量的值。

如：

控制器中有一个$username变量，您如果想在bg.html视图中以$uName访问该变量值，应该首先在控制器中使用如下语句：

$this->assign(‘uName’,$username);

$this->display(‘bg.html’);

然后在bg.html中，就可以以$uName使用该变量了：

<php? echo $uName; ?>

**一次性传递多个变量：**

assign()方法还支持以数组的形式向视图传递多个变量，这种情况下，assign()的参数只能是一个数组。

如：

控制器向视图bg.html传递$username、$userlevel、$usergroup，您可以在控制器中连续写3行assign();

$this->assign(‘username’,$username);

$this->assign(‘userlevel’,$userlevel);

$this->assign(‘usergroup’,$usergroup’);

这种方法不仅繁琐，效率也更低。另一种解决方案是:

$vars=array(‘username’=>$username,’userlevel’=>$userlevel,’usergroup’=>$usergroup);

$this->assign($vars);

或者

$this->assign(array(‘username’=>$username,’userlevel’=>$userlevel,’usergroup’=>$usergroup));

相应的，在采用该方式传递一个或多个变量的时候，数组元素的元素键值key,就是在视图中引用变量的变量名。

**向视图传递对象：**

当您需要向视图传递单个对象的时候，和传递单个变量一样，您必须指定视图中访问该对象的对象名。当您需要传递多个对象的时候，不管用N句assign()，或者array方式，您都必须要指定对象名称。

**模型：**

**配置：**

数据库配置信息必须放在您的APP目录下的 \Model\conf\config.php文件当中，且必须完成以下字段的配置：

‘user’,’host’, ’password’,’database’,’port’。

如：

<?php

return array(

'user'=>'root',//用户名

'host'=>'localhost',//主机名

'password'=>'root123',//登录密码

'database'=>'makephp',//数据库名

'port'=>3306//数据库端口

);

?>

**单例：**

MakePHP的模型是一个单例类，因此只会在第一次使用模型变量的时候才会进行数据库连接，优化了程序效率。

**使用模型对象：**

$con = M(‘login’);

全局函数M()原型：M(<表名>)。改函数返回一个指向指定表的模型对象。

**连贯操作：**

MakePHP支持以 -> 方式的连贯操作，如：

$con = M(‘user);

$reData = $con->where(‘level > 3’)->limit(0,10)->select();//除动词方法（这里的select,更多动词方法请参考动词方法表）不支持连贯操作之外，其它方法都支持连贯操作，且顺序可以自定义。动词方法 如 select(),craete(),add(),delete()等，必须放在语句末尾。

**数据操作方法：**

|  |  |  |  |
| --- | --- | --- | --- |
| **方法名** | **支持的参数类型** | **示列** | **功能** |
| **Where** | 数组/字符串/数值 | $con->where(array(‘id’=>1))->select(); | 同SQL where |
| **field** | 数组/字符串 | $con->filed(‘name as nickname’)->select(); | 获取指定字段值，可指定多个 |
| **limit** | 数组/字符串/数值 | $con->limit(0,5)->select(); | 同SQL limit |
| **Order** | 数组/字符串 | $con->order(‘level desc’)->select(); | 同 ORDER BY |
| **Group** | 数组/字符串 | $con->field(‘SUM(score)’) ->group(‘name’)->select(); | 同 GROUP BY |
| **table** | 字符串 | $con->table(‘user’); | 修改数据表，同$con = M(‘user’); |

**参数缓存机制：**

MakePHP模型采用参数缓存机制，即：在调用任何动词方法之前，如果有设置任何参数，那么在下一次调用动词方法时，会影响到该次查询。比如：

$con = M(‘user’);

$con->limit(10);

$con->where(array(‘name’=>’admin’));

$reData = $con->select();

执行以上代码后，$reData中的值，并不是user 表中的所有记录，而是存储的所有 name字段值为 ‘admin’ 的前10条记录，相当于 $con->where(array(‘name’=>’admin’))->limit(10)->select();

而在任何动词方法调用完毕后，缓存中的参数都将被清除。例如：

$con = M(‘user’);

$con->limit(10);

$con->where(array(‘name’=>’admin’));

$reData = $con->select();

$reData = $con->select();

在执行完以上代码后，$reData 变量的值为 user 表中所有的记录。因为预先设置的参数，都在第一个select()方法执行完毕后全部清除了，这时候再次不加任何参数地调用select()动词方法，就相当于’select \* from user’。

**资源访问宏：**

在视图中访问指定资源，如：css，js脚本，图片，视频，文件等资源时可以在html中使用\_\_SRC\_\_宏来访问public目录下的资源，示例：

<img src="\_\_SRC\_\_/img/makePHP.jpg" class="exampleImg">

<link rel="stylesheet" href="\_\_SRC\_\_/css/index.css" />

<script type="text/javascript" src="\_\_SRC\_\_/js/common/mobileCommon.js" ></script>

<source src="\_\_SRC\_\_/./video/MVI\_7651.webm" type="video/webm"></video>

MakePHP 目前支持的可访问资源类型都存放在 system/config/headerTypeHelper.php中，可以自行修改、添加类型。