# Bee PHP开发文档

# 目录结构

### /index.php

为方便表示，这里定义‘/’为框架主目录，此时的index.php文件是入口文件，里面定义了很多常用的路径常量，有：

BEE:为当前框架所在的绝对路径目录(/)

LIB:库文件路径(/beelib)

APP: 视图控制器模型目录(/app)

CONF：配置文件目录（/app /config）

SELFLIB: 自定义库(/beelib/extlib/self)

PUBLICRES: 存放CSS 、JS、图片等资源(/public)

将加载类include到其中。在后面框架开启后，可以加载对应的类文件。

### /public

此文件是存放JS和CSS代码，以及一些图片，字体等资源。

### /beelib/lib

存放框架库的目录，有控制器类（action.class.php），模型类（model.class.php）,路由类（route.class.php），加载类（load.class.php）,框架启动类（beePhp.class.php）,配置类（conf.class.php）

### /beelib/extlib/composer

放composer第三方库的目录，自带composer.json文件。

### /beelib/extlib/self

放开发者写的库类的目录。

### /app/config

存放配置文件，用框架，要先写配置文件。

### /app/controllers

控制器目录

### /app/models

模型目录，一个模型对应一张表

### /app/views

视图目录，一个控制器对应一个视图文件夹，文件夹名字为对应控制器的名字。

# 配置文件

Db\_type 数据库类型，Db\_Host 数据库地址，Db\_Name 数据库名字，Db\_User 用户名，Db\_pwd 密码，PDO\_islong PDO是否使用长连接，Db\_set 数据库编码，Display\_Error 是否显示错误提示 。

也可以自己写配置变量，只需在后面加上key = value,每个键值对占一行。

通过conf::instance()静态方法，来获取配置的关联数组，conf::instance()[‘key’],能获得value。

# 路由

假设URL为xxx.com/controller/action/key/val/key/val，这里controller就是要创建的控制类，action是当前控制器类内的函数名，默认控制器和函数都是index，key/val可以用$\_GET[‘key’]=val,来获得。

# 模型

创建模型类，只需在/app/models里创建一个php文件，假设创建的是xxx.class.php，则在里面必须定义一个xxxModel的类，该类必须继承Bee\_Db\_Table类（也可以换成别的，在后面的扩展类篇中会详解），类中要重写2个protected的 变量，$\_name模型对应的表名，$\_primary主键名字默认是id。Bee\_Db\_Table类提供了数据库的curl操作。

insert($data)是插入数据函数，$data必须是数组，如果不是函数会返回-1不进行任何操作，$data格式为：array(set1=>val1,set2=>val2,….),set为字段名，val是对应插入的数据，操作成功会返回1失败则为0。

update($set,$where)是更新函数，$set是更新的字段和数据组成的数组，格式和上面的insert是 一样的，不是数组还会返回-1不进行任何操作。$where是限制条件。$where在使用前更新函数需要处理一下，用的是getAdapter()->quoteInto($where,$value)函数，这里$where格式为‘set = ?’,$value是set对面的值，函数会对$value进行防sql注入处理，返回是一个字符串，把改字符串放入update的$where即可，操作成功会返回1失败则为0。

delete($where)是删除数据函数，$where同样要用getAdapter()->quoteInto($where,$value)做预处理操作成功会返回1失败则为0。

find($primary)是根据主键查找数据，$primary可以是数组和数字，数组格式array(id1,id2,id3),会返回一个result二维数组，对应每个id的数据。

fetchRow($where = NULL,$order = '\*')是取回一条数据，$where可以是数组或者像上面预处理过的字符串。$order是要取回数据的字段。

function fetchAll($where,$order,$count = 1,$offset = 0)是取回自定义数据，$where可以是数组或者像上面预处理过的字符串，$order是要取回数据的字段，$count 是取回数据条数，$offset是偏移量，转化成sql语句就是limit $count, $count+$offset。

自定义查询语句，在模型类中的适配器中，定义了一系列查询字符拼接函数，select（），from（），where（），group（），having（），order（），limit（）。拼接完的sql语句可以用get\_sql（）获取。然后用模型类中的get\_select\_res（）来执行。

具体用法请看第六部分的样例代码演示。

注：如果对文字描述看不懂，可以最后的代码例子。

# 控制器

创建一个路由器，就是在/app/controllers目录中创建一个控制器类php文件，文件名字规范为xxxxController.class.php,这里的xxxx为控制器名字。在文件需要定义xxxxController类，名字和文件名前面是一样的，名字必须一样，如果出错就无法加载该类，就会导致路由错误，定义xxxxController类必须继承Bee\_Controller\_Action类，xxxxController类中函数的名字必须严格按照xxxxAction，这里的xxxx就是上面路由的URL中提到的action。

在Bee\_Controller\_Action类提供了很多有用的函数，下面一一介绍。

getModel($modelName)获取一个模型对象，$modelname是对应模型的名字（后面不需要加Model）。

render($viewName = 'index')加载一个视图，由于一个控制器对应的是一个视图文件夹，在文件夹里可以放多个视图文件，视图文件的命名规范是xxxx.phtml，这里的xxxx是视图名称，就是对应函数中的变量viewName。

view($key,$val)，把数据传入到要加载的视图中，视图中获取数据的方法是直接 $this->view[$key]能获取对应的$val。

redirect($url)，能跳转到外网和别的控制器的视图。

# 函数样例代码

class testModel extends Bee\_Db\_Table {

protected $\_name = 'test ';

protected $\_primary = 'id ';

}

class indexController extends Bee\_Controller\_Action{

public function indexAction()

{

### 1.插入数据

$data = array(

'noble\_title' => 'King',

'first\_name' => 'Arthur',

'favorite\_color' => 'blue'

);

获取表模型类

$table = $this-> getModel('test');

$table->insert($data);

### 2.用适配器预处理

$adapter = $table->getAdapter();

$set = array(

'favorite\_color' => 'yellow',

)

$where=$ adapter ->quoteInto('first\_name = ?', '\\/Robin');

### 3.更新数据

$rows\_affected = $table->update($set, $where);

### 4.删除数据

$where = $adapter ->quoteInto('first\_name = ?', '\\//\Patsy');

$rows\_affected = $table->delete($where);

### 5.根据主键查找

$row = $table->find(1);

$rowset = $table->find(array(1, 2, 3));

### 6.获取一条数据

$where = $ adapter ->quoteInto('noble\_title = ?', 'Sir'). $adapter ->quoteInto('AND first\_name = ?', 'Robin');

$order = 'favorite\_color';

$row = $table->fetchRow($where, $order);

### 7.取回多条数据

$where = $ adapter ->quoteInto('noble\_title = ?', 'Sir');

$order = 'first\_name';

$count = 10;

$offset = 20;

$rowset = $table->fetchAll($where, $order, $count, $offset);

### 8.自定义查询

$adapter->select($set)

查询字段名，可以为数组，array(set1,set2…)

->from($name)

查询的表名

->where($set)

可以是用适配器处理过的字符串，也可是关联数组，array(set1=>val1,set2=>val2)。

->group($set)

分组字段名

->having($where)

直接传入字符串即可，不用做任何处理

->order($order)

可以直接使用字符串，也可以用数组array(set1=>desc,set2=>NULL)如果为空则表示升序。

->limit($count,$offset)

$count为数据条数，$offset为偏移量，和上面的fetchAll的最后两个参数作用是一样的。

用$adapter->get\_sql（）获取sql语句

然后用模型类的get\_select\_res（$sql）获取数据结果集。

### 加载文件

loadClass:: loadfile ($def,$type,$class,$suffix)

参数对应的目录为：$path = $def.'/'.$type.'/'.$class.$suffix;

例如加载框架启动类：loadClass::loadfile(LIB,'lib','beePhp','.class.php');

# composer的应用

在example文件夹中的框架包含了

"filp/whoops": "1.\*",

"catfan/Medoo": "^1.2",

"twig/twig": "\*"

这三个第三方库，错误展示库filp/whoops，数据库操作库catfan/Medoo，已经视图操作库。Twig为视图引擎。

这3个库的具体用法请开发者自行查阅相关文档。

catfan/Medoo只要模型类继承Bee\_Db\_Medoo即可。

Twig视图，是调用控制器的display方法即可，传入视图的名称，不需要加后缀，如果要创建视图，则视图后缀必须为.phtml。

Whoops是否启动在配置文件里设置是否显示错误提示即可。

如需加载其他类库，推荐一下几个启动地点。

/beelib/lib/beePhp.class.php中的start函数中，可以启动一些日志，错误显示，系统压力监听等库。

如需加载自己的数据库类，可以把文件放在/lib/extlib/self中，

在/lib/lib/action.class.php中改写，getModel()方法，来重新加载模型类。

# 域名重定向

详情可以看/doc文件里的域名重定向文档。

# 注意事项

创建控制前，视图和模型文件，命名要严格安装框架格式。

视图文件引入资源路径为：../../public/\*\*\*

本框架要求PHP版本为：5.4或及以上版本