## initialize和construct的不同

initialize是thinkphp中的方法，php中没有；

initialize: 子类自动调用父类的\_initialize方法；

\_\_construct:子类想要调用父类的构造方法，必须先调用parent::\_\_construct();

例：

class Father {

public function \_initialize(){

echo “I’m the father”;

}

}

class Son extends Father{

public function index(){}

}

$son = new Son;

$son->index();//自动调用Father的\_initialize()方法；

## import(‘类库命名空间字符串’ , ’起始路径’ , ‘类库后缀’)

导入系统基类库 即library/Think下的

1. 例：import('Think.Util.Array');

等同于：

require THINK\_PATH.'Lib/Util/Array.class.php';

导入扩展类库 Org等

第三方类库：Vender，用法跟import一样，只是它引入的类库都放在vender目录下面

详解见：http://www.thinkphp.cn/info/126.html

## 空操作

空操作是指系统在找不到请求的操作方法的时候，会定位到空操作（\_empty）方法来执行，利用这个机制，我们可以实现错误页面和一些URL的优化。

http://网址/index.php/Home/Main/hello 空操作

1. 在Main控制器里定义一个\_empty(){}方法

function \_empty(){

$this->display(‘error’);

}

在View/Main目录下建一个error.html页面；

当访问的操作不存在时，会自动走\_empty()方法；

## 空控制器

Home/Controller/EmptyController.class.php

<?php

namespace Home\Controller;

use Think\Controller;

class EmptyController extends Controller

{

public function \_empty()

{

echo "访问的页面不存在！";

}

}

当访问不存在的控制器，系统会尝试定位空控制器EmptyController；

## TP下的事务支持

事务是针对数据库本身的，所以可以跨模型操作的 。

|  |
| --- |
| 例：  // 在User模型中启动事务  $User->startTrans()  // 进行相关的业务逻辑操作  $Info = M("Info"); // 实例化Info对象  $Info->save($User); // 保存用户信息  if (操作成功){  // 提交事务  $User->commit()  }else{  // 事务回滚  $User->rollback()  } |