# 登陆原理

加载额外的配置文件LOAD\_EXT\_CONFIG

功能：实现用户登录原理：

1 用户输入账号，密码 点击登陆 触发onclick事件绑定 前端js事件

<type="button" onclick="login.check()">注意： type为button login.check的写法 而不是：

2 通过jquey获取账号密码 判断是否为空

var username = $('input name="username"]').val();接着实现服务器端的校验

$.post(url,data,function (result){ },'JSON');

注意：需要'json'的参数通过规则实现数据的格式检查

3最后将表单提交的数据移交给model进行数据查找

实现用户的登陆

# 退出功能

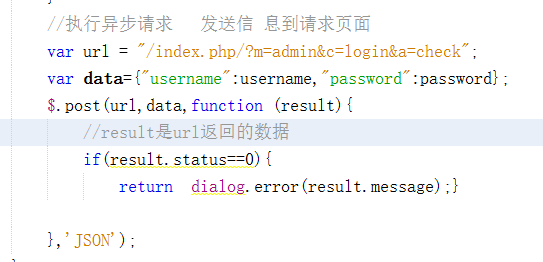
Session的清空 地址的跳转

通过检测session的值 来判断是否登录

# 关于ajx异步

同步：点击提交表单 页面将数据action到某个指定页面 特定的页面进行数据处理

异步:点击登陆以后 ajax获取到表单数据提交到某个页面进行数据校验

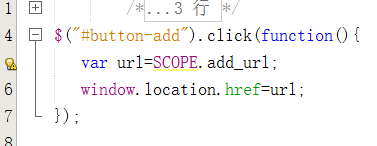


# 菜单管理的添加

一切操作从用户点击开始



当用户点击添加之后，会触发js事件，利用js（window.location=url）跳转到指定的页面



进行相应的操作 此时当用户点击表单进行添加的时候如下:



服务器端（控制器）对数据的处理：

<?php

/\*

\* 菜单管理

\*/

namespace Admin\Controller;

use Think\Controller;

class MenuController extends Controller {

function add() {

//收到js的数据进行处理 否则 展示新增模板

if ($\_POST) {

//根据收取到的数据 看是否有值 执行相应的逻辑

if (!isset($\_POST['name']) || !$\_POST['name']) {

return show(0, "菜单名不能为空");

}

if (!isset($\_POST['m']) || !$\_POST['m']) {

return show(0, "模块名不能为空");

}

if (!isset($\_POST['c']) || !$\_POST['c']) {

return show(0, "控制器名不能为空");

}

if (!isset($\_POST['f']) || !$\_POST['f']) {

return show(0, "方法名不能为空");

}

//数据检查无误以后 会递交给model增删处理

$menuId = D('Menu')->insert($\_POST);

if ($menuId){

return show(1,"数据添加成功",$menuId);

}

return show(0,"数据添加失败");

} else {

$this->display();

}

}

function index() {

$this->display();

}

}

控制器会将收集的表单数据通过实例化model对象来讲数据传送到model层 进行处理如下model层处理数据代码：

/\*

\* 登陆模型数据处理

\*/

namespace Common\Model;

use Think\Model;

class MenuModel extends Model {

protected $db = '';

function \_\_construct() {

//M方法是实例化特定的表对象 D方法是实例化自定义的model对象

$this->db = M('menu');

}

/\*

\* 实现菜单的数据添加

\*/

public function insert($data = array()) {

// 如果传送的数据为空或者不是数组 则返回0

if (!$data || !is\_array($data)) {

return 0;

}

return $this->db->add($data);

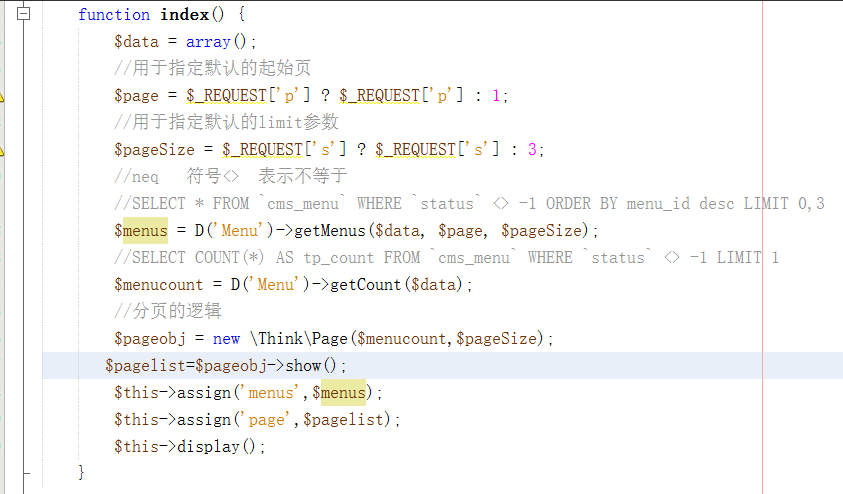
}

}

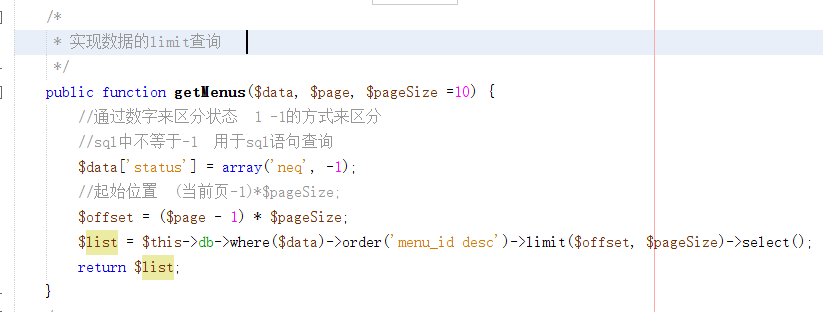
# 菜单数据展示跟分页

原理：从数据库读取数据 传送到控制器层 实例化分页对象 进行分页的操作

对menu控制器的index方法的处理



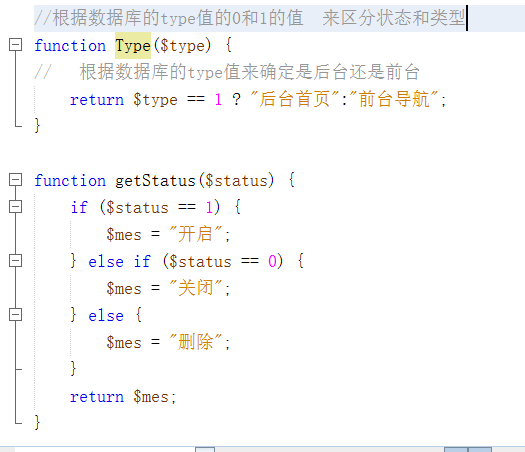
数据的处理：将数据从数据库进行limit读取返回给控制器



前端页面的处理



关于状态码的处理



一、$\_request与$\_post、$\_get的区别和特点  
  
$\_REQUEST[]具用$\_POST[] $\_GET[]的功能,但是$\_REQUEST[]比较慢。通过post和get方法提交的所有数据都可以通过$\_REQUEST数组获得  
$\_post、$\_get的区别和特点  
  
1. get是从服务器上获取数据，post是向服务器传送数据。  
2. get是把参数数据队列加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址。用户看不到这个过程。  
3. 对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。  
4. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。但理论上，IIS4中最大量为80KB，IIS5中为100KB。  
5. get安全性非常低，post安全性较高。  
  
举例：mypage?id=1这种就是GET方式传值，可以用$\_request和$\_get接受传值。

# 搜索功能的实现

根据name=“type”的属性 来实现搜索功能

$\_request获取到相应的值 进而传入到sql中进行相应的查询 将type的值assign到模板中进行相应的默认选中

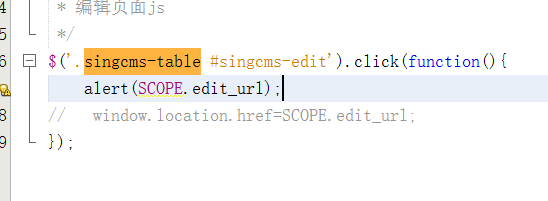


默认选中当前导航

<if condtion=”assign的属性值 eq 1”>selected=”selected”</if>



关于scope的引用 该click事件在哪里触发的 该变量就定义在哪里



# 修改模块的功能

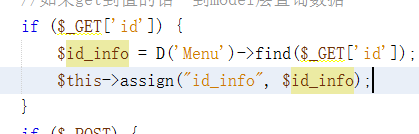
当用户点击编辑模块，跳转到指定的页面

通过该属性 传递当前记录id

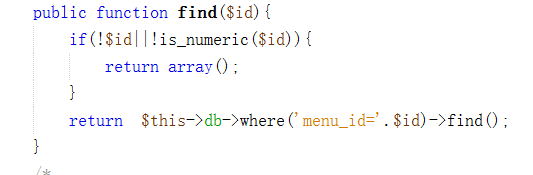




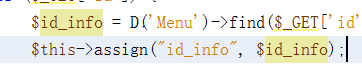
根据用户点击查询相对应的数据并且展示



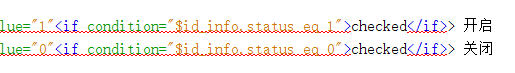
根据ID查询对应的信息



查询以后传到控制器 将数据展示到assign到模板展示



将assign的数据展示 单选的通过查询的数据0,1来区分



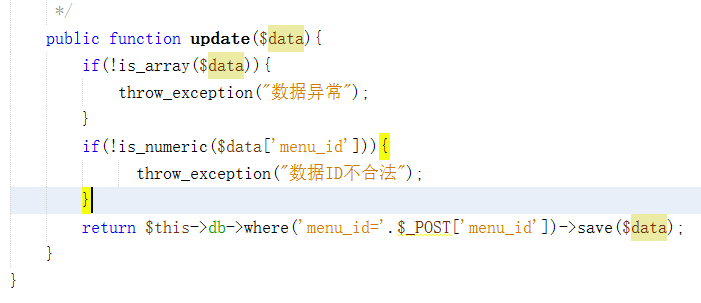
通过js收集表单数据 传递到控制器 进行修改

记住$.post一定要+json

通过$(表单name).serializeArray()；收集表单数据转换为数组 传递到model层



Model层对数据的更新操作



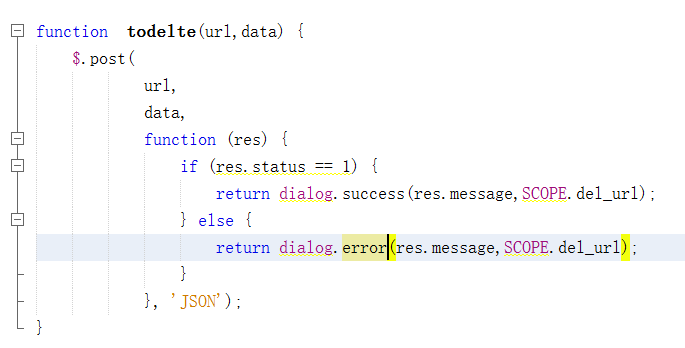
# 删除功能实现

Js通过attr的节点属性获取到当前点击的ID，将ID和status保存为数组 通过ajax发送到指定的位置 进行处理

实现点击弹窗，获取当前的点击ID，消息 保存为数组



Ajax将获取到的数组发送到指定的位置 进行处理 注意当没信息显示可能是type的值没有‘JSON‘



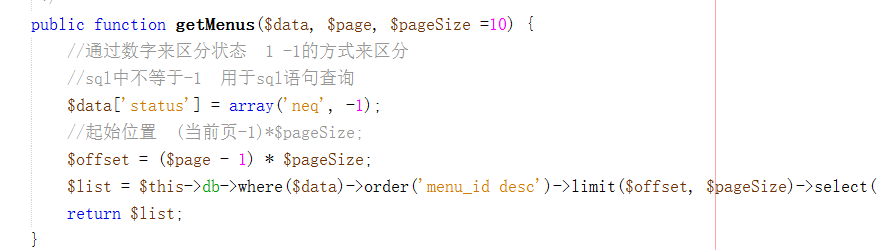
控制器对ajax提交的数据的处理

Try{}catch(exception $e)可以使逻辑更加的严谨

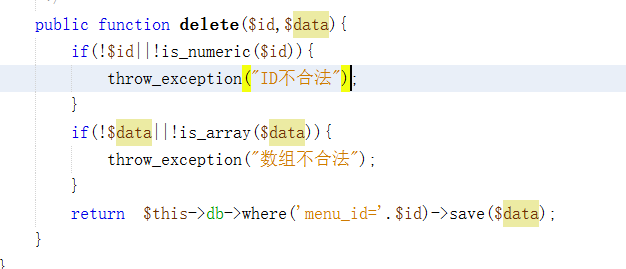


控制器将数据传到model调用model层处理

其实是对status的修改 在数据展示的时候 默认不显示-1的status的数据



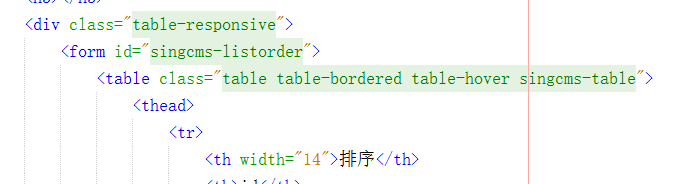
具体的修改操作



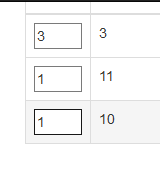
# 排序功能实现

原理：找到menu\_id对应的listorder字段 根据用户输入的值 更新Listorder字段的值 最后根据listorder字段来进行排序

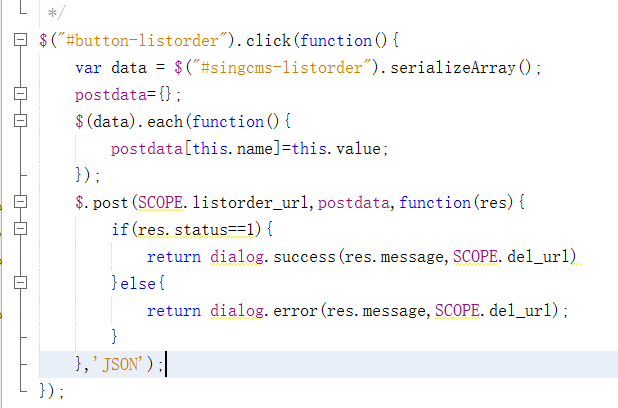
Html页面元素搭建



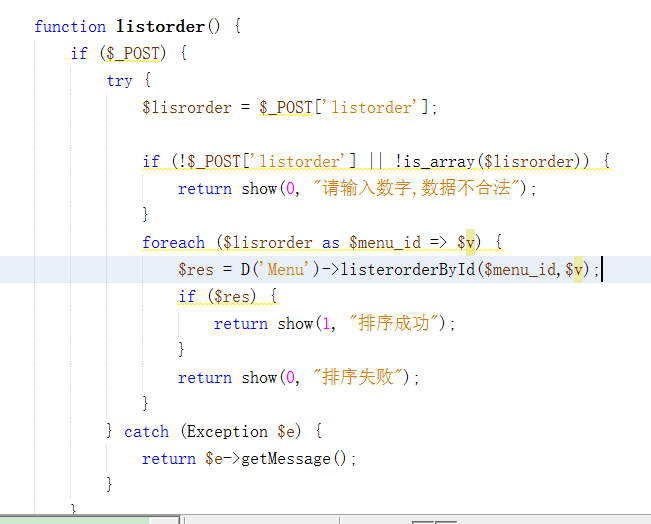
具体的效果



用户点击更新排序以后，触发click事件,收集用户输入的值 和menu\_id的值 通过ajax技术提交到服务器



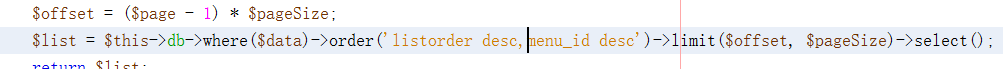
控制器对ajax已提交的数据进行处理通过foreach遍历数组 取到下标值和对应的值 然后进行更改



调用model层去处理用户输入的值 将menu\_id的listorder字段更新为用户输入的值



最后将listorder字段desc查询显示

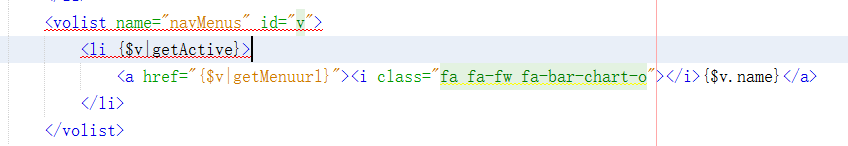


# 左侧导航的显示

从数据库读取指定的数据 特定的条件查询 条件存放到数组中



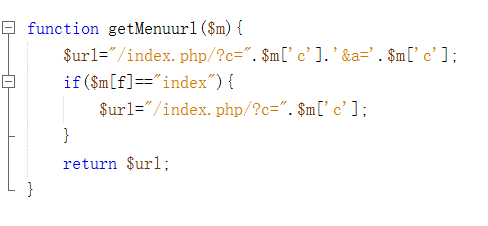
前端页面对数据的处理



点击操作的反应 a链接跳转到控制器 读取到所对应的控制器



根据用户点击从数据库拼接url地址



高亮的处理 获取到当前控制器的名称 与读取的控制器对比 如果一样 则显示高亮

