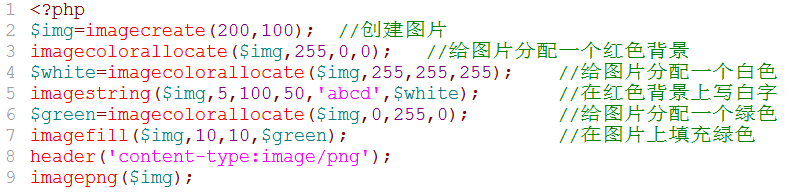


主讲教师：传智.神龙教主

传智播客 PHP学院 <http://php.itcast.cn> 发布

## 回顾

1. 要使用验证码要开启GD库
2. imagecreate(宽度，高度)：创建图片
3. imagecolorallocate(图片资源，红色，绿色，蓝色)
4. imagefill(图片资源，x轴，y轴，颜色)：给图片填充颜色

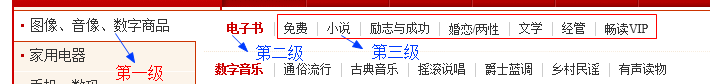


1. imagestring()：在图片上写字符串
2. range()：生成指定范围的数组
3. array\_merge()合并数组
4. array\_rand()：随机抽取元素
5. shuffle()：打乱数组
6. imagesx()：测定图片的宽度
7. imagesy()：测定图片的高度
8. Imagefontwidth()，imagefontheight()：测定字符的宽高
9. Imagegif()、imagepng()、imagejpeg()
10. Imagedestroy()：销毁图片
11. Imagecreatefromgif()、imagecreatefromjpeg()、imagecreatefrompng()

## 多级联动

### 应用

在项目中，二级联动和三级联动用的最多，比如商品类别的分类，省市县等等。



### 分析

以省市县三级联动为例，需要建三个表

省表（province）

|  |  |
| --- | --- |
| 省id | 省名 |
| 1 | 江苏 |
| 2 | 安徽 |

市表（city）

|  |  |  |
| --- | --- | --- |
| 市id | 市名 | 省id |
| 1 | 苏州 | 1 |
| 2 | 安庆 | 2 |
| 3 | 合肥 | 2 |

县表（country）

|  |  |  |
| --- | --- | --- |
| 县id | 县名 | 市id |
| 1 | 昆山 | 1 |

思考：从上面的分析可知，二级联动需要两个表，三级联动需要三个表，这种思路扩展性不强，比如实现无限级联动就麻烦了。那如何实现无限级联动？（问如果通过一个表实现无限级联动？）

|  |  |  |  |
| --- | --- | --- | --- |
| id | name | 排序 | 父级id |
| 1 | 江苏 | 50 | 0 |
| 2 | 安徽 | 40 | 0 |
| 3 | 苏州 | 50 | 1 |
| 4 | 安庆 | 50 | 2 |
| 5 | 昆山 | 50 | 3 |
| 6 | 花桥 | 50 | 5 |

一个表实现无限级联动关键点在于添加一个parentid的字段。

## 实现商品类别的无限级联动

### 创建表

drop table if exists category;

-- 创建商品类别表

create table category(

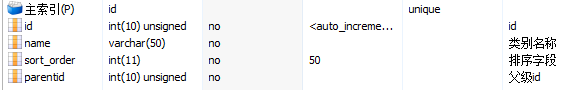
id int unsigned auto\_increment primary key comment 'id',

name varchar(50) not null comment '类别名称',

sort\_order int not null default 50 comment '排序字段',

parentid int unsigned not null comment '父级id'

)engine=innodb charset=utf8;



--插入测试数据

insert into category values (1,'图像、音像、数字商品',default,0);

insert into category values (2,'家用电器',default,0);

insert into category values (3,'手机、数码',default,0);

insert into category values (4,'电子书',default,1);

insert into category values (5,'数字音乐',default,1);

insert into category values (6,'励志与成功',default,4);

insert into category values (7,'文学',default,4);

insert into category values (8,'大家电',default,2);

insert into category values (9,'生活电器',default,2);

### 最简单的二级联动实现

<script type="text/javascript">

//当第一级选项发生变化的时候跳转

function jump() {

location.href='?firstid='+document.getElementById('first').value;

}

</script>

<?php

//连接数据库

mysql\_connect('localhost','root','aa');

mysql\_select\_db('php4');

mysql\_query('set names utf8');

$firstid=isset($\_GET['firstid'])?$\_GET['firstid']:''; //获取提交的firstid

?>

<!--获取第一级-->

<select id='first' onchange='jump()'>

<option value="">---请选择---</option>

<?php

$sql='select \* from category where parentid=0';

$rs=mysql\_query($sql);

while($rows=mysql\_fetch\_assoc($rs)):

?>

<option value="<?php echo $rows['id']?>" <?php echo $rows['id']==$firstid?'selected':''?>><?php echo $rows['name']?></option>

<?php

endwhile;

?>

</select>

<!--获取第二级-->

<?php

$sql="select \* from category where parentid=$firstid";

$rs=mysql\_query($sql);

?>

<select id='second'>

<option value="">---请选择---</option>

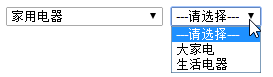
<?php while($rows=mysql\_fetch\_assoc($rs)): ?>

<option value="<?php echo $rows['id']?>"><?php echo $rows['name']?></option>

<?php endwhile; ?>

</select>

//结果



## 二级联动的数据一次循环取出

<?php

//连接数据库

mysql\_connect('localhost','root','aa');

mysql\_query('use php4');

mysql\_query('set names utf8');

//获取数据

$rs=mysql\_query('select \* from category order by sort\_order');

//将资源转成二维数组

$array=array();

while($rows=mysql\_fetch\_assoc($rs)) {

$array[]=$rows;

}

//循环二维数组

echo '<ul>';

foreach($array as $rows1) {

if($rows1['parentid']==0) {

echo "<li>{$rows1['name']}</li>";

echo '<ul>';

foreach($array as $rows2) {

if($rows2['parentid']==$rows1['id']){

echo "<li>{$rows2['name']}</li>";

}

}

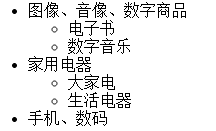
echo '</ul>';

}

}

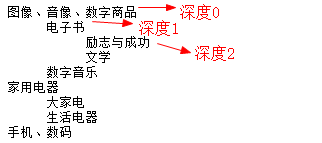
echo '</ul>';

?>

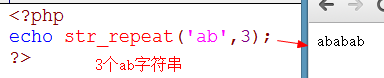


## 获取无限级联动的数据

无限级使用递归来实现，给每个记录添加一个”深度”字段



### str\_repeat(字符串，重复次数)



### 完整代码

<?php

//连接数据库

mysql\_connect('localhost','root','aa');

mysql\_select\_db('php4');

mysql\_query('set names utf8');

//获取数据

$sql='select \* from category order by sort\_order';

$rs=mysql\_query($sql);

//将资源转成二维数组

$array=array();

while($rows=mysql\_fetch\_assoc($rs)) {

$array[]=$rows;

}

//创建树型结构

function createTree($list,$parentid=0,$deep=0) {

static $tree=array();

foreach($list as $rows) {

if($rows['parentid']==$parentid){

$rows['deep']=$deep; //给每个记录添加一个“深度”

$tree[]=$rows;

createTree($list,$rows['id'],$deep+1);

}

}

return $tree;

}

//测试

echo '<pre>';

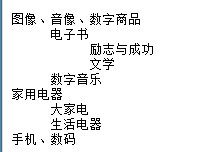
$temp=createTree($array);

foreach($temp as $rows) {

echo str\_repeat('&nbsp;',$rows['deep']\*3),$rows['name'],'<br>';

}

?>



## 封装万能的insert语句

insert语句中变化的是字段名和值，字段名作为关联数组的键，值作为关联数组的值。

$table='category';

$data['name']='电视';

$data['sort\_order']=30;

$data['parentid']=8;

$fields=array\_keys($data); //获取所有的键，

$values=array\_values($data); //获取所有值

//将fields数组中的每个元素添加上反引号

$fields=array\_map(function($field){

return "`{$field}`";

},$fields);

$fields\_str=implode(',',$fields); //将字段用逗号分隔

//将values数组中的值每个元素都加上引号

$values=array\_map(function($value){

return "'{$value}'";

},$values);

$values\_str=implode(',',$values);

//拼接字符串

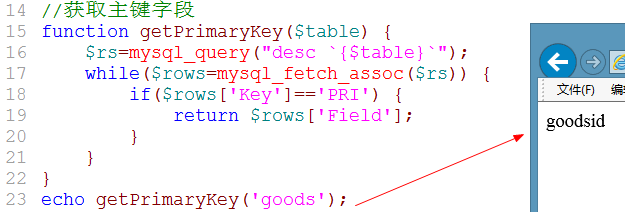
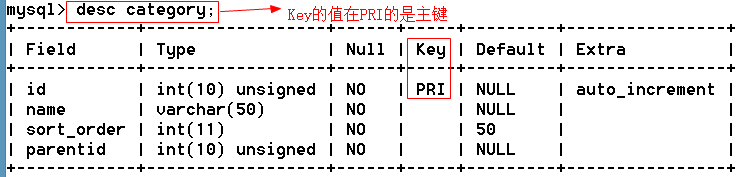
$sql="insert into `{$table}` ({$fields\_str}) values ($values\_str)";

echo $sql;



## 封装一个万能的update语句

### 难点：判断哪个字段是主键



注意事项：匿名函数内部要使用外部的数据可以通过use将外部数据引入到匿名函数内部

header('content-type:text/html;charset=utf-8');

mysql\_connect('localhost','root','aa');

mysql\_query('set names utf8');

mysql\_query('use php4');

$table='category';

$data['id']=10;

$data['name']='电视';

$data['sort\_order']=30;

$data['parentid']=8;

//获取字段

$fields=array\_keys($data);

//获取主键字段

function getPrimaryKey($table) {

$rs=mysql\_query("desc `{$table}`");

while($rows=mysql\_fetch\_assoc($rs)) {

if($rows['Key']=='PRI') {

return $rows['Field'];

}

}

}

$pk=getPrimaryKey($table);

//去除主键

$index=array\_search($pk,$fields); //主键的下标

unset($fields[$index]); //销毁主键

//遍历循环字段

$fields=array\_map(function($field) use ($data){

return "`{$field}`='{$data[$field]}'";

},$fields);

$fields=implode(',',$fields);

$sql="update `{$table}` set {$fields} where `{$pk}`=$data[$pk]";

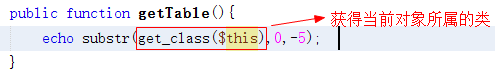
echo $sql;



## 将数据操作语句封装到项目中

### 难点：在Model类中获取当前的表名

因为模型型就是表名，只要获取模型的名就可以知道表名



### 代码

abstract class Model {

protected $db; //用来保存MySQLDB的单例

/\*\*

\*构造函数初始化连接数据库

\*/

public function \_\_construct() {

$this->initDB();

}

//获取MySQLDBm类的实例

private function initDB() {

$this->db=MySQLDB::getInstance($GLOBALS['config']['database']);

}

//获取当前操作的表名

private function getTable(){

return substr(get\_class($this),0,-5);

}

//获取主键字段

private function getPrimaryKey($table) {

$rs= $this->db->query("desc `{$table}`");

while($rows=mysql\_fetch\_assoc($rs)) {

if($rows['Key']=='PRI') {

return $rows['Field'];

}

}

}

/\*

\* insert，封装一个万能的insert语句

\* @param $data array 有字段名和值组成的关联数组

\*/

public function insert($data){

$fields=array\_keys($data); //获取所有的键，

$values=array\_values($data); //获取所有值

//将fields数组中的每个元素添加上反引号

$fields=array\_map(function($field){

return "`{$field}`";

},$fields);

$fields\_str=implode(',',$fields); //将字段用逗号分隔

//将values数组中的值每个元素都加上引号

$values=array\_map(function($value){

return "'{$value}'";

},$values);

$values\_str=implode(',',$values);

//拼接字符串

$table=$this->getTable();

$sql="insert into `{$table}` ({$fields\_str}) values ($values\_str)";

return $this->db->query($sql);

}

/\*

\* update,万能的更新方法

\* @param $data array 关联数组

\*/

public function update($data){

//获取字段

$fields=array\_keys($data);

$table=$this->getTable(); //获取当前表

$pk=$this->getPrimaryKey($table); //获取表的主键

//去除主键

$index=array\_search($pk,$fields); //主键的下标

unset($fields[$index]); //销毁主键

//遍历循环字段

$fields=array\_map(function($field) use ($data){

return "`{$field}`='{$data[$field]}'";

},$fields);

$fields=implode(',',$fields);

$sql="update `{$table}` set {$fields} where `{$pk}`=$data[$pk]";

return $this->db->query($sql);

}

/\*

\* 封装一个万能的删除方法

\* @param $id mixed 主键字段的值

\*/

public function del($id){

$table= $this->getTable();

$pk= $this->getPrimaryKey($table);

$sql="delete from `{$table}` where `{$pk}`=$id";

return $this->db->query($sql);

}

}

## 在Model类中封装数据查询语句

### 返回所有数据（二维数组）

/\*

\* 封装一个万能的查询语句

\* @param $field string 排序字段

\* @param $order string 排序方法

\*/

public function select($field='',$order='asc'){

$table= $this->getTable();

$sql="select \* from `{$table}`";

if($field!=''){

$sql.=" order by `{$field}` {$order}";

}

return $this->db->fetchAll($sql);

}

### 通过主键返回一维数组

/\*

\* 封装一个万能的查询语句，通过主键的值获取记录

\* @param $id mixid 主键的值

\* @return array 一维数组

\*/

public function find($id){

$table= $this->getTable();

$pk= $this->getPrimaryKey($table);

$sql="select \* from `{$table}` where `{$pk}`='{$id}'";

return $this->db->fetchRow($sql);

}

## 在项目中实现商品类别管理

### 将商品类别列表和添加商品类别界面联通起来

将商品分类列表(category\_list.html)和添加商品分类(category\_add.html)联通起来

控制器

class CategoryController extends Controller{

public function listAction(){

require \_\_VIEW\_\_.'category\_list.html';

}

public function addAction(){

require \_\_VIEW\_\_.'category\_add.html';

}

}

视图

menu.html



category\_list.html



category\_add.html



### 显示商品类别列表

模型

<?php

class CategoryModel extends Model{

/\*

\* 创建树型结构

\*/

private function createTree($list,$parentid=0,$deep=0) {

static $tree=array();

foreach($list as $rows) {

if($rows['parentid']==$parentid){

$rows['deep']=$deep;

$tree[]=$rows;

$this->createTree($list,$rows['id'],$deep+1);

}

}

return $tree;

}

/\*

\* 获得商品类别的树型结构

\*/

public function getCategoryTree(){

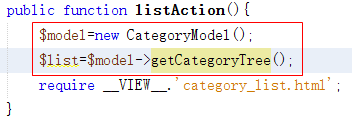
$list= $this->select();

return $this->createTree($list);

}

}

控制器



视图

<?php foreach($list as $rows):?>

<tr align="center" class="0" id="0\_6">

<td align="left" class="first-cell" style="background-color: rgb(255, 255, 255); padding-left:<?php echo $rows['deep']\*30?>px">

<img src="/Public/Admin/images/menu\_minus.gif" id="icon\_0\_6" width="9" height="9" border="0" style="margin-left:0em" onClick="rowClicked(this)">

<span><?php echo $rows['name']?></span>

</td>

<td width="10%" align="right"><?php echo $rows['sort\_order']?></td>

<td width="24%" align="center" >

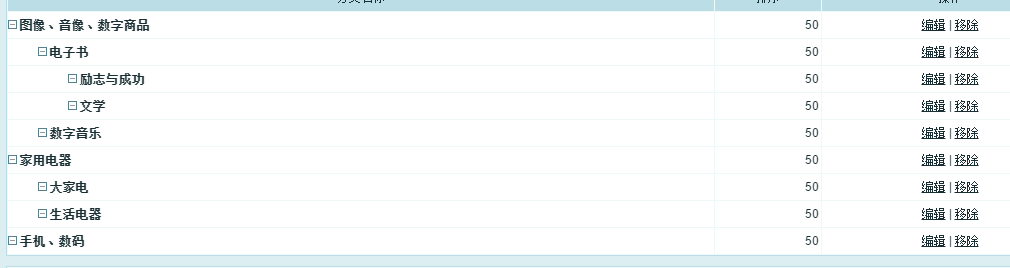
<a href="category.php?act=edit&amp;cat\_id=6">编辑</a> |

<a href="javascript:;" onClick="listTable.remove(6, '您确认要删除这条记录吗?')" title="移除">移除</a>

</td>

</tr>

<?php endforeach;?>



### 添加商品类别

模型

无

控制器

public function addAction(){

$model=new CategoryModel();

if(!empty($\_POST)){ //执行添加逻辑

$data['name']=$\_POST['cat\_name'];

$data['parentid']=$\_POST['parent\_id'];

$data['sort\_order']=$\_POST['sort\_order'];

if($model->insert($data))

$this->success ('index.php?p=Admin&c=Category&a=list', '添加成功',1);

else

$this->error ('index.php?p=Admin&c=Category&a=add', '添加失败');

exit;

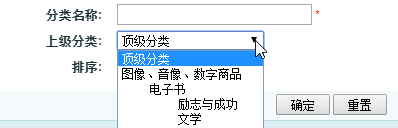
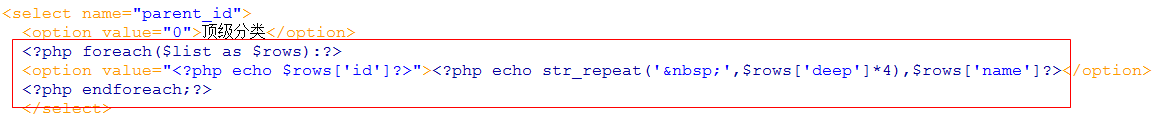
}

$list=$model->getCategoryTree();

require \_\_VIEW\_\_.'category\_add.html';

}

视图



### 更新商品类别

难点：修改的时候要判断

1. 自己不能是自己的子级
2. 指定的父级不能是自己的后代

模型

没有

控制器

public function updateAction(){

$id=(int)$\_GET['id'];

$model=new CategoryModel();

if(!empty($\_POST)){

$data['name']=$\_POST['cat\_name'];

$data['parentid']=$\_POST['parent\_id'];

$data['sort\_order']=$\_POST['sort\_order'];

$data['id']=$id;

//自己不能是自己的子级

if($id==$data['parentid']){

$this->error('index.php?p=Admin&c=Category&a=list', '自己不能是自己的子级', 3);

exit;

}

//指定的父级不能是自己的后代

$sublist=$model->getCategoryTree($id); //当前节点下的所有子元素

foreach($sublist as $rows){

if($rows['id']==$data['parentid']){

$this->error('index.php?p=Admin&c=Category&a=list', '指定的父级是自己的后代');

exit;

}

}

if($model->update($data))

$this->success ('index.php?p=Admin&c=Category&a=list', '修改成功',1);

else

$this->error ('index.php?p=Admin&c=Category&a=add', '修改失败');

exit;

}

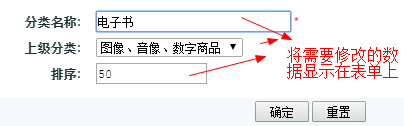
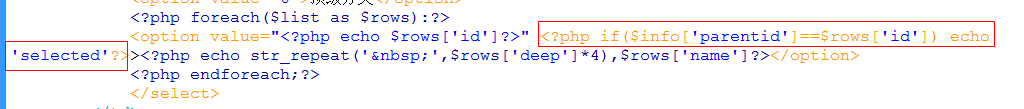
$info=$model->find($id); //需要修改的商品类别信息

$list=$model->getCategoryTree(); //商品类别树型结构

require \_\_VIEW\_\_.'category\_update.html';

}

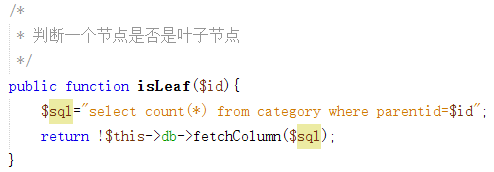
视图



### 删除商品类别

难点：删除的时候要判断是否是叶子节点

CategoryModel模型



控制器

/\*

\* 删除商品类别

\*/

public function delAction(){

$id=(int)$\_GET['id'];

$model=new CategoryModel();

if($model->isLeaf($id)){

if($model->del($id))

$this->success ('index.php?p=Admin&c=Category&a=list', '删除成功',1);

else

$this->error ('index.php?p=Admin&c=Category&a=list', '删除失败');

}

else

$this->error ('index.php?p=Admin&c=Category&a=list', '此节点下有节点，不允许删除');

}

视图

