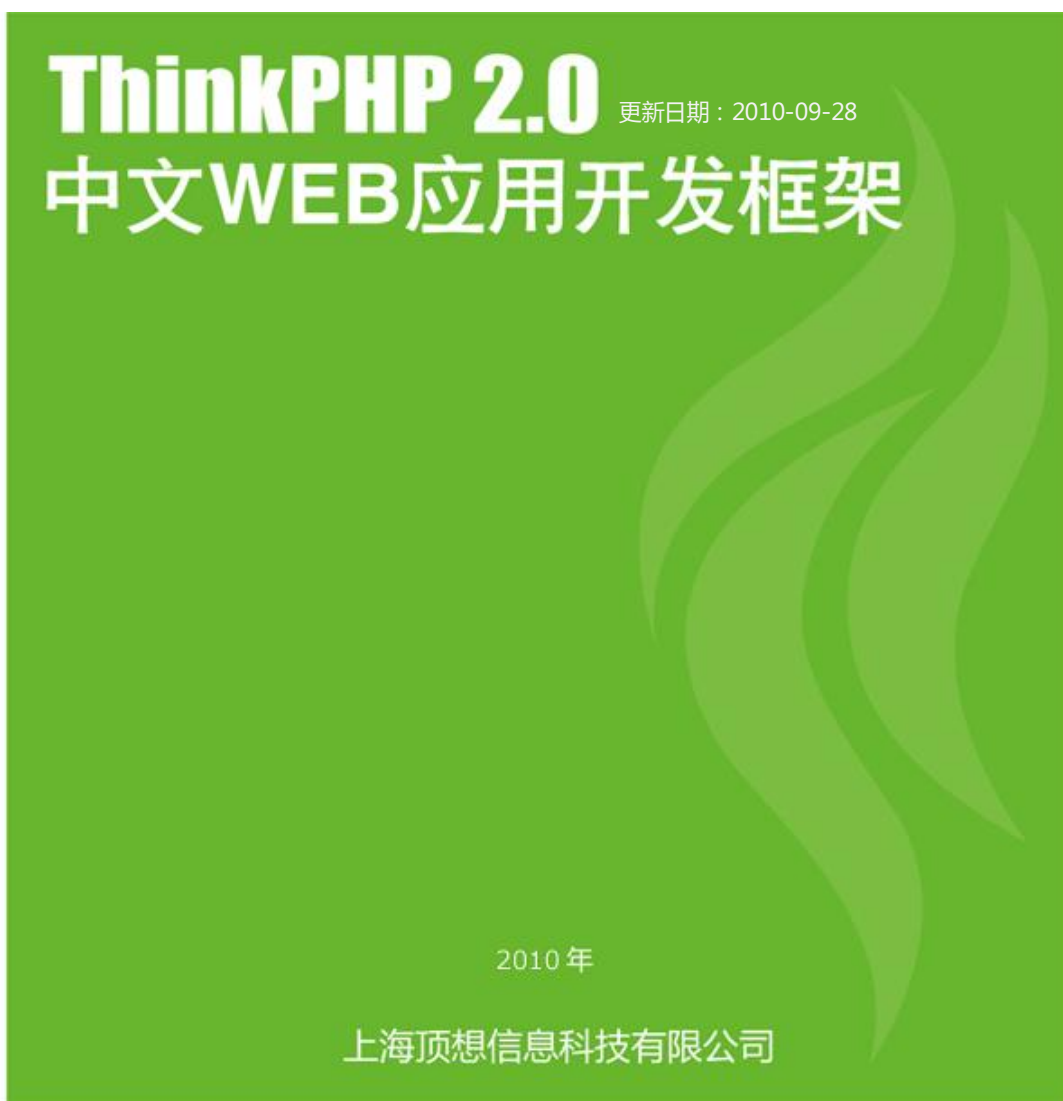




# RBAC 类使用简析



## 目录

1	RBAC 类使用简析	3
1.1	数据表 .....	3
1.2	几个重要文件 .....	6

## 1 RBAC 类使用简析

### 1.1 数据表

#### 用到的数据表

`think_access`	角色访问权限表
`think_node`	系统节点表
`think_role`	角色分组表
`think_role_user`	用户角色关系表
`think_user`	后台账号（用户或者管理员）表

#### 数据表结构

```
CREATE TABLE IF NOT EXISTS `think_access` (  
  `role_id` smallint(6) unsigned NOT NULL,  
  `node_id` smallint(6) unsigned NOT NULL,  
  `level` tinyint(1) NOT NULL,  
  `module` varchar(50) DEFAULT NULL,  
  KEY `groupId` (`role_id`),  
  KEY `nodeId` (`node_id`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
CREATE TABLE IF NOT EXISTS `think_node` (  
  `id` smallint(6) unsigned NOT NULL AUTO_INCREMENT,  
  `name` varchar(20) NOT NULL,  
  `title` varchar(50) DEFAULT NULL,  
  `status` tinyint(1) DEFAULT '0',  
  `remark` varchar(255) DEFAULT NULL,  
  `sort` smallint(6) unsigned DEFAULT NULL,
```

```
`pid` smallint(6) unsigned NOT NULL,  
`level` tinyint(1) unsigned NOT NULL,  
PRIMARY KEY (`id`),  
KEY `level` (`level`),  
KEY `pid` (`pid`),  
KEY `status` (`status`),  
KEY `name` (`name`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
CREATE TABLE IF NOT EXISTS `think_role` (  
  `id` smallint(6) unsigned NOT NULL AUTO_INCREMENT,  
  `name` varchar(20) NOT NULL,  
  `pid` smallint(6) DEFAULT NULL,  
  `status` tinyint(1) unsigned DEFAULT NULL,  
  `remark` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `pid` (`pid`),  
  KEY `status` (`status`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 ;
```

```
CREATE TABLE IF NOT EXISTS `think_role_user` (  
  `role_id` mediumint(9) unsigned DEFAULT NULL,  
  `user_id` char(32) DEFAULT NULL,  
  KEY `group_id` (`role_id`),  
  KEY `user_id` (`user_id`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `think_user` (  
  `id` smallint(5) unsigned NOT NULL auto_increment,  
  `account` varchar(64) NOT NULL,  
  `nickname` varchar(50) NOT NULL,  
  `password` char(32) NOT NULL,  
  `bind_account` varchar(50) NOT NULL,  
  `last_login_time` int(11) unsigned default '0',  
  `last_login_ip` varchar(40) default NULL,  
  `login_count` mediumint(8) unsigned default '0',
```

```
`verify` varchar(32) default NULL,  
`email` varchar(50) NOT NULL,  
`remark` varchar(255) NOT NULL,  
`create_time` int(11) unsigned NOT NULL,  
`update_time` int(11) unsigned NOT NULL,  
`status` tinyint(1) default '0',  
`type_id` tinyint(2) unsigned default '0',  
`info` text NOT NULL,  
PRIMARY KEY (`id`),  
UNIQUE KEY `account` (`account`)  
) ENGINE=MyISAM AUTO_INCREMENT=35 DEFAULT CHARSET=utf8
```

## 1.2 几个重要文件

### Conf/config.php

Rbac 示例默认配置如图 1:

```

5  'USER_AUTH_ON'           =>true,           // 开启认证
6  'USER_AUTH_TYPE'        =>1,              // 默认认证
7  'USER_AUTH_KEY'         =>'authId',        // 用户认证
8  'ADMIN_AUTH_KEY'        =>'administrator',
9  'USER_AUTH_MODEL'        =>'User',          // 默认验证
10 'AUTH_PWD_ENCODER'       =>'md5',           // 用户认证
11 'USER_AUTH_GATEWAY'      =>'/Public/login', // 默认认证
12 'NOT_AUTH_MODULE'       =>'Public',         // 默认无需
13 'REQUIRE_AUTH_MODULE'  =>'',              // 默认需要
14 'NOT_AUTH_ACTION'       =>'',              // 默认无需
15 'REQUIRE_AUTH_ACTION'  =>'',              // 默认需要
16 'GUEST_AUTH_ON'         =>false,           // 是否开启
17 'GUEST_AUTH_ID'         =>0,               // 游客的用
18 'SHOW_RUN_TIME'         =>true,            // 运行时间
19 'SHOW_ADV_TIME'         =>true,            // 显示详细
20 'SHOW_DB_TIMES'         =>true,            // 显示数据
21 'SHOW_CACHE_TIMES'     =>true,            // 显示缓存
22 'SHOW_USE_MEM'          =>true,            // 显示内存
23 'DB_LIKE_FIELDS'        =>'title|remark',
24 'RBAC_ROLE_TABLE'       =>'think_role',    // 角色表表
25 'RBAC_USER_TABLE'       =>'think_role_user',// 用户角色
26 'RBAC_ACCESS_TABLE'    =>'think_access',   // 角色访问
27 'RBAC_NODE_TABLE'      =>'think_node',     // 系统节点

```

图 1

### Lib/PublicAction.class.php

图 1 第 7 行：配置了认证 SESSION 标记为 authId，这个值随便配置，只要与系统中现有的\$\_SESSION 不冲突便可；即图 2 代码\$\_SESSION['USER\_AUTH\_KEY'] 相当于\$\_SESSION['authId'];

图 1 第 11 行：‘USER\_AUTH\_GATEWAY’=>’/Public/login’ // 也就是用户登录后台的一道门，若已经登录，则会直接跳转到后台主页面；相反，没有设置\$\_SESSION[‘authId’]这个值，或者已经过期，将会显示登录页面；具体代码如图 2；

图 1 第 12 行：‘NOT\_AUTH\_MODULE’=>’Public’ // 默认不需要进行认证的模块，也就是没有登录，也可以访问该模块以及该模块下的方法；这里默认是 Public 模块，多个模块的话可以用半角逗号进行分隔；如 A,B,C；所以登陆页面 login()方法和登录检测 checkLogin()方法，都写在 PublicAction 类中；

```
// 用户登录页面
public function login() {
    if(!isset($_SESSION[C('USER_AUTH_KEY')])) {
        $this->display();
    }else{
        $this->redirect('Index/index');
    }
}
```

图 2

在方法 checkLogin()中，我们需要将 RBAC 类库通过 import()方法来引入（图 3 第 144 行），这里我们把类库拷贝到项目的 Lib/ORG 目录下，所以使用 import('@.ORG.RBAC')，import 方法的正确使用，请参阅帮助文档相关章节；

```

136      //生成认证条件
137      $map      =      array();
138      // 支持使用绑定帐号登录
139      $map['account'] = $_POST['account'];
140      $map["status"] = array('gt',0);
141  曰      if($_SESSION['verify'] != md5($_POST['verify'])) {
142          $this->error('验证码错误! ');
143      }
144      import ( '@.ORG.RBAC' );
145      $authInfo = RBAC::authenticate($map);
146      //使用用户名、密码和状态的方式进行认证

```

图 3

图 3 第 145 行，调用了 RBAC::authenticate()方法，传入\$map 数组，通过 authenticate 方法，来检测后台账号表中是否有该用户，并且账号状态是否正常；C('USER\_AUTH\_MODEL')我们在图 1 第九行配置为 User；对应的后台账号表就是`think\_user`表；该方法会返回该账号的相关信息，赋值到\$authInfo 中；如果有该用户，通过密码的比对，即完成了一个简单的登录认证，如图 5 第 150、151 行；

```

76  class RBAC extends Think
77  曰 {
78      // 认证方法
79      static public function authenticate($map,$model='')
80  曰 {
81          if(empty($model)) $model = C('USER_AUTH_MODEL');
82          //使用给定的Map进行认证
83          return M($model)->where($map)->find();

```

图 4

图 5 第 153 行：将登录后的用户 ID 记录在\$\_SESSION['authId']中；

图 5 第 158 行：如果用户名是超级管理员“admin”，则会设置\$\_SESSION['administrator']为 true；在 RBAC 类中对于该用户，是拥所有的权限的；图 5 第 158 行：建议写成



`$_SESSION[C('ADMIN_AUTH_KEY')] = true` 因为我们在配置文件图 1 第 8 行有配置  
`ADMIN_AUTH_KEY => 'administrator' ;`

```

146      //使用用户名、密码和状态的方式进行认证
147      if(False == $authInfo) {
148          $this->error('帐号不存在或已禁用! ');
149      }else {
150          if($authInfo['password'] != md5($_POST['password'])) {
151              $this->error('密码错误! ');
152          }
153          $_SESSION[C('USER_AUTH_KEY')] = $authInfo['id'];
154          $_SESSION['email'] = $authInfo['email'];
155          $_SESSION['loginUserName'] = $authInfo['nickname'];
156          $_SESSION['lastLoginTime'] = $authInfo['last_login_time'];
157          $_SESSION['login_count'] = $authInfo['login_count'];
158          if($authInfo['account']=='admin') {
159              $_SESSION['administrator'] = true;
160          }

```

图 5

图 6 第 173 行：在 `checkLogin()` 方法最后，我们调用类 `RBAC::saveAccessList()` 方法，把该用户所拥有的各个角色的节点访问权限保存在 SESSION 中；

```

171
172      // 缓存访问权限
173      RBAC::saveAccessList();
174      $this->success('登录成功! ');

```

图 6

图 7 第 92 行：图 1 第六行，我们配置了 `USER_AUTH_TYP` 的值为 1，所以在这里，如果我们的账号不是超级管理员 `admin`，则会继续执行类中的方法 `getAccessList()`，将返回值（一个数组）设置在 `$_SESSION['_ACCESS_LIST']` 中；。

```

86      //用于检测用户权限的方法,并保存到Session中
87      static function saveAccessList($authId=null)
88      {
89          if(null==$authId)    $authId = $_SESSION[C('USER_AUTH_KEY')];
90          // 如果使用普通权限模式，保存当前用户的访问权限列表
91          // 对管理员开发所有权限
92      92      if(C('USER_AUTH_TYPE') !=2 && !$_SESSION[C('ADMIN_AUTH_KEY')]) )
93      93      $_SESSION['_ACCESS_LIST']  =  RBAC::getAccessList($authId);
94          return ;|
95      }

```

图 7

图 8 第 212 行：getAccessList()方法根据用户的认证 authId，来查找与之关联的角色，角色所拥有的节点访问权限列表；这里的

C('RBAC\_ROLE\_TABLE')          对应于图 1 第 24 行；

C('RBAC\_USER\_TABLE')          对应于图 1 第 25 行；

C('RBAC\_ACCESS\_TABLE')        对应于图 1 第 26 行；

C('RBAC\_NODE\_TABLE')          对应于图 1 第 27 行；

所以说这几个值在项目配置文件中也要设置正确了，否则 RBAC 会运行不正常；

```

201      * 取得当前认证号的所有权限列表
202      +-----+
203      * @param integer $authId 用户ID
204      +-----+
205      * @access public
206      +-----+
207      */
208      static public function getAccessList($authId)
209      {
210          // Db方式权限数据
211          $db      =  Db::getInstance();
212      212      $table =
                array('role'=>C('RBAC_ROLE_TABLE'),'user'=>C('RBAC_USER_TABLE'),
                'access'=>C('RBAC_ACCESS_TABLE'),'node'=>C('RBAC_NODE_TABLE'));
213      $sql      =  "select node.id,node.name from "

```

图 8

到这里，假设我们已经通过了认证网关的认证，已经成功登录，此时我们要跳转到

Index 模块，而此时我们打开 Lib/IndexAction.class.php，发现 IndexAction 类是继承了类 CommonAction，如图 9；而一些 RBAC 的权限检测也是放在这个类里面；以后新增一个功能模块，只要是需要认证的，都继承这个公共类，即在 RBAC 的控制范围之类了。

```
1 <?php
2 class IndexAction extends CommonAction {
3     // 增加方法
```

图 9

### CommonAction.class.php

所有的权限认证，放在初始化方法\_initialize()中,如图 10；这样只要是继承该类的子类（模块），执行任何操作，都是要经过这里来检测；一些公共的操作的权限控制，我们可以把共用的一些方法写在此类中，比如：

```
public function add()      新增页面显示

public function insert()  写入的权限

public function delete()  删除的权限

.....
```

```

2  class CommonAction extends Action {
3  曰 function _initialize() {
4      // 用户权限检查
5  曰  if (C ( 'USER_AUTH_ON' ) &&
6      !in_array(MODULE_NAME,explode(',',$C('NOT_AUTH_MODULE')))) {
7      import ( '@.ORG.RBAC' );
8  曰  if (! RBAC::AccessDecision ()) {
9      //检查认证识别号
10  曰  if (! $_SESSION [C ( 'USER_AUTH_KEY' )]) {
11      //跳转到认证网关
12      redirect ( PHP_FILE . C ( 'USER_AUTH_GATEWAY' ) );
13  }
14  // 没有权限 抛出错误
15  曰  if (C ( 'RBAC_ERROR_PAGE' )) {
16      // 定义权限错误页面
17      redirect ( C ( 'RBAC_ERROR_PAGE' ) );
18  曰  } else {
19  曰  if (C ( 'GUEST_AUTH_ON' )) {
20      $this->assign ( 'jumpUrl', PHP_FILE . C ( 'USER_AUTH_GATEWAY' ) );
21  }
22  // 提示错误信息
23  $this->error ( L ( '_VALID_ACCESS_' ) );
24  }

```

图 10

图 10 第 5 行：我们再图 1 第 5 行，配置 USER\_AUTH\_ON 为 true，即我们是启用了 RBAC 的验证；

图 10 第 6 行：这里的 NOT\_AUTH\_MODULE 我们在图 1 第 12 行，默认配置只有 Public 模块，则表示其他模块，只要是继承于 CommonAction 类，则都需要进行 RBAC 验证；

图 10 第 7 行：和上面提到的一样，因为 RBAC 已分离到核心之外，现在使用的时候，我们先要进行 import()方法来引入到当前类中即可。

图 10 第 8 行：调用 RBAC::AccessDecision()方法，如图 11；有权限则返回 true,其下代码则不用执行；无权限则返回 false，继续下面的代码，根据项目配置，来给出错误提示；

图 10 第 10 行：没有登录或者超时退出了登录，则跳转到默认认证网关  
USER\_AUTH\_GATEWAY;

图 10 第 15 行，正常登录得账号，没有权限，则跳转到 RBAC\_ERROR\_PAGE 所配置的错误提示页面；若没有该配置，则直接给出没有权限错误提示页面；

```

162 //权限认证的过滤器方法
163 static public function AccessDecision($appName=APP_NAME)
164 {
165     //检查是否需要认证
166     if(RBAC::checkAccess()) {
167         //存在认证识别号，则进行进一步的访问决策
168         $accessGuid = md5($appName.MODULE_NAME.ACTION_NAME);
169         if(empty($_SESSION[C('ADMIN_AUTH_KEY')])) {
170             if(C('USER_AUTH_TYPE')==2) {
171                 //加强验证和即时验证模式 更加安全 后台权限修改可以即时生效
172                 //通过数据库进行访问检查
173                 $accessList = RBAC::getAccessList($_SESSION[C('USER_AUTH_KEY')]
174             }else {
175                 // 如果是管理员或者当前操作已经认证过，无需再次认证
176                 if( $_SESSION[$accessGuid]) {
177                     return true;
178                 }
179                 //登录验证模式，比较登录后保存的权限访问列表
180                 $accessList = $_SESSION['_ACCESS_LIST'];
181             }
182             //判断是否为组件化模式，如果是，验证其全模块名
183             $module = defined('P_MODULE_NAME')? P_MODULE_NAME : MODULE_NAME;
184             if(!isset($accessList[strtoupper($appName)][strtoupper($module)][strtoupper($accessGuid)])
185                 $_SESSION[$accessGuid] = false;
186             return false;
187         }
188         else {
189             $_SESSION[$accessGuid] = true;
190         }
191     }else{
192         //管理员无需认证
193         return true;

```

图 11

图 11 第 166 行：调用 RBAC::checkAccess()方法，如图 12；总的来说功能就是检测当前模块，当前操作是否需要进行权限认证；会根据图 1 的配置来检测，并且如果设置了

REQUIRE\_AUTH\_MODULE, 则会忽略 NOT\_AUTH\_MODULE 这个设置; 操作列表也类似;

第 5 行: 'USER\_AUTH\_ON' => true,

第 12 行: 'NOT\_AUTH\_MODULE' => 'Public',

第 14 行: 'NOT\_AUTH\_ACTION' => "",

第 13 行: 'REQUIRE\_AUTH\_MODULE' => "",

第 15 行: 'REQUIRE\_AUTH\_ACTION' => "",

图 11 第 178 行: 若配置 USER\_AUTH\_TYPE 为 2, 即实时认证, 则每次的权限访问列表都是通过 RBAC::getAccessList()方法来直接读取数据库; 我们的配置默认为 1, 如图 1 第 6 行; 所以如果修改了某个角色的权限, 则该用户下次登录时才具有该权限, 因为是保存 SESSION 中的; 最后根据\$\_SESSION[\$accessGuid]的值来返回 true 或者 false; 当然如果你是超级管理员, 则直接返回 true;

```

106 //检查当前操作是否需要认证
107 static function checkAccess()
108 {
109     //如果项目要求认证，并且当前模块需要认证，则进行权限认证
110     if( C('USER_AUTH_ON') ){
111         $_module = array();
112         $_action = array();
113         if('' != C('REQUIRE_AUTH_MODULE')) {
114             //需要认证的模块
115             $_module['yes'] = explode(',',strtoupper(C('REQUIRE_AUTH_MODULE')));
116         }else {
117             //无需认证的模块
118             $_module['no'] = explode(',',strtoupper(C('NOT_AUTH_MODULE')));
119         }
120         //检查当前模块是否需要认证
121         if((!empty($_module['no']) && !in_array(strtoupper(MODULE_NAME),$_module['in_array(strtoupper(MODULE_NAME),$_module['yes'])')) {
122             if('' != C('REQUIRE_AUTH_ACTION')) {
123                 //需要认证的操作
124                 $_action['yes'] = explode(',',strtoupper(C('REQUIRE_AUTH_ACTION'))
125             }else {
126                 //无需认证的操作
127                 $_action['no'] = explode(',',strtoupper(C('NOT_AUTH_ACTION')));
128             }

```

图 12

到这里 RBAC 大致的执行过程都的分析过了；当然正确的后台操作，包括角色的分组，设置后台账号的拥有的角色，项目、模块、操作节点的添加；最后对角色的正确授权也是很重要；具体的操作演示会有一个 RBAC 操作演示文档，大家也可以先对着演示文档，实际操作几遍；还是会加深对 RBAC 的理解；

ThinkPHP 文档小组 2010-9-28

[HTTP://ThinkPHP.CN](http://ThinkPHP.CN) | WEB 应用开发最佳实践框架

大道至简，开发由我 WE CAN DO IT, JUST THINK