**自动登陆**

1. class LoginController extends FrontendController{
2. /\* 会员登录验证 \*/
3. public function check(){
4. // ...
5. //登录成功后
6. $this->remember(); //两周内记住账号
7. // ...
8. }
9. /\* 两周内记住账号 \*/
10. private function remember(){
11. $remember\_me = I('remember\_me', null, 'int');
12. if (!$remember\_me) {
13. return false;
14. }
15. $mobile = I('post.mobile', '', '/^1[34578]\d{9}$/');
16. $pwd = I('post.pwd', '');
17. if (empty($mobile) || empty($pwd)) {
18. return false;
19. }
20. $pwd = md5($pwd);
21. $account = $mobile . '|' . $pwd;
22. $account = encrypt($account, '', 1209600); //两周时间=60X60X24X14秒   加密存储
23. cookie('\_RM', $account, array('expire' => 1209600, 'httponly'=>1));  //写入客户端缓存
24. }
25. //...
26. }

FrontendController控制器为前台基础控制器

1. class FrontendController extends Controller{
2. /\* 两周内自动登录 \*/
3. public function \_initialize(){
4. if (!IS\_LOGIN && $account = cookie('\_RM')) {
5. $account = decrypt($account);  //解密
6. list($mobile, $pwd) = explode('|', $account);  //解析出用户名和密码
7. $user = D('User');
8. $id = $user->getUserField(array('mobile' => $mobile, 'pwd' => $pwd), 'id'); //判断是否存在这个用户
9. if (!is\_null($id)) {
10. $user->autoLogin($id); //传入用户ID，实现自动登录
11. } else {
12. cookie('\_RM', null); //清除记录密码的缓存
13. }
14. }
15. }
16. //....
17. }

**[应用Session变量控制用户登录时间](https://www.cnblogs.com/haohaoyuan/p/7797820.html)**

1、创建login\_ok.php页面，在页面中通过查询数据库来判断用户名密码是否正确，如果正确，则为Session变量赋值

$\_SESSION['name']=$name;

$\_SESSION['time']=time();

2、创建index.php页面，在页面中进行判断，如果Session变量有值且登录时间没有超过10分钟显示欢迎回来，如果十分钟内未操作，则显示登录超时，请重新登录

if($\_SESSION['time']==""){

echo "<script> alert('您无权限查看，请登录！');

location='login.php'</script>"; //不允许直接访问

} elseif((time()-$\_SESSION['time'])<60){

$\_SESSION['time']=time(); //将当前的时间戳赋值给Session变量

**[通过session\_id限制单个用户重复登录](https://www.cnblogs.com/haohaoyuan/p/7797941.html)**

我们实现该功能只需要在普通用户验证的基础上，加一层session\_id的验证就可以了。我们在用户登录的时候，将此时的session\_id写入数据表，session\_id在本次session有效期是不会变化的。如果在别的地方，登录了该账户，则新的session\_id值会写入数据表，则在验证用户信息的时候，当前用户的session\_id和数据表里面的session\_id不一致。

| **id** | **name** | **password** | **session\_id** |
| --- | --- | --- | --- |
| 1 | root | 63a9f0ea7bb98050796b649e85481845 | 3olcdjkj5jjaq2u9t30mbuna96 |

主流的做法貌似是，用户登陆的时候，除了验证登陆的SESSION，另外产生并保存一个HASH过的信息。每次需要检查是否登陆的时候，把用户提交上来的HASH跟服务器的HASH对比，如果不一致说明异地登陆了，就注销该用户的所有session，重新登陆。

对于单点登录的话，可以将用户ID，一个秘钥值，一个浏览器信息，IP地址，sessionId 等着几个值加密存储到缓存中，姑且称其为令牌token，这样的话，A用户用一个UserID登录后，就会生成对应的token，这样B用户再用同样的UserID登录后，校验是否存在该UserID的token，如果不存在就登录生成token，如果存在则对比B用户【用户ID，一个秘钥值，一个浏览器信息，IP地址，sessionId 】这几项值加密生成的token和缓存已存在的token是否一样，不一样则删除原token，使用新的token。这样的话A刷新网页的话，去找token发现和对应生成的token不一样则退出。当然缓存也要设置一下时间，必须在多长时间重新登录一次就行。

//ini\_set('session.cookie\_lifetime', 0);//设置session在浏览器关闭时失效

//ini\_set('session.gc\_maxlifetime', 3600); //session在浏览器未关闭时的持续存活时间

//$ipdress = $\_SERVER['REMOTE\_ADDR'];

//权限验证的时候一定要判断数据库写入session\_id是不是和$\_session里面的session\_id是不是一致，如果不一致就退出，这个防止同一个账户被多个人同时登陆的实现方法就是基于这个session\_id

//一定要设置session的存活时间，和session关掉页面就session消除

//如果有其他人登陆的时候，会写入新的session\_id，这样旧的用户就会被权限验证不通过，但是不能做到实时效果，这样就挤掉了旧用户，这里有个缺陷，就是每次验证权限的时候就需要查询数据库，如果可以吧这个数据放在redis就比较好

//或者统一管理session的时候，比如放在数据库就更好了，记住吧session\_id也存入数据库，或者放在memcash，redis就比较方便，比如在redis进行管理的时候，就可以直接更新掉存在session\_id

indexController.class.php

<?php

class indexController extends Controller{

function \_\_construct(){

parent::\_\_construct();

}

// 显示用户信息页面

function index(){

$loginController = new loginController();

$loginController->isLogin(); //验证是否合法登录

$this->smarty->assign("name",$\_SESSION['name']);

$this->smarty->assign("id",$\_SESSION['id']);

$this->smarty->display("info.html");

}

}

loginController.class.php

<?php

class loginController extends Controller{

private $loginModel;

function \_\_construct(){

parent::\_\_construct();

$this->loginModel = new loginModel();

}

//@return bool\* 判断当前登录用户是否合法

function isLogin(){

$res = $this->loginModel->getUserInfoById($\_SESSION['id']);

if(empty($res)){

echo "未登录";

exit;

}else{

$sql\_token = md5($res['name'].$res['password']);

if($sql\_token != $\_SESSION['token']){

echo "用户验证失败";

exit;

}else{

if(session\_id()!=$res['session\_id']){ //最关键的代码--------------！！！！

echo "该账户已在别处登录";exit;

}else{

return true;

}

}

}

}

// 执行登录操作

function action(){

$name = $\_POST['name'];

$password = md5($\_POST['password']);

$session\_id = session\_id();

$res = $this->loginModel->loginCheck($name,$password,$session\_id); //验证登录成功就更新session\_id字段并返回用户信息，否则返回false

if($res){

$\_SESSION['id'] = $res['id'];

$\_SESSION['name'] = $name;

$\_SESSION['token'] = md5($name.$password);

header("Location:http://login.com/index.php/index/index");

}else{

header("Location:http://login.com/index.php/login/form");

}

}

}

loginModel.class.php

<?php

class loginModel extends Model{

function getUserInfoById($id){

$sparam = array('id','name','password','session\_id');

$wparam = array("id[=]"=>$id);

$data = $this->select("admin",$sparam,$wparam);

if(empty($data)){

return false;

}else{

return $data[0];

}

}

function loginCheck($name,$password,$session\_id){

$sparam = array('id','name','password',"session\_id");

$wparam = array("name[=]"=>$name);

$data = $this->select("admin",$sparam,$wparam);

if(empty($data)){

return false;

}else{

$info = $data[0];

if($info['password']!=$password){

return false; //验证失败

}else{

//验证成功，更新session

$newdata = ["session\_id"=>$session\_id];

$wparam = ["id[=]"=>$info['id']];

$this->update("admin",$newdata,$wparam);

return $info;

}

}

}

}