**session 存入redis缓存**

**为什么要把SESSION保存在缓存**

就PHP来说，语言本身支持的session是以文件的方式保存到磁盘文件中，保存在指定的文件夹中，保存的路径可以在配置文件中设置或者在程序中使用函数session\_save\_path()进行设置，但是这么做有弊端，

第一就是保存到文件系统中，效率低，只要有用到session就会从好多个文件中查找指定的sessionid，效率很低。

第二就是当用到多台服务器的时候可能会出现，session丢失问题（其实是保存在了其他服务器上）。

当然了，保存在缓存中可以解决上面的问题，如果使用php本身的session函数，可以使用 session\_set\_save\_handler()函数很方便的对session的处理过程进行重新控制。如果不用php的session系列函数， 可以自己编写个类似的session函数，也是可以的，我现在做的这个项目就是这样，会根据用户的mid、登录时间进行求hash作为 sessionId，每次请求的时候都必须加上sessionId才算合法（第一次登录的时候是不需要的，这个时候会创建sessionId，返回给客户 端），这么做也很方便、简洁高效的。当然了，我这篇文章主要说的是在php自身的SESSION中”做做手脚”。

**SESSION保存在缓存中**

php将缓存保存到redis中，可以使用配置文件，对session的处理和保存做修改，当然了，在程序中使用ini\_set()函数去修改也可以，这个很方便测试，我这里就使用这种方式，当然了，要是生产环境还是建议使用配置文件。

如果想简单操作session入redis操作可以将一下代码运行一下

<?php

ini\_set("session.save\_handler", "redis");

ini\_set("session.save\_path", "tcp://localhost:6379");

session\_start();

header("Content-type:text/html;charset=utf-8");

$\_SESSION['view'] = 'zhangsan';

echo $\_SESSION['view'];

这里设置session.save\_handler方式为redis，session.save\_path为redis的地址和端口，设置之后刷新，再回头查看redis，会发现redis中的生成了sessionId，sessionId和浏览器请求的是一样的，

如果是memcache

<?php

ini\_set("session.save\_handler", "memcache");

ini\_set("session.save\_path", "tcp://localhost:11211");

session\_start();

header("Content-type:text/html;charset=utf-8");

$\_SESSION['view'] = 'zhangsan';

echo $\_SESSION['view'];

也可以使用

Session\_set\_save\_handler(‘open’,’close’,’ read’,’ write’,’ destory’,’ gc’);

用法如下自定义一个Redis\_session类

<?php

class RedisSession{

private $\_redis = array(

'handler' => null, //数据库连接句柄

'host' => null, //redis端口号

'port' => null,

);

public function \_\_construct($array = array()){

isset($array['host'])?$array['host']:"false";

isset($array['port'])?$array['host']:"false";

$this->\_redis = array\_merge($this->\_redis, $array);

}

public function begin(){

//设置session处理函数

session\_set\_save\_handler(

array($this, 'open'),

array($this, 'close'),

array($this, 'read'),

array($this, 'write'),

array($this, 'destory'),

array($this, 'gc')

);

}

public function open(){

$redis = new Redis();

$redis->connect($this->\_redis['host'], $this->\_redis['port']);

if(!$redis){

return false;

}

$this->\_redis['handler'] = $redis;

$this->gc(null);

return true;

}

//关

public function close(){

return $this->\_redis['handler']->close();

}

//读

public function read($session\_id){

return $this->\_redis['handler']->get($session\_id);

}

//写

public function write($sessionId, $sessionData){

return $this->\_redis['handler']->set($sessionId, $sessionData);

}

public function destory($sessionId){

return $this->\_redis['handler']->delete($sessionId) >= 1 ? true : false;

}

public function gc(){

//获取所有sessionid，让过期的释放掉

$this->\_redis['handler']->keys("\*");

return true;

}

}

$ses = new RedisSession(array('host'=>'127.0.0.1','port'=>'6379'));

$ses->begin();

session\_start();

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

echo $\_SESSION['name'];

这样就可以实现session数据如redis代码执行过程中必须安装redis才可以