

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. 项目架构

Application 项目代码目录

Config 配置文件目录

Controller 控制器

Admin 后台控制器

Home 前台控制器

Model 模型

View 视图

Admin 后台视图

Home 前台视图

Framework 框架代码目录

Core 框架核心

Lib 扩展类库

Public 静态资源

1. 所有的参数放在配置文件目录中
2. 实现步骤

1、定义路径常量

2、导入配置文件

3、确定路由：需要3个参数，p：平台 c：控制器 a：方法

4、自定义加载类

5、注册自定义加载类，让加载类的方法在缺少类的时候自动执行

6、请求分发

1. 执行成功($this->success())和失败($this->error)的跳转

## Cookie

### 思考一个场景

在demo2.php中如何访问demo1.php的变量？

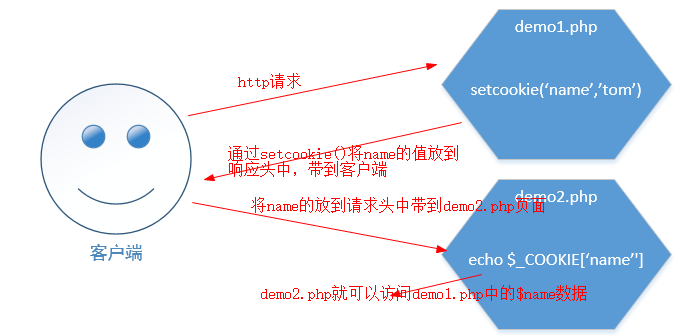
方法一：包含文件

方法二：GET传递

方法三：cookie技术

### cookie介绍

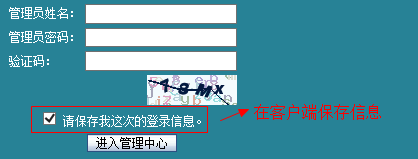
cookie是一个小的信息包，保存在客户端



### cookie的应用

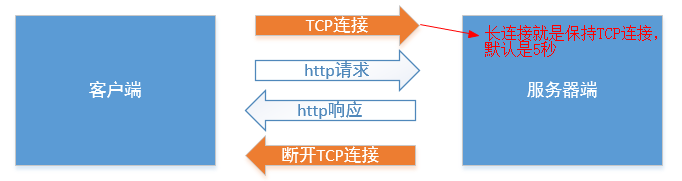
应用一：在不同的页面访问同一个字

应用二：记住用户名和密码

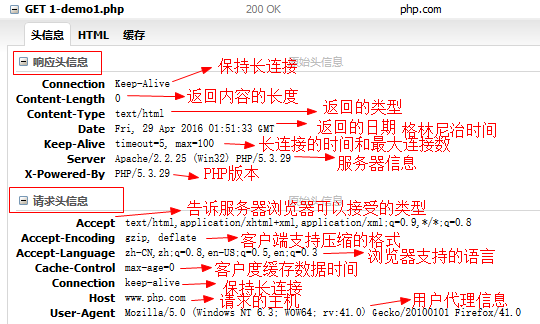


### 长连接

在客户端执行http请求的之前，会通过TCP协议与服务器建立连接，http响应后，TCP连接不销毁，以便在下一次http请求的时候直接使用连接，保存时间默认是5秒。



### 头信息



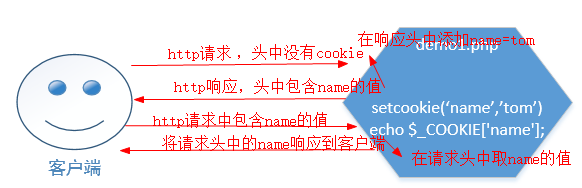
### cookie语法

setcookie(名字，值，过期时间，有效目录，子域名)

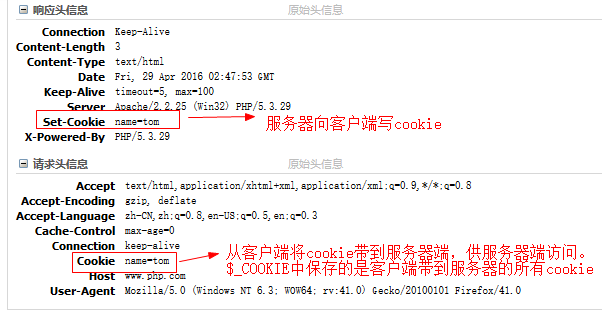
思考：如下代码执行结果是什么？

setcookie('name','tom');

echo $\_COOKIE['name'];



答：第一次访问的时候不能取出name的值，第二次以后就可以访问到name的值了。

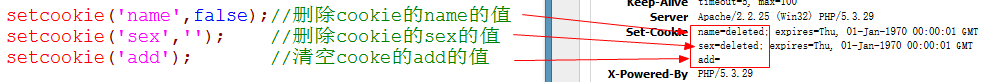


### 设置cookie

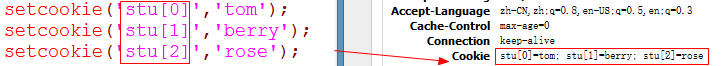


脚下留心：cookie只能存放字符串和数字

### 删除cookie



### cookie支持下标



脚下留心：Cookie在浏览器上没有数组的概念，只是浏览器请求到服务器后，PHP将同名的值转成数组。

### 临时性cookie和永久性cookie

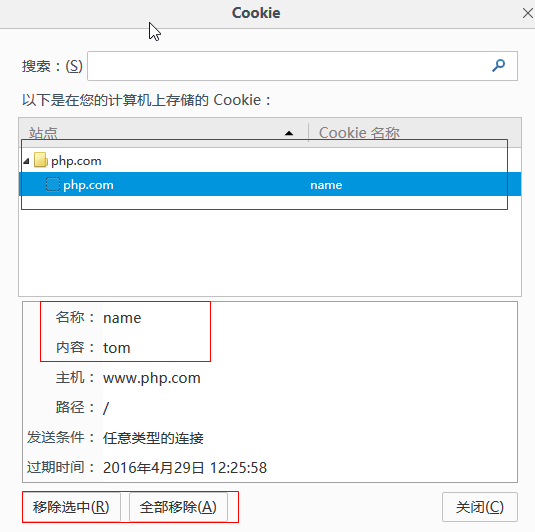
1. 临时性cookie：关闭浏览器后cookie消失
2. 永久性cookie：关闭浏览器后cookie不消失，给临时性cookie添加一个过期时间即可



脚下留心：cookie的过期时间是一个时间戳

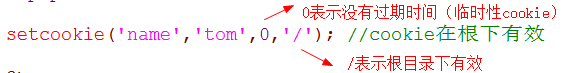
永久性cookie用一个文件的形式保留在客户端，可以删除

以Firefox为例



### cookie的有效目录

cookie默认只能在当前目录和后代目录中生效



### cookie支持子域名

不同的子域名是独立的网站。他们之间默认只能访问自己的cookie.

默认情况下，cookie变量是区分子域名的。



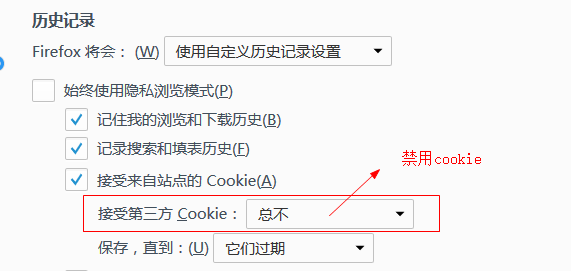
### cookie缺点

1. 安全性低
2. 增加了请求时的数据负载，大部分浏览器对cookie的限制是4K。
3. 可控性差

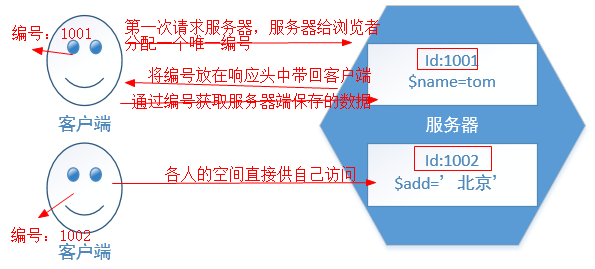
ie



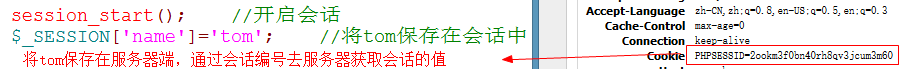
火狐



## session（会话）



1. session技术基于cooke技术
2. 要使用一个会话，必须显式的开启一个会话，默认情况下，会话不会自动开启



1. 要自动启动会话，在php.ini中找到session.auto\_start=0这行，把0改为1，并重启服务器



1. session可以保存除了资源以外的任何数据类型
2. session\_id()获取会话编号，session\_name()获取会话名称



1. 会话重复开启会提示性报错

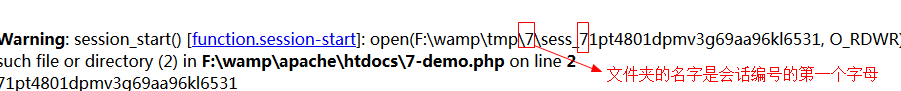
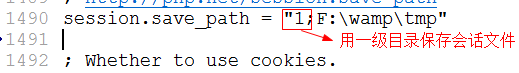


### 会话的执行过程

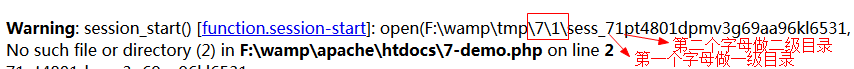
1. 使用session\_start()函数，PHP从session仓库中加载已经存储的会话
2. 当session第一次启动时候，服务器给客户分配一个唯一的会话id保存在cookie中
3. 执行PHP的时候，通过会话编号去仓库中存入或读取值
4. 当PHP执行结束时，没有被销毁的session变量被自动保存到服务器的session仓库中。

### 关于session的配置

1. session.save\_path = "N;/path"：用来设置会话保存的路径



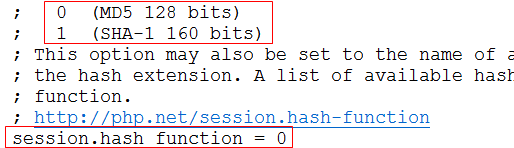
多学一招：session.save\_path可以设置通过N级目录来分类存放session文件，如果N为1，就是用会话编号的第一个字母做文件夹名称；如果N为2，第一级用第一个字母，第二级用第二个字母



1. session.name = PHPSESSID 设置会话编号的名字



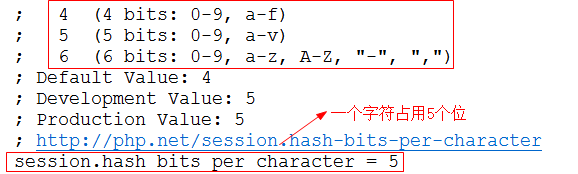
1. session.hash\_function = 0 会话算法



1. session.hash\_bits\_per\_character = 5 一个字符占用5个位

2的5次方是32，会话有32个字符组成

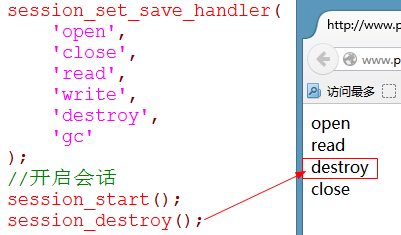
0-9是10个字符，a-v是22个字符，加起来正好是32个字符



1. session.cookie\_lifetime = 0：sessionID在客户端cookie中存储的时间
2. session.cookie\_path = / 保存会话的cookie整站有效
3. session.cookie\_domain = 保存会话的cookie在当前域名下有效
4. session.save\_handler = files 会话按文件的形式存储
5. session.gc\_maxlifetime = 1440 session在服务器存储的时间（生命周期），1440秒=24分钟

### session\_destroy()：销毁会话

1. 调用此函数会删除储存介质中的文件
2. 如果是自定义销毁会话函数，调用此函数会自动调用自定义销毁函数
3. 一旦销毁会话就不执行写操作



1. 正常情况下脚本结束要执行写操作。

## 修改session存储（session入库）

### 创建数据表(sess)

此表用来存储会话

drop table if exists sess;

create table sess(

sess\_id varchar(32) primary key comment '会话编号',

sess\_value text not null comment '会话的值',

sess\_expires int not null

)engine=innodb charset=utf8;

### session\_set\_save\_handler()

— 设置用户自定义会话存储函数

<?php

ini\_set('session.save\_handler','user'); //自定义会话存储

/\*\*

\*打开会话

\*/

function open() {

//echo 'open<br>';

mysql\_connect('localhost:3306','root','aa');

mysql\_query('set names utf8');

mysql\_query('use php4');

}

/\*\*

\*关闭会话

\*/

function close() {

//echo 'close<br>';

}

/\*\*

\*读取会话

\*@param $sess\_id string 会话编号

\*/

function read($sess\_id) {

//echo 'read<br>';

$sql="select sess\_value from sess where sess\_id='$sess\_id'";

$rs=mysql\_query($sql);

if($rows=mysql\_fetch\_row($rs))

return $rows[0];

}

/\*\*

\*写入会话

@param $sess\_id string 会话编号

@param $sess\_value string 会话值

\*/

function write($sess\_id,$sess\_value) {

//echo 'write<br>';

$time=time();

$sql="insert into sess values ('$sess\_id','$sess\_value',$time) on duplicate key update sess\_value='$sess\_value'";

return mysql\_query($sql);

}

/\*\*

\*销毁会话，销毁自己的会话

\*/

function destroy($sess\_id) {

//echo 'destroy<br>';

$sql="delete from sess where sess\_id='$sess\_id'";

return mysql\_query($sql);

}

/\*\*

\*垃圾回收，所有的过期会话

\*/

function gc($maxlifetime) {

//echo 'gc';

$time=time()-$maxlifetime;

$sql="delete from sess where sess\_expires<$time";

return mysql\_query($sql);

}

//更改会话存储

session\_set\_save\_handler(

'open',

'close',

'read',

'write',

'destroy',

'gc'

);

//开启会话

session\_start();

//session\_destroy();

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

$\_SESSION['sex']='男';

echo $\_SESSION['name'];

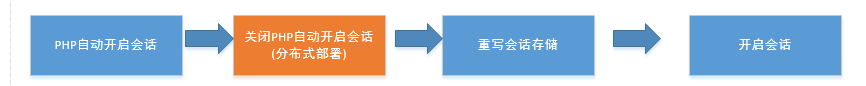
$\_SESSION['name']='berry';

脚下留心：session\_set\_save\_handler()必须写在session\_start()之前。

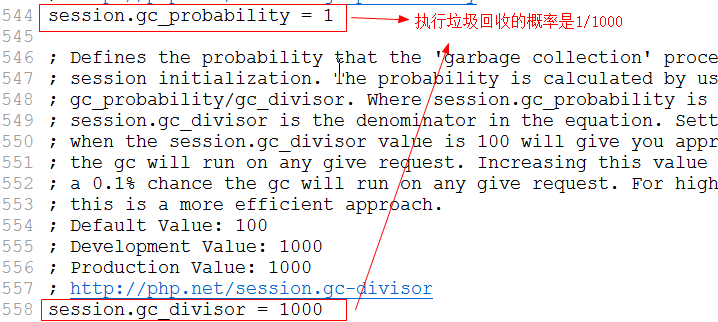
思考题：

已知session\_set\_save\_handler()必须写在session\_start()之前；如果session.auto\_start=1的开启会话就在重写会话存储之前了，如何解决？

答：通过分布式部署文件(.htaccess)关闭自动开启会话



### 垃圾回收



默认情况下，gc执行的概率是1/1000。

## session和cookie的区别

### 相同点

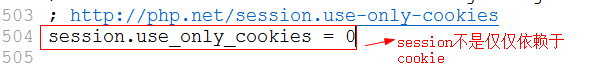
1. 都是会话技术，有生命周期
2. 都是无状态性：服务器将请求内容发送到客户端后，服务器不在记录客户端的信息成为无状态性。

### 不同点

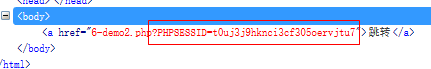
|  |  |  |
| --- | --- | --- |
|  | Cookie | session |
| 存储地方 | 浏览器中 | 服务器端 |
| 安全性 | 低 | 高 |
| 数据负载 | 4K | 没有限制 |
| 数据类型 | 字符串和数字 | 除了资源以外的所有类型 |
| 可控性 | 低 | 高 |

## 禁用cookie

1. 如果禁用了cookie，基于cookie的所有技术默认都无法使用
2. 可以通过配置文件改变



1. PHP自动在URL地址上，表单中加上会话编号



## 将session入库封装到项目中

在框架的Lib文件夹下，新建一个SessionLib.class.php页面

<?php

/\*

\* session入库的工具类

\*/

class SessionLib{

private $db; //用来保存MySQLDB类的示例

public function \_\_construct(){

//用户自定义存储

session\_set\_save\_handler(

array($this,'open'),

array($this,'close'),

array($this,'read'),

array($this,'write'),

array($this,'destroy'),

array($this,'gc')

);

}

/\*\*

\*打开会话

\*/

public function open() {

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

}

/\*\*

\*关闭会话

\*/

public function close() {

return true;

}

/\*\*

\*读取会话

\*@param $sess\_id string 会话编号

\*/

public function read($sess\_id) {

$sql="select sess\_value from sess where sess\_id='$sess\_id'";

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

}

/\*\*

\*写入会话

@param $sess\_id string 会话编号

@param $sess\_value string 会话值

\*/

public function write($sess\_id,$sess\_value) {

$time=time();

$sql="insert into sess values ('$sess\_id','$sess\_value',$time) on duplicate key update sess\_value='$sess\_value'";

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

}

/\*\*

\*销毁会话，销毁自己的会话

\*/

public function destroy($sess\_id) {

//echo 'destroy<br>';

$sql="delete from sess where sess\_id='$sess\_id'";

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

}

/\*\*

\*垃圾回收，所有的过期会话

\*/

public function gc($maxlifetime) {

$time=time()-$maxlifetime;

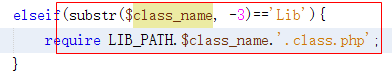
$sql="delete from sess where sess\_expires<$time";

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

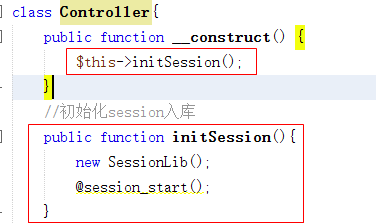
}

}

自动加载sessionLib类



在基础控制器中，初始化session入库



测试

