**PHP核心编程第四天--会话技术**

目录

[二、 初识会话技术 3](#_Toc485053794)

[1. 会话及会话技术 3](#_Toc485053795)

[1) 为什么需要会话技术 3](#_Toc485053796)

[2) 什么是会话技术 3](#_Toc485053797)

[3) 什么是会话周期 3](#_Toc485053798)

[2. 会话技术分类 4](#_Toc485053799)

[三、 COOKIE技术初步 4](#_Toc485053800)

[1. 原理初步 4](#_Toc485053801)

[1) 概念 4](#_Toc485053802)

[2. 设置cookie 4](#_Toc485053803)

[1) 设置语法 4](#_Toc485053804)

[2) 如何在浏览器查看COOKIE 4](#_Toc485053805)

[3) 数据类型的要求 5](#_Toc485053806)

[3. 读取COOKIE数据 5](#_Toc485053807)

[4. 修改cookie数据 6](#_Toc485053808)

[5. 删除cookie 6](#_Toc485053809)

[6. COOKIE运行原理进阶 6](#_Toc485053810)

[1) 原理图 7](#_Toc485053811)

[2) HTTP协议角度（结合原理图） 7](#_Toc485053812)

[四、 COOKIE数据的属性 8](#_Toc485053813)

[1. 有效期（生命周期） 8](#_Toc485053814)

[1) 有效期 8](#_Toc485053815)

[2) 默认生命周期 9](#_Toc485053816)

[3) 自定义生命周期 9](#_Toc485053817)

[4) 案例：展示用户的上次登录时间： 10](#_Toc485053818)

[2. cookie的有效路径（数据的作用范围：目录） 11](#_Toc485053819)

[3. COOKIE的有效域名 12](#_Toc485053820)

[1) 默认有效域名 12](#_Toc485053821)

[2) 自定义有效域名，实现跨子域 12](#_Toc485053822)

[4. COOKIE案例：记住用户名 14](#_Toc485053823)

[5. COOKIE总结 16](#_Toc485053824)

[1) 语法 16](#_Toc485053825)

[2) 默认值 16](#_Toc485053826)

[五、 session技术初步 16](#_Toc485053827)

[1. 什么是session 16](#_Toc485053828)

[2. 原理 17](#_Toc485053829)

[3. 基本操作 17](#_Toc485053830)

[1) 开启session机制 17](#_Toc485053831)

[2) session数据的读写操作 18](#_Toc485053832)

[3) 一个重要错误提示 18](#_Toc485053833)

[4) 应用：登录模块 19](#_Toc485053834)

[5) 删除一个session数据 20](#_Toc485053835)

[6) 删除所有session信息 20](#_Toc485053836)

[7) 销毁session信息 21](#_Toc485053837)

[4. 从HTTP协议及COOKIE角度分析session原理： 21](#_Toc485053838)

[1) 运行原理示意图 21](#_Toc485053839)

[2) session之老王的保险箱 21](#_Toc485053840)

[3) 深层次理解SESSION 21](#_Toc485053841)

[4) 两问SESSION 23](#_Toc485053842)

[5. SESSION配置选项 23](#_Toc485053843)

[1) session.name 23](#_Toc485053844)

[2) session.cookie\_lifetime 23](#_Toc485053845)

[3) session.cookie\_path 24](#_Toc485053846)

[4) session.cookie\_domain 24](#_Toc485053847)

[6. session与COOKIE之间的区别与联系 24](#_Toc485053848)

[1) 区别 24](#_Toc485053849)

[2) 联系 25](#_Toc485053850)

[7. 案例：使用SESSION技术实现购物车 25](#_Toc485053851)

[1) 实现步骤 25](#_Toc485053852)

[六、 SESSION的垃圾回收机制 26](#_Toc485053853)

[1. 垃圾回收机制介绍 26](#_Toc485053854)

[2. session垃圾回收机制 27](#_Toc485053855)

[3. 参数设置 27](#_Toc485053856)

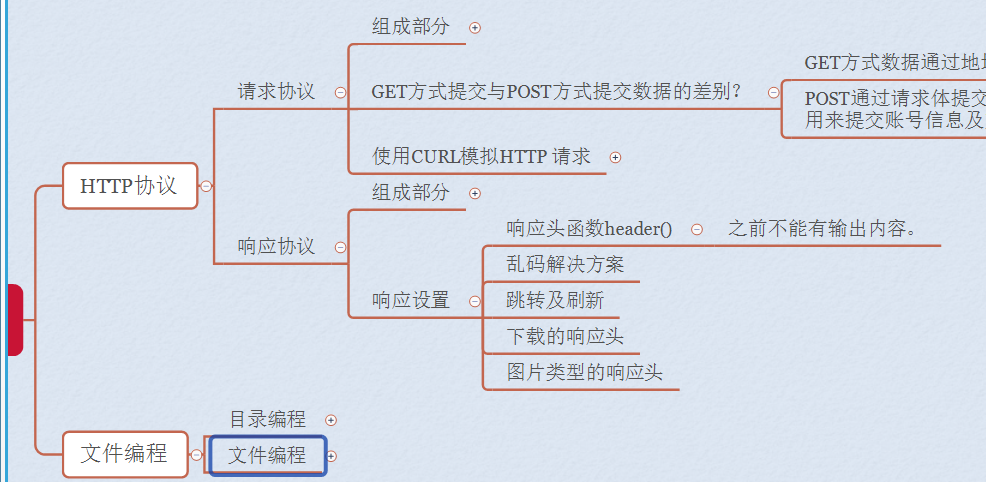
[七、 禁用COOKIE，如何使用SESSION 27](#_Toc485053857)

[1. 禁用COOKIE的方法： 28](#_Toc485053858)

[2. 禁用COOKIE后,购物车模块的解决思路 28](#_Toc485053859)

[3. 实现禁用COOKIE情况下的session使用 28](#_Toc485053860)

**回顾：**



**思考：**

* COOKIE如何增删改查？
* SESSION如何增删改查？
* COOKIE的属性有哪些？如何设置？
* SESSION与COOKIE的区别及联系
* 禁用COOKIE，如何使用SESSION？

# 初识会话技术

HTTP协议的特点：无记忆、无状态。同一浏览器，先后多次请求服务器，服务器不能识别浏览器。一些功能无法实现，如登录模块。

## 会话及会话技术

会话：浏览器与服务器之间的数据交流。

### 为什么需要会话技术

HTTP协议是无状态、无记忆的。多次请求之间，无任何的联系，无法把请求的状态保存下去，即同一用户请求同一网站的不同页面，服务器无法识别是否是同一用户发起的请求。用户无法进行连续的业务逻辑。

如：

登录，在A页面登录，请求B页面，提示登录。

### 什么是会话技术

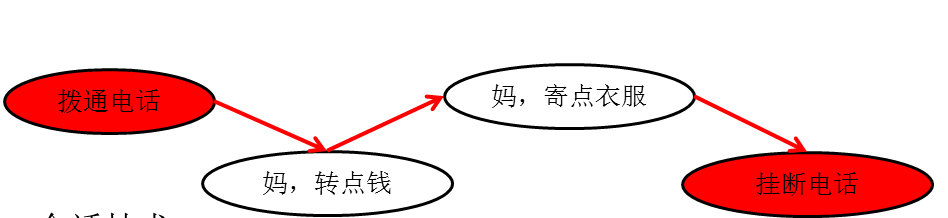
在同一台浏览器对服务器的多次请求中，将数据持久化存储的技术，以实现连续的业务逻辑

### 什么是会话周期

默认会话周期：从打开浏览器，访问一些web资源，直至关闭浏览器为止，这个时间间隔为会话周期。

类似于事务的处理过程。

会话周期结束可能导致某些数据失效。



## 会话技术分类

按照数据持久化的位置进行区分，

数据持久化存储在服务器，为**session技术**。

数据持久化存储在浏览器，为**cookie技术**。

# COOKIE技术初步

应用如：已登录过的网站，可实现用户名的自动填充。



## 原理初步

### 概念

简单理解：浏览器请求时携带自身的身份信息。（带的胸牌）

允许服务器脚本（PHP脚本）在浏览器端存储数据，并以此跟踪及识别用户的技术。

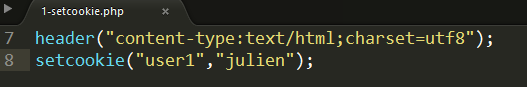
数据设置后，浏览器再次请求服务器指定页面时，将相关的数据发送给服务器脚本，供脚本使用。

## 设置cookie

cookie本质为简单数据，基本操作逻辑为增删改查。

### 设置语法

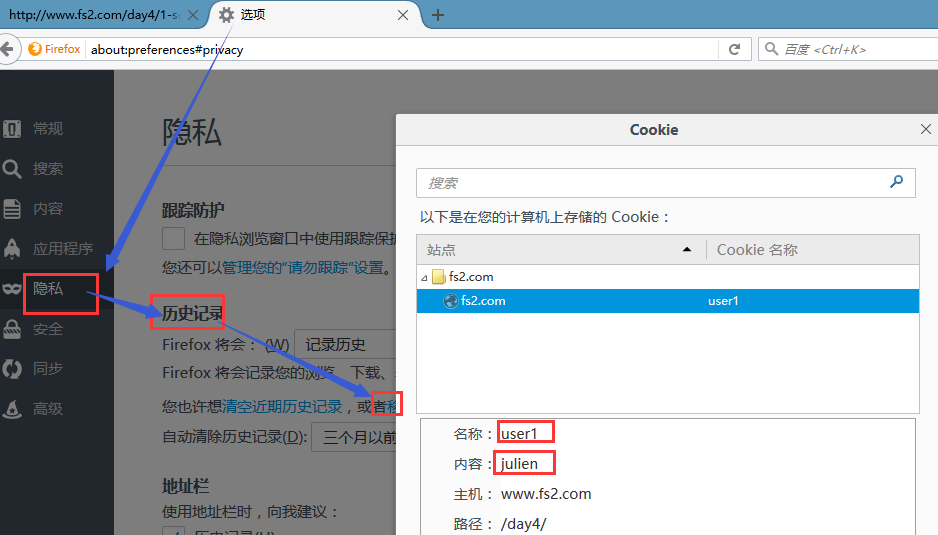
setcookie(数据名，数据)；



### 如何在浏览器查看COOKIE

火狐浏览器：选项---隐私---历史记录---移除单个COOKIE。

谷歌浏览器：设置—隐私设置—内容设置—cookie—所有cookie和网站数据

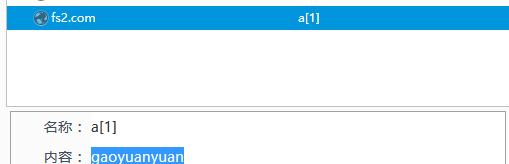


### 数据类型的要求

存储的类型一般为字符串。无法存储复杂的数据。可以存储汉字，在浏览器查看时为转码后的数据。

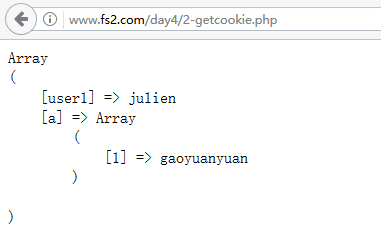
类型特例：类似数组的方式。

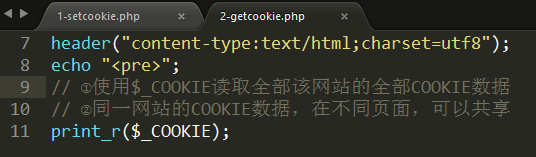




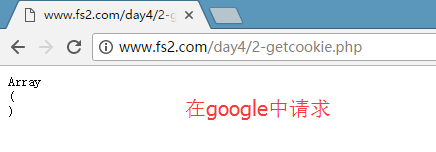
## 读取COOKIE数据

* 可以通过全局变量$\_COOKIE读取。
* 同一网站的不同页面的数据共享：对于同一个浏览器，在网站的A页面设置的COOKIE数据，可以在网站的B脚本中读取到。

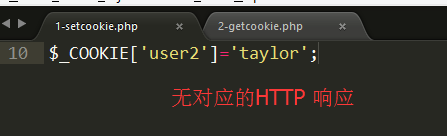




* COOKIE数据是基于浏览器的：在A浏览器请求页面设置的COOKIE数据，在B浏览器请求同样的页面，无法访问。



* 无法通过$\_COOKIE添加数组元素的方式设置数据，实现数据在客户端的长久保存（setcookie本质为通过HTTP响应协议来通知浏览器对数据进行操作）。

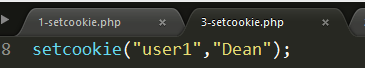


## 修改cookie数据

setcookie语法，将已经存在的COOKIE变量设置为其他值。

setcookie(“变量名”，“新值”);

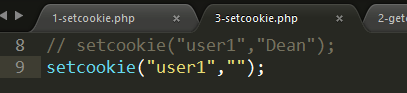
执行过程，相当于修改变量值。



## 删除cookie

没有显式的删除函数。可以通过设置COOKIE数据为空字符串来实现。COOKIE数据为空后，浏览器自动消除该数据空间。

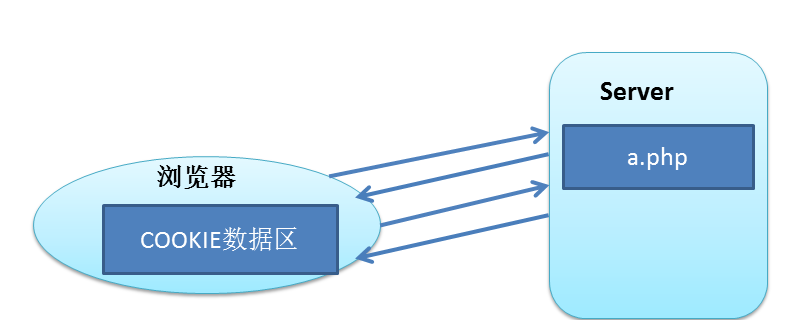
setcookie(“变量名”，“”)；



## COOKIE运行原理进阶

### 原理图

请求页面时，浏览器按照服务器脚本的要求处理数据。



* 1. COOKIE数据保存在浏览器端。
  2. COOKIE数据本身为浏览器的身份信息。在请求中，通过HTTP协议传输COOKIE数据。
  3. 服务器脚本接收到COOKIE数据后，可以识别浏览器的身份信息。

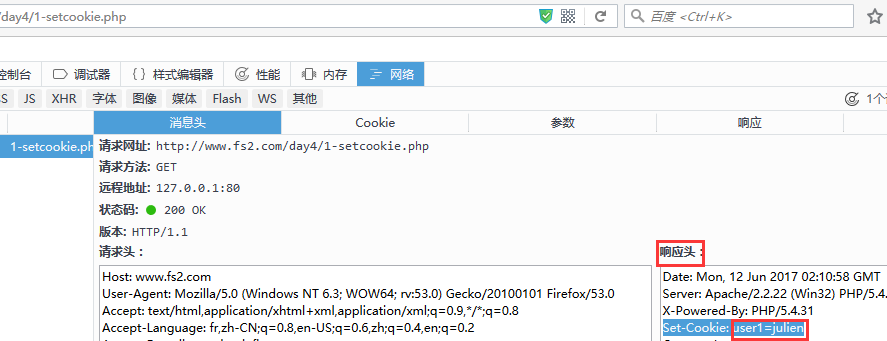
### HTTP协议角度（结合原理图）

通过HTTP请求及响应进行COOKIE数据的读取及设置。

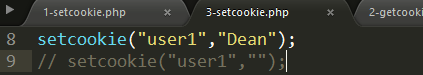
* 1. 第一次请求。在响应中：服务器通知浏览器设置数据。

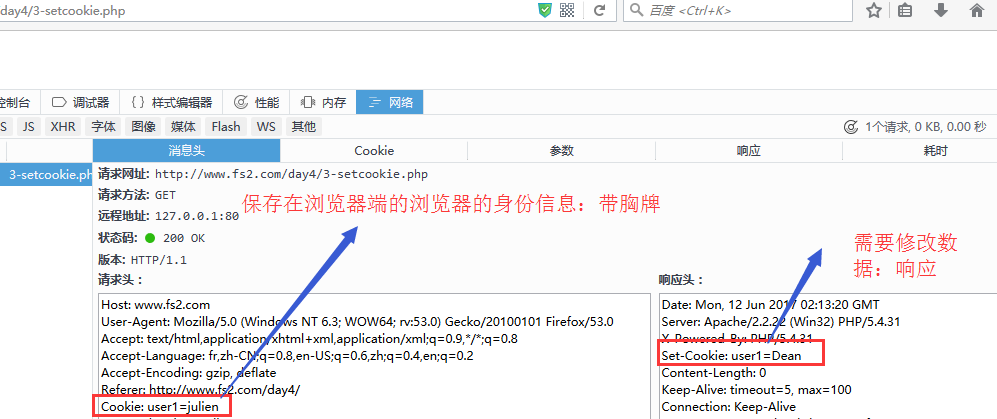


set-cookie:通知浏览器在浏览器端保存对应的数据。

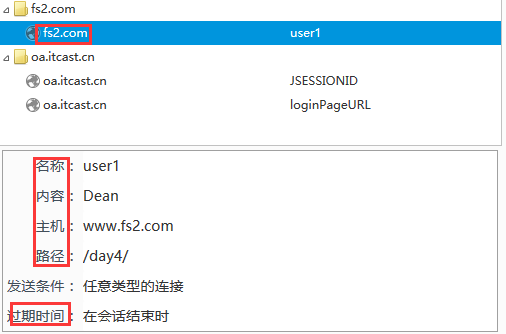


* 1. **其他次**：修改数据



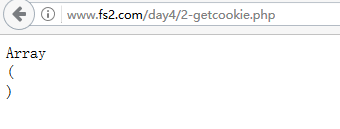


# COOKIE数据的属性



## 有效期（生命周期）

关闭浏览器之后再读取：



COOKIE数据的默认的生命周期是会话周期，即在浏览器窗口关闭之前，一直存在。关闭后立即消失。

关闭浏览器：相当于结束默认的会话，COOKIE数据自动消失。再次打开浏览器，相同页面读取数据，无法读到。

好处：减少空间占用，减少HTTP传输的数据量，提高数据安全。

### 有效期

设置COOKIE的第三个参数，为设置其有效期。

setcookie(“名称”,”值”，“有效期”);

即COOKIE数据的有效时间，使用时间戳（秒数）来表示。HTTP请求中，仅携带在有效期内的COOKIE数据。可以减少携带的数据量。

### 默认生命周期

为会话周期：浏览器关闭，数据即消失。

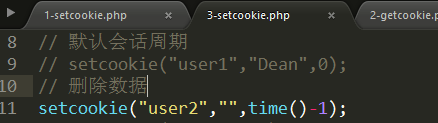


### 自定义生命周期

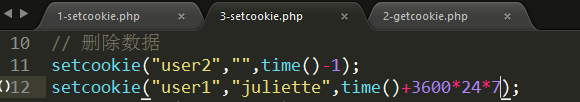
* 默认会话周期：



* 删除单个COOKIE数据的另外一种方法：



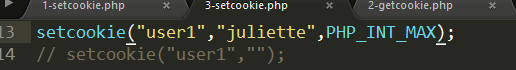
* 一周：





* 永久有效：

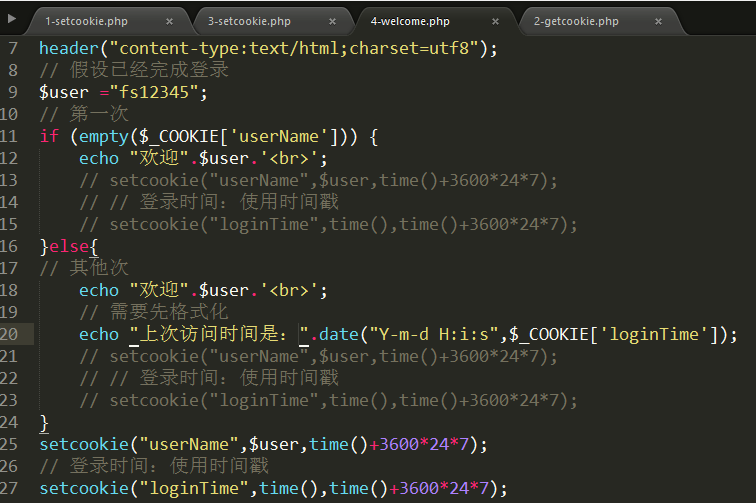
使用常量PHP\_INT\_MAX



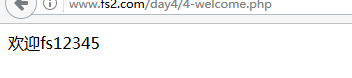


### 案例：展示用户的上次登录时间：

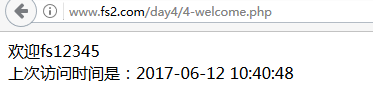
第一次访问提示欢迎访问，其他次提示用户名及上次的访问时间。



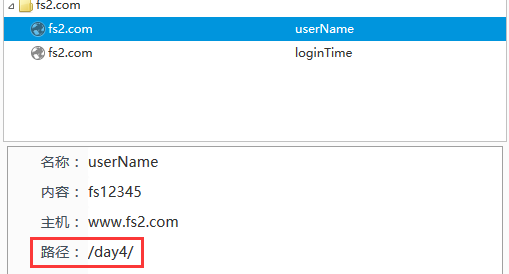
第一次：



其他次：



## cookie的有效路径（数据的作用范围：目录）



设置后，默认当前目录及子目录有效。



上一级目录，不能读取：

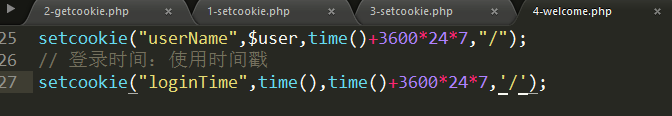


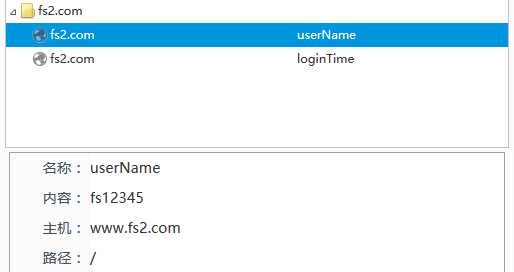
设置COOKIE的第四个参数，为设置其有效路径。

* 默认在当前目录（脚本所在的目录）及子目录，脚本可以直接访问对应COOKIE信息。父目录无法访问子目录的COOKIE信息。
* 通常设置为/，表示网站根目录、整站有效

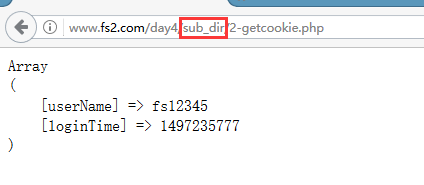
将上例的COOKIE数据设置为整站有效：

setcookie(名称，值，有效期，有效路径);

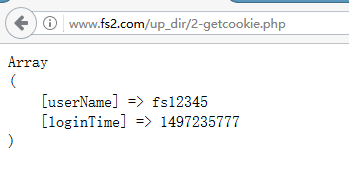




子目录：

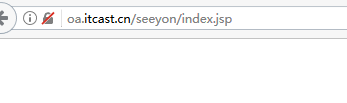


上一级：



## COOKIE的有效域名



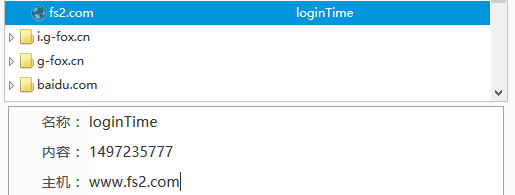


域名：itcast.cn

主机(子域名或者二级域名)：[www.itcast.cn](http://www.itcast.cn) oa.itcast.cn tb.itcast.cn mail.itcast.cn

### 默认有效域名

当前域名有效。



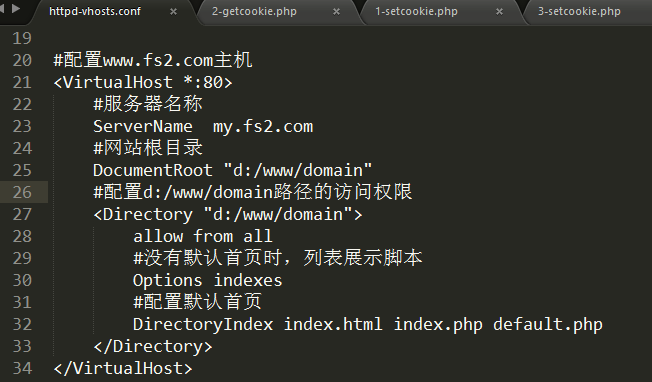
A网站设置的COOKIE数据，不会发送到B网站。

### 自定义有效域名，实现跨子域

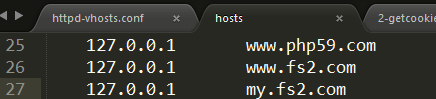
一级域名与二级域名：

整站(整个域名)有效的配置方法：

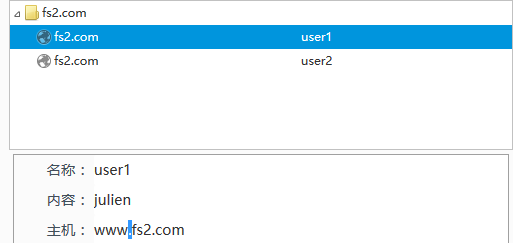
1. 虚拟主机配置



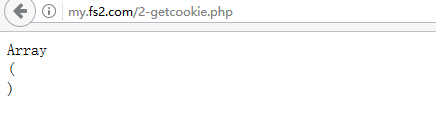
1. 域名解析



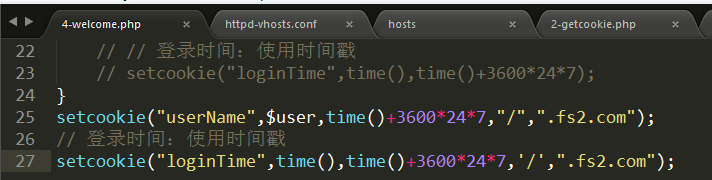
1. 设置COOKIE数据：[www.fs2.com](http://www.fs2.com).

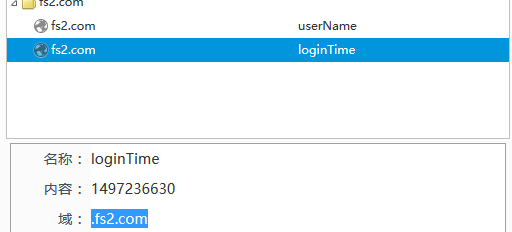


1. 读取COOKIE数据：my.fs2.com

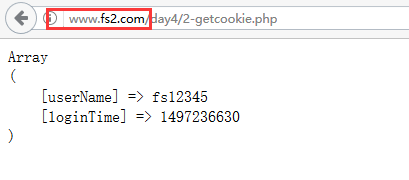


1. 正确的设置方式为：





在[www.fs2.com](http://www.fs2.com)读取：



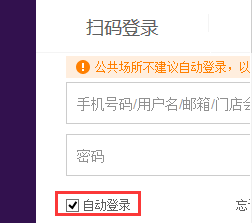
同级域名：my.fs2.com:



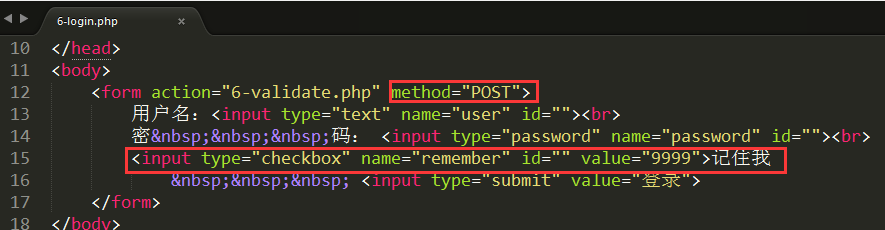
## COOKIE案例：记住用户名

需求：用户勾选记住我选项后，下次登陆自动填充用户名。

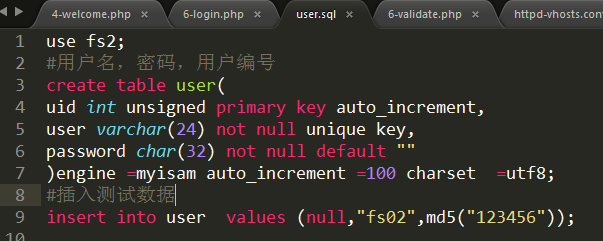
<https://login.gome.com.cn/login?intcmp=sy-public01002>

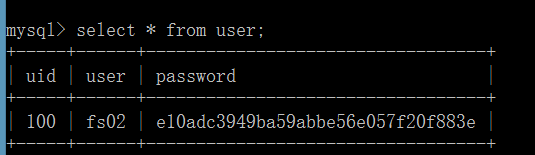
7

* 1. 登录表单：用户名、密码、记住我多选项。

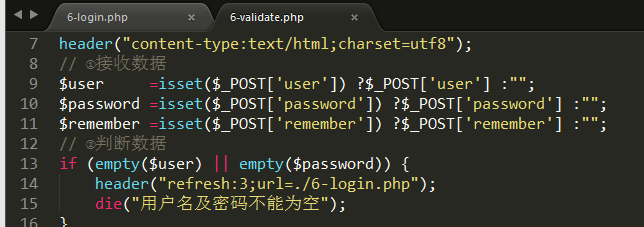


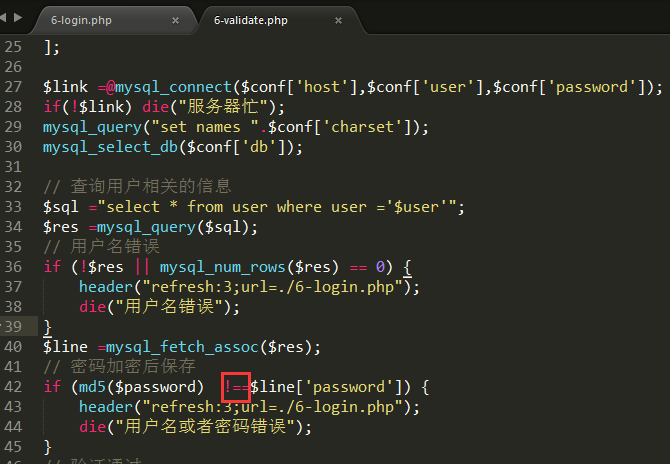
* 1. 需要一个用户表。



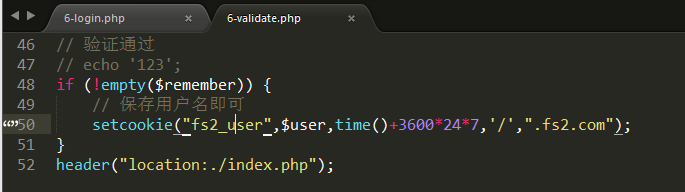


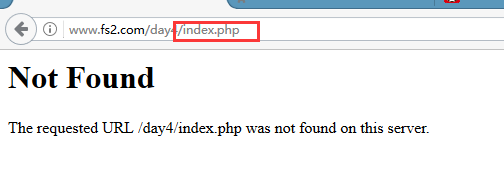
* 1. 数据验证处理。验证用户名及密码。





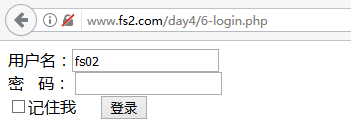
* 1. 检查是否勾选了记住我。勾选，将用户名名使用COOKIE保存下来。

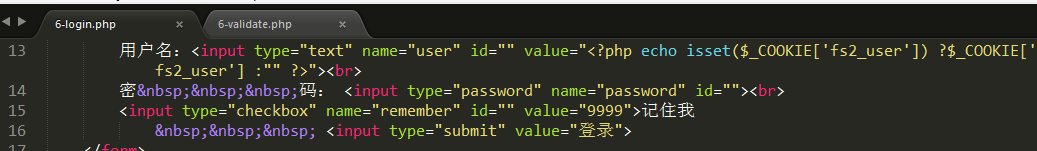






* 1. 下次登录。如果有COOKIE，自动填充用户名。





## COOKIE总结

### 语法

setcookie(“名称”,”值”，“有效期”，“有效路径”，“有效域名”)；

### 默认值

有效期：默认会话周期。浏览器不关闭，数据一直存在。浏览器关闭数据消失。0.

有效路径：当前目录及子目录有效。设置整站有效:/

有效域名：当前域名。实现跨子域：.itcast.cn.

# session技术初步

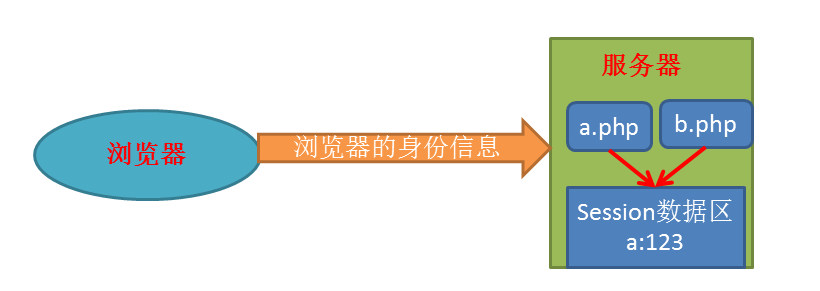
当前COOKIE技术实现登录的缺陷：

* 1. COOKIE数据通过HTTP协议传输。关于该网站的全部COOKIE数据，在每次请求中都会携带。
  2. COOKIE数据存放在客户端，有安全隐患。

## 什么是session

为了实现业务逻辑的连续性，将部分信息持久化存储在服务器端，可以保证数据的安全性及各个脚本之间的共享。

## 原理



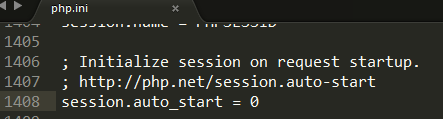
* 将客户端的状态信息存储在服务器端。
* 可以在网站的不同页面之间传递数据（页面数据共享）
* 可以识别不同的浏览器（基于浏览器）。

## 基本操作

### 开启session机制

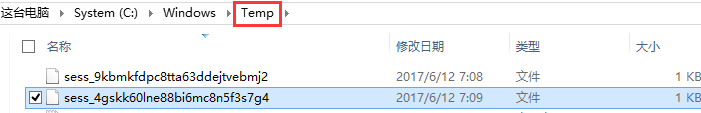
* PHP.ini文件中配置SESSION自动开启的选项（永久生效，任意页面可以直接使用session技术。不建议）

session.auto\_start=1



* 使用session\_start()在当前的脚本中开启

session文件的默认保存路径session.save\_path：/tmp

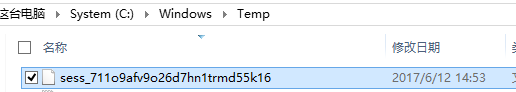


查看session文件的方法：使用win+r调出运行窗口---temp。其中以sess开头的文件。

注意：session\_start()必须在程序最开始执行。不能有任何输出内容。

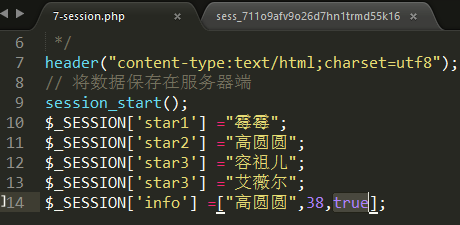
开启之后，立即创建浏览器对应的session文件。

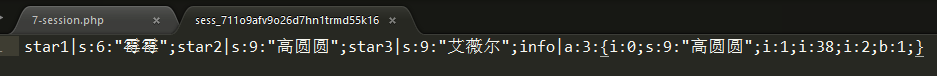




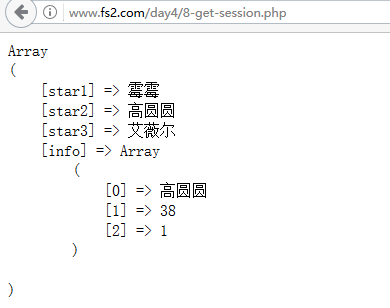
### session数据的读写操作

* SESSION数据的增删改查操作都是通过数组$\_SESSION进行的。元素的下标不能使用数字。
* 存储的数据类型几乎可以为PHP的全部数据类型。(除资源型外，本质为序列化与反序列化的过程)。
* 一个session文件中可以存放多个键值对。





* 数据存储在session文件中，实现多个页面共享session文件中的数据。



* 服务器通过浏览器的身份信息，来返回存储在服务器的相应数据信息。



### 一个重要错误提示

未开启session机制的情况：

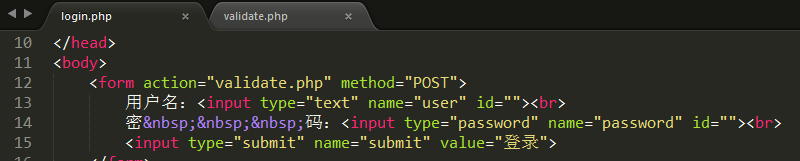


### 应用：登录模块

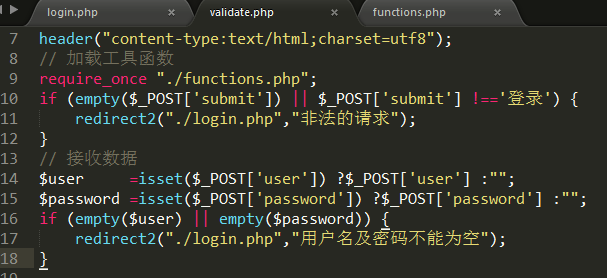
* 1. 创建用户数据表



* 1. 设计登录表单

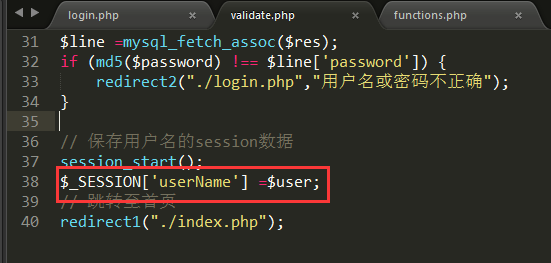


* 1. 接收及判断数据



* 1. 验证的核心逻辑





需要在表单页面完善：

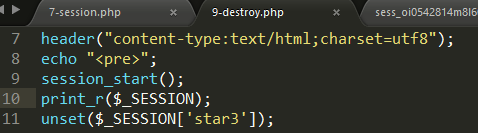


* 1. 重要的页面需要登录验证：添加、修改及删除



### 删除一个session数据

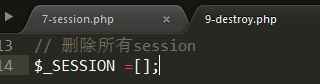
可以使用unset语法删除一个session数据（数组的一个元素）。删除的是session文件中的指定数据，不删除整个session文件。



delete :删除行。数据表还在。

### 删除所有session信息

$\_SESSION设置为空数组。本质为使用空数组覆盖session文件中对应的数据。session文件不被删除。



类似于：truncate 清空。

### 销毁session信息

* 销毁含义

删除session文件，导致所有的与该客户端有关的session数据消失。

* 销毁语法

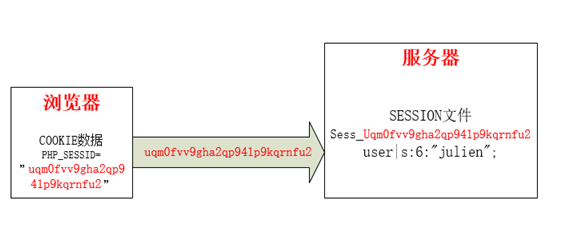
session\_destroy();

直接删除session文件，导致所有的session数据丢失。

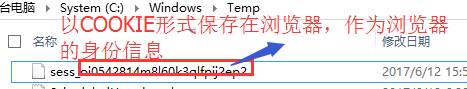
## 从HTTP协议及COOKIE角度分析session原理：

登录成功后，在新的浏览器中打开首页，仍然需要登录：

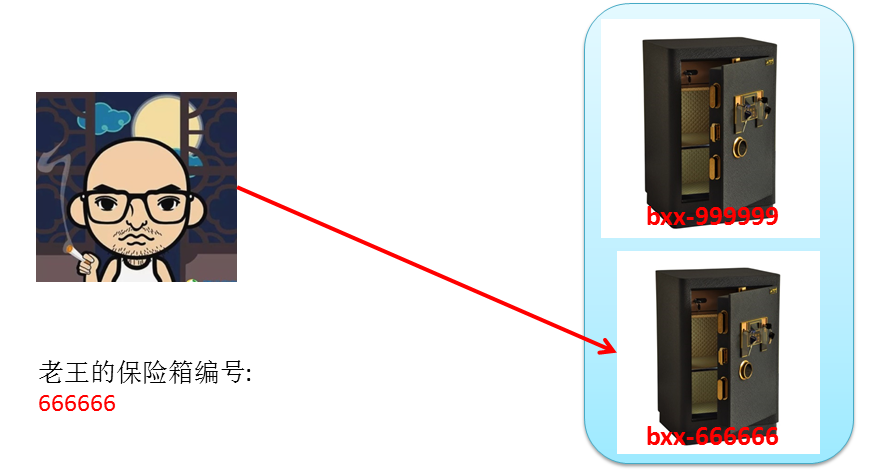
### 运行原理示意图



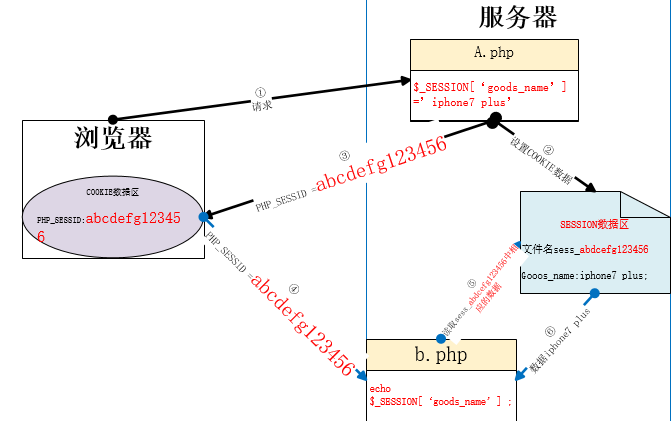
核心：使用身份信息，读取对应文件的session数据。



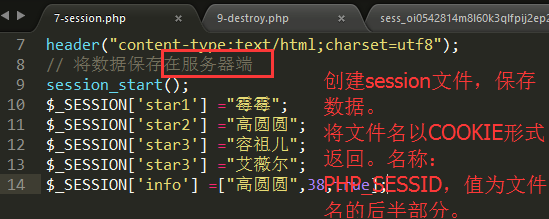
### session之老王的保险箱



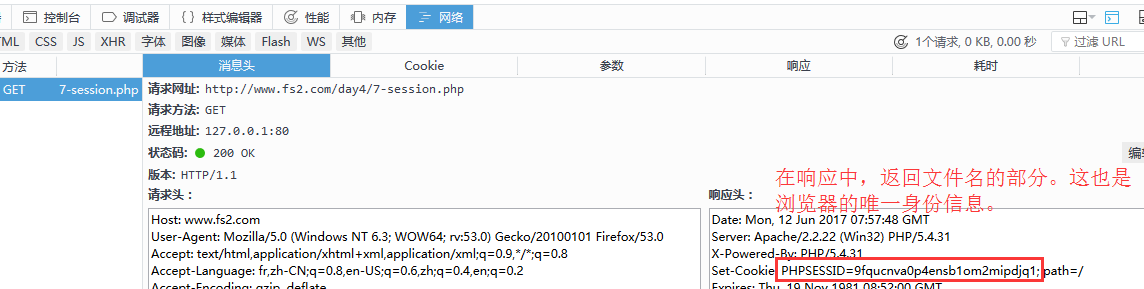
### 深层次理解SESSION



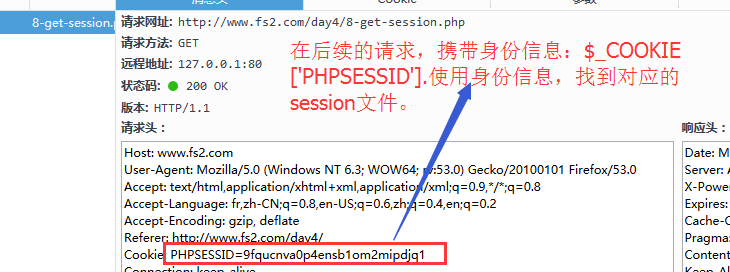
* 1. 浏览器发起请求
  2. a脚本设置SESSION数据，自动创建session文件。文件名是按照浏览器版本等、使用一定的算法生成的，唯一对应浏览器的身份信息。



* 1. 将浏览器的身份信息以COOKIE数据返回，保存在浏览器的COOKIE和数据区。数据名为PHPSESSID，数据值为浏览器的身份信息。



* 1. 请求其他页面：携带身份信息：数据名为PHPSESSID的COOKIE信息



* 1. 按照cookie中PHPSESSID的数据值，对应地打开及读取session的全部信息$\_SESSION
  2. 可以读取单个session数据
* **服务器端设置session信**息：

基于浏览器设置（对每个浏览器计算出唯一的身份信息）使用其他浏览器读取session信息，无法读取。原因是：身份信息不一致。服务器判断没有该浏览器对应的session文件，自动创建新的session。

* **session共享的原理：**

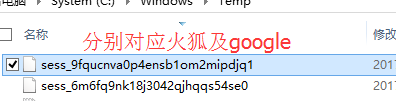
COOKIE信息一致，可以打开同一个session文件。

### 两问SESSION

* session\_start（）的作用？

根据客户端（COOKIE[‘PHPSESSID’]），创建不同的session文件。session文件已经存在，则将session数据读取到超全局变量$\_SESSION中,不创建新的session文件。不存在，创建session文件。

可以通过在不同的浏览器执行相同的脚本进行检查。



* session数据如何实现基于浏览器的数据存储而不混淆？

SesssionID按照一定的算法生成，用来唯一识别浏览器，可以保证其唯一性及随机性。

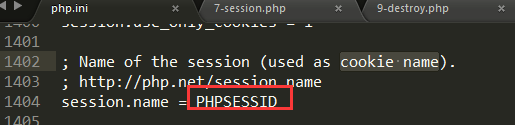
ssessionID是客户端和服务器之间通过HTTPRequest及HTTPResponse传来传去的。(即为session文件的一部分，唯一的确定对应的session文件)

使用相同浏览器访问，按照sessionID判断。如果session文件已经存在，则直接读取写入。使用不同浏览器访问，每次使用不同的sessionID，导致服务器认为请求来自是不同的客户端，自动创建新的session文件来处理数据。

## SESSION配置选项

### session.name

设置COOKIE中保存信息的名称。

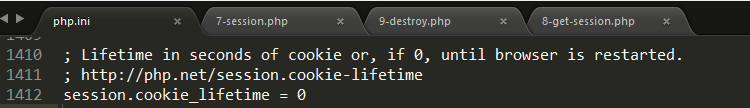


session文件的名称返回给COOKIE，保存在对应的COOKIE数组元素PHPSESSID中。

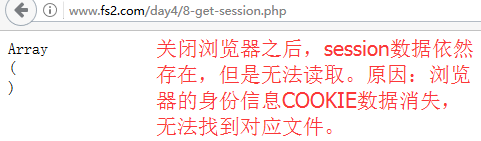
### session.cookie\_lifetime

凭借cookie信息（浏览器的身份信息）读取服务器端的相应的session文件的信息。

* 1. session.cookie\_lifetime=0,意味着COOKIE数据的生命周期为会话周期，关闭浏览器，COOKIE数据消失，无法使用浏览器的身份信息查到session信息。但是session暂时还是存在的。



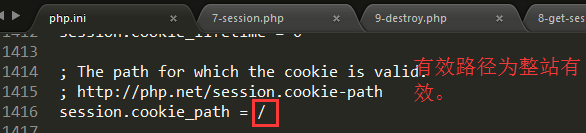




* 1. 浏览器禁用COOKIE，无法读取已存储的信息。

### session.cookie\_path

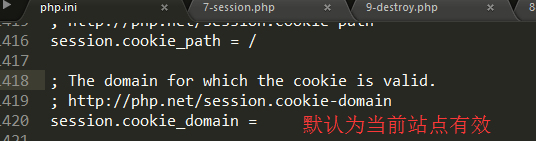
session所使用的COOKIE的有效路径。



### session.cookie\_domain

session所使用的COOKIE的有效域名。

默认为：当前站点有效



## session与COOKIE之间的区别与联系

### 区别

* **从存储位置**：

COOKIE存储在浏览器，SESSION存储的服务器端。

* 数据安全性：

session存储在服务器端，不易获取，更加安全。

* 传输的数据量：

每次请求，COOKIE传输量较大。传递该网站相关的全部的COOKIE数据。

* 支持的数据类型：

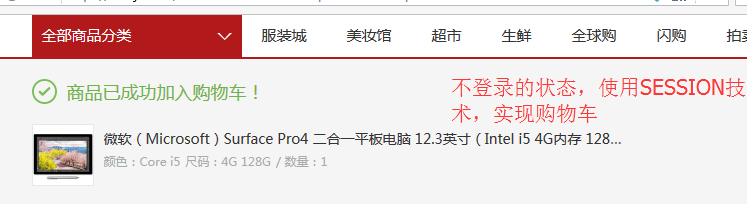
COOKIE支持字符串，SESSION几乎所有的数据类型。

### 联系

* 都是用来解决HTTP协议的无状态、无记忆的不足之处，以实现业务的连续性，如登录及购物车模块的实现。
* 二者都是会话技术的范畴 。
* SESSION需要依赖COOKIE（浏览器的身份信息）进行数据的传递。禁用COOKIE后，session无法正常使用。

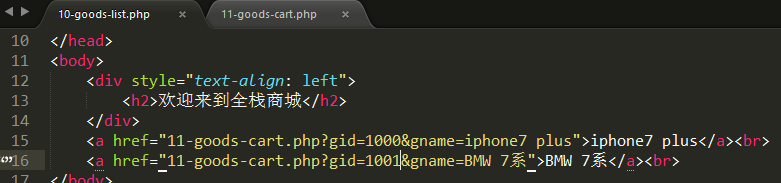
## 案例：使用SESSION技术实现购物车

用户未登录状态，即可添加商品到购物车：



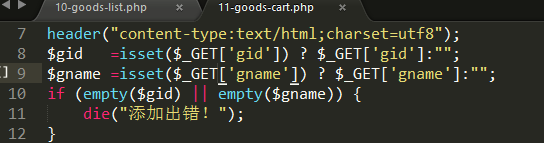
### 实现步骤

* 1. 展示商品的列表（有链接地址，必须添加参数）

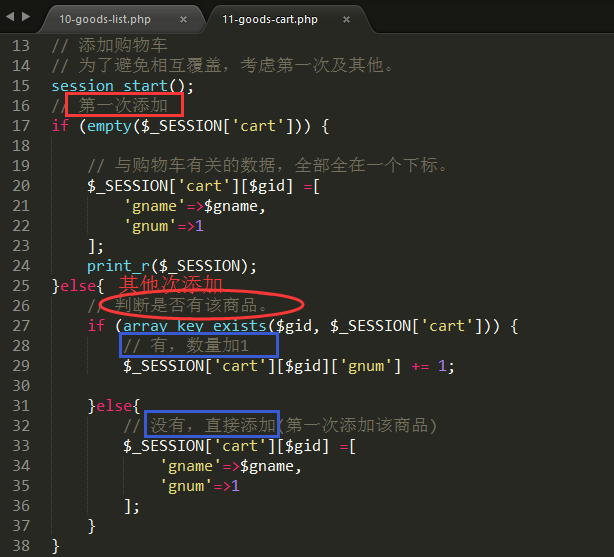




* 1. 点击链接（传递商品编号，名称）
  2. 在处理页面，先接受数据，保存在session中。



* 1. 在SESSION中添加数据



如何判断数组元素的下标：

array\_key\_exists(下标，数组)；

* 1. 以列表形式展示购物车的商品。



# SESSION的垃圾回收机制

$a =100; //在内存中变量名与值有引用关系。

unset($a); //断开该引用关系。

## 垃圾回收机制介绍

垃圾回收机制GC（garbage collection ）是一种动态存储分配方案。它会自动释放程序不再需要的、已分配的内存块。 垃圾回收机制可以让程序员不必过分关心程序内存分配，从而将更多的精力投入到业务逻辑.

## session垃圾回收机制

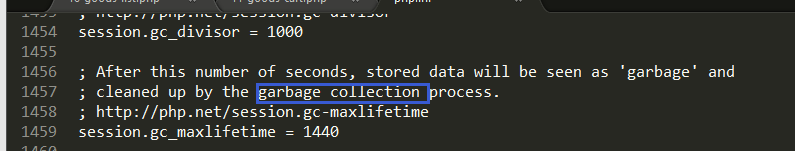
减少空间的占用。

– 扫描所有的Session信息，逐个判断session是否是垃圾文件。

垃圾文件的标准：

当前时间 ­­- session最后修改的时间 > session.gc\_maxlifetime（垃圾回收机制规定的最大有效期时间，默认为24分钟）

GC机制启动时，删除对应session文件。



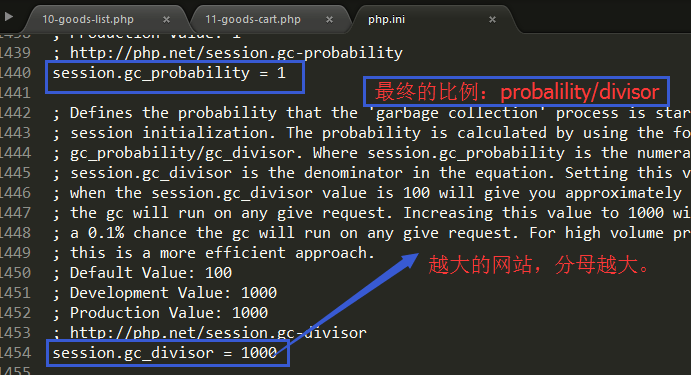
## 参数设置

客户端请求时，PHP按照一定的比例（概率）启动垃圾回收机制，大型网站一般设置分母较大。

session.gc\_probability

session.gc\_divisor

session.gc\_maxlifetime：默认为24分钟



* 1. 在24分钟之内（默认的session数据的生命周期），关闭浏览器（COOKIE数据使用默认的会话周期）。COOKIE数据丢失，浏览器身份信息丢失。此时SESSION数据还在，但是无法读取。
  2. 一直没有关闭浏器（COOKIE数据没有丢失，有身份信息），超过24分钟，SESSION 可能丢失，导致无法读取该数据。

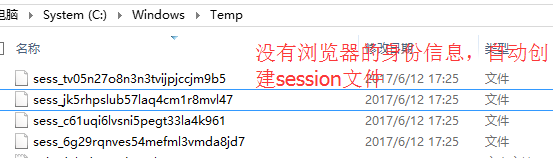
# 禁用COOKIE，如何使用SESSION

session基于COOKIE技术。

客户端方面，如果用户禁用COOKIE数据（无法保存浏览器的身份信息），购物车应用效果：每次请求创建新的session文件。

## 禁用COOKIE的方法：





## 禁用COOKIE后,购物车模块的解决思路

每次请求，浏览器没有携带身份信息（COOKIE），session\_start()执行找不到对应的session文件，自动创建新的session。商品的信息保存在多个session文件，无法实现共享。

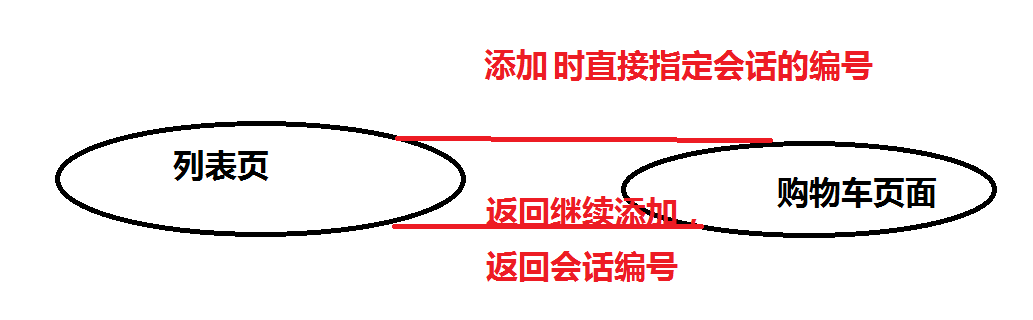
即解决多个页面不能共享session数据的问题。确保不同页面读取、写入同一个session文件。



使用session\_id函数：

session\_id();

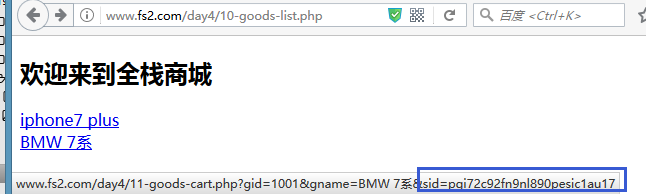
获取或者设置当前会话的编号。

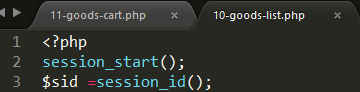


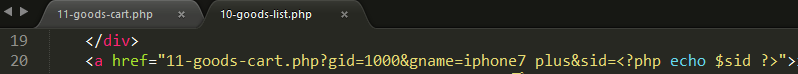
## 实现禁用COOKIE情况下的session使用

保证session\_start()操作时，已经读取对应的SESSION文件的名称。

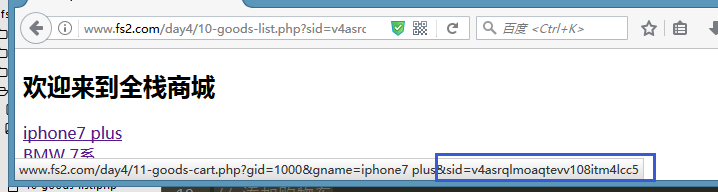
1. 在列表页添加SESSIONID参数：





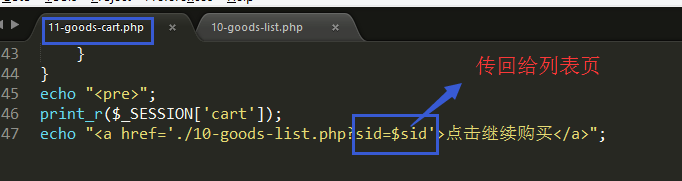


1. 在购物车页面，使用session\_start()之前接收参数，并作为会话的编号。





1. 在提示连接中包含会话参数：





1. 在列表页接收参数，作为本次会话的会话编号使用。

