# 跳转

业务逻辑：立即跳转，提示后跳转

## 立即跳转

Header(‘Location: URL’);

PHP（服务器）向浏览器发送一条指令Location，告知浏览器，应该重定向的URL上。

Header(‘Content-Type: ‘);

Header();就是PHP向浏览器发送指令（响应头信息），指令有多种。



原理：

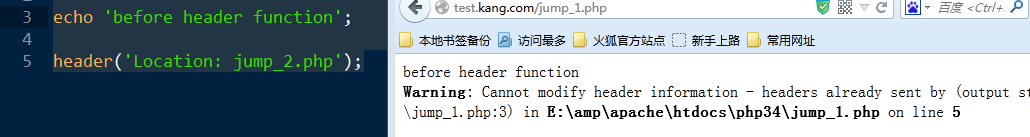


通过监视请求可以看到，共两次请求：



语法：

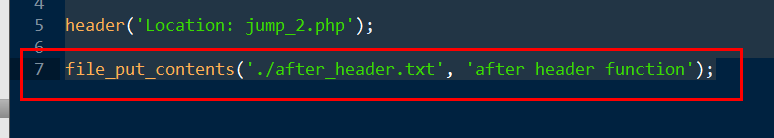
Header()函数前不能存在任何的输出内容。

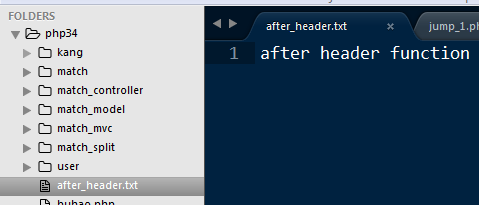


Tip：使用输出缓存output\_buffering可以处理。Php.ini



Header()函数后边的代码也会照常执行：





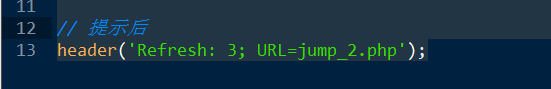
因此，跳转后要立即终止当前脚本的执行，才符合常规的跳转逻辑！



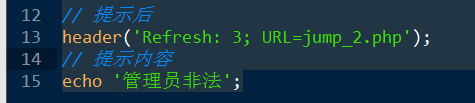
## 提示跳转

Header(‘Refresh: TIME; URL=目标URL’);

Refresh这个响应指令，是在N秒后，刷新到指定的目标URL上。



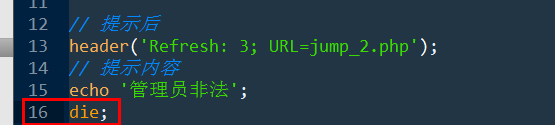
此时，会在当前URL请求上停留三秒。通过显示输出完成提示后跳转。



由于就是一个普通的页面展示，提示的样式，可以轻松定制！

语法：

Header()后的代码会执行，因此在处理好跳转相关操作后，脚本也应该被DIE掉！



## 案例中实现跳转

封装一个可以完成跳转的方法，需要时，调用即可！

控制器的功能！

该功能应该被所有的控制器动作所共享，因此，应该在基础控制器中被定义：

Controller.class.php



实现：

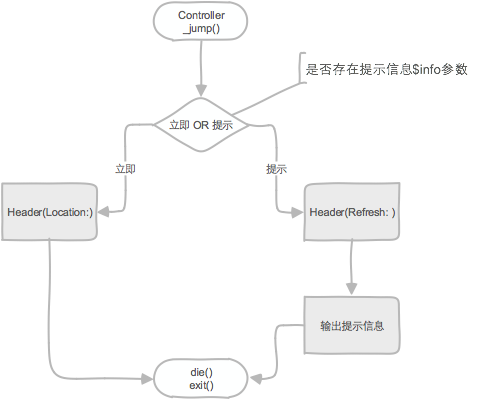
分析，参数：

$url, 目标URL地址

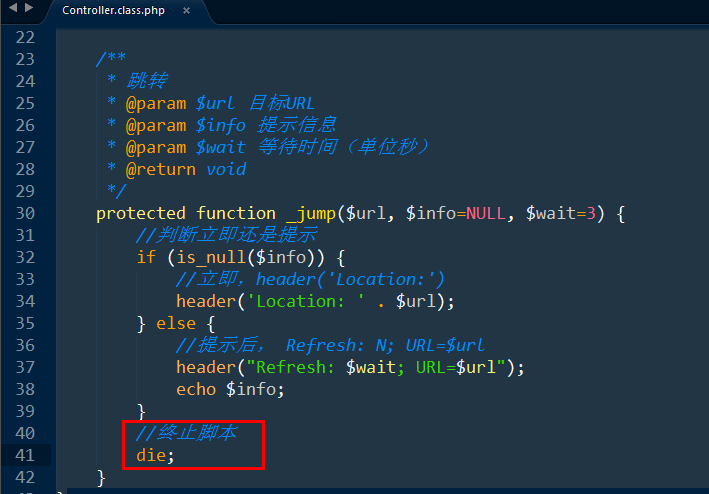
$info, 提示信息

$wait, 等待时间

为 info和wait增加默认参数，可以做到，即可完成立即，也可以完成提示跳转！

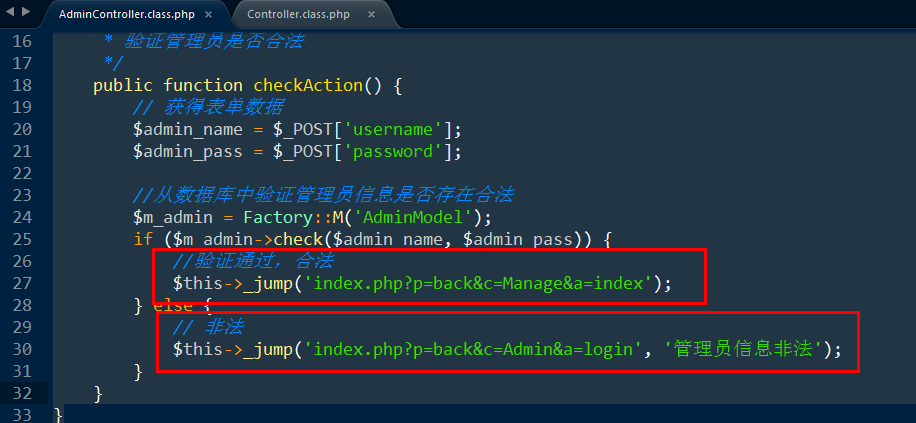


代码实现：



项目中使用：

Back/AdminController->checkAction()



测试：

创建一个后台首页动作！

Application/back/controller/ManageController.class.php

indexAction();

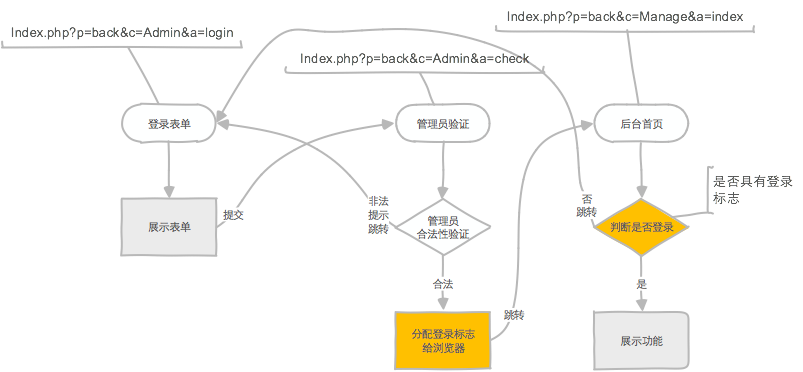


# 浏览器登录验证分析

在访问到后台首页（后台所有功能）时，应该先判断当前浏览器是否处于登录状态，再做功能的实现！如果没有登录，则跳转到登录页面。

如何判断当前浏览器用户是否登录？

管理员合法 –> 处于登录状态



登录标志：

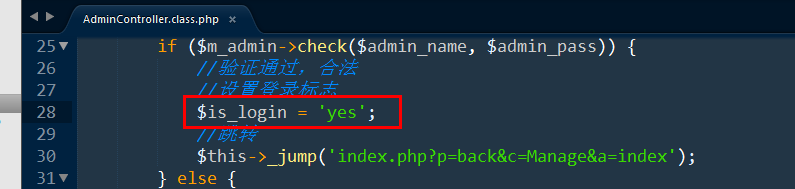
一个数据。

如何存储这个数据呢？

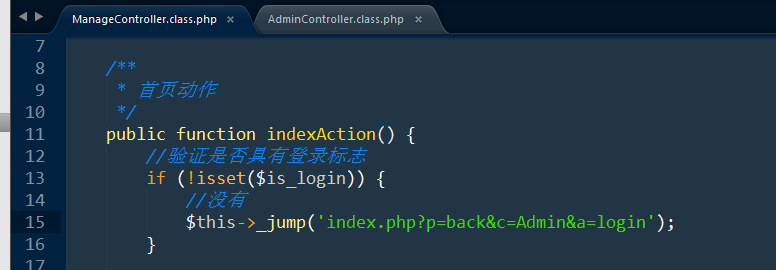
[不合理的]测试：

变量：

分配（back/AdminController->checkAction()）



验证（back/ManageController->indexAction()）



总结：不行，原因是：

数据的生命周期问题，对于PHP来说，所有数据（变量，常量）的最大生命周期，一个脚本周期（浏览器向服务器发出一次请求，服务器将该请求交由PHP处理，PHP处理完了给服务器，服务器响应给浏览器，就是一个脚本周期）

可见：登录标志的存储，需要可以跨脚本周期存储数据的一种形式（持久性存储数据的机制）

[不合理的]测试：

文件：

不行，原因是：

在服务器端管理的文件（数据库中）是不区分浏览器端，对于任何一个浏览器都是通用的！

可见：登录标志的存储，需要可以区分浏览器，并在该浏览器的多次请求周期内，可以持久存储的一种数据存储机制！

以上的存储机制，就是会话技术！

# 会话技术

可以使 在同一台浏览器与服务器的多次请求间，将数据持久存储的一种技术！

前提：B/S请求响应模式，是 无状态的。任意的请求间，不存在任何的联系，不能将请求状态保持下去。

会话技术的实现：

1Cookie，2Session

# COOKIE

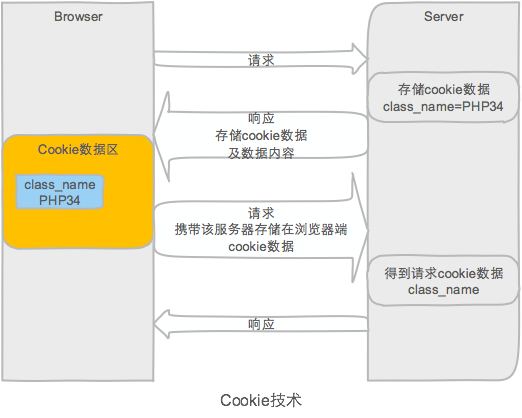
## 基本原理

允许服务器端脚本在浏览器端存储数据的一种技术。（可以见cookie是浏览器的技术）。

其特点是：

允许服务器向浏览器发送指令，用来管理存储在浏览器端的cookie数据。

浏览器如果存储了某台服务器所设置的cookie数据，浏览器再向服务器请求时，会携带存储的cookie数据到服务器端。



## Cookie的基本操作

### 设置（增删改）

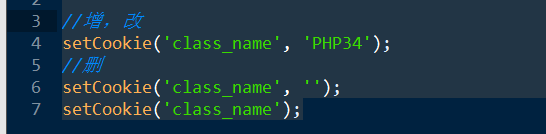
函数：

Setcookie(KEY, VALUE)

完成cookie数据的设置,

Key存在则更新，不存在则添加。

如果值为空字符串，表示删除该KEY



### 获取（查）

获取浏览器在请求时携带的COOKIE数据

使用 超全局数组变量，$\_COOKIE 完成对cookie数据获取。

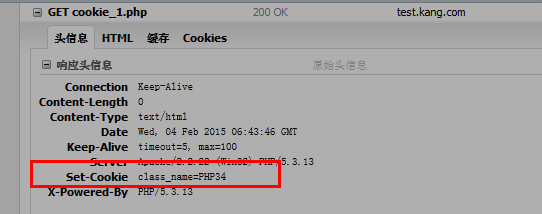
PHP核心，在初始化阶段，会将所有请求的cookie数据，整理到$\_COOKIE变量中，供PHP脚本所使用！

其中 每个$\_COOKIE中的元素，对应一个cookie变量，元素的键就是cookie的KEY，元素的值，就是cookie的Value。



## 通过firebug的网络面板监视cookie的传输情况

设置cookie：在响应数据区，会出现set-cookie指令，用来服务器告知浏览器需要设置的cookie信息。



浏览器请求时：如果存在则在请求数据中携带：



## Cookie的特征

### 有效期

默认：称之为临时cookie，也叫会话cookie。

会话周期结束（浏览器关闭）。

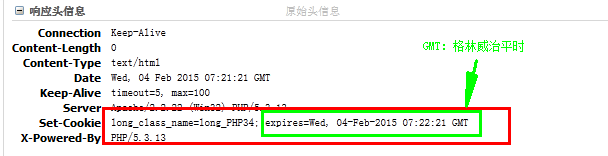
支持设置有效期：

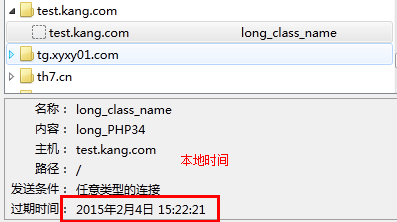
Setcookie()的第三个参数可以对有效期进行设置，有效期采用一个时间戳进行表示！



是谁判定cookie失效了呢？

浏览器，服务器在设置cookie时，会同时告知浏览器该cookie的有效期。





浏览器在向服务器发送请求时，会检测cookie的是否有效，只有没有过有效期的cookie数据请求时，才会携带。

语法：

特殊的有效期值：

1. 默认的，表示会话周期

Time()-1，删除cookie的标准做法。setCookie(‘KEY’,‘’, time()-1)

PHP\_INT\_MAX: 逻辑上表示永久有效的cookie。

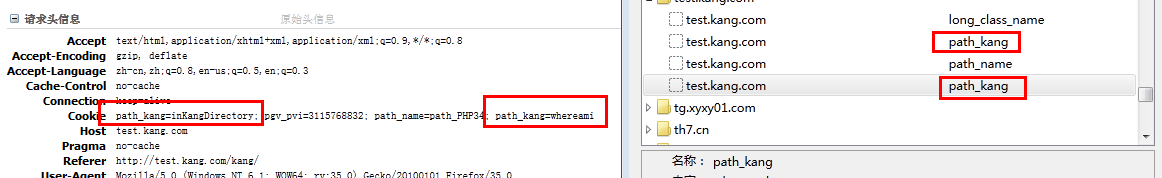
### 有效路径

默认：

Cookie在当前路径及其后代路径有效。

Tip：路径，不是代码所在文件的本地磁盘路径，而是URL请求的路径关系。

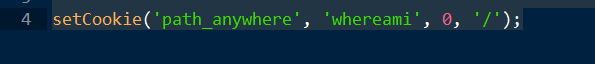
Tip：不同路径，下同名的cookie可以同时存储于浏览器端。



不过是，浏览器发出请求时，会先查找当前目录内有效的cookie，再向上查找，将所有有效的，都携带到服务器端，服务器在形成$\_COOKIE时，会出现重写效果，先出现的保留！

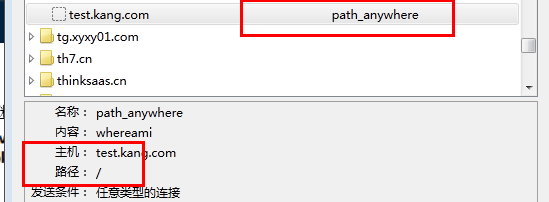
可以通过setCookie()的第四个参数进行修改：

通常设置为/表示，站点根目录有效。也就是整站有效！



Tip：有效路径的判断也是在浏览器端。服务器会将有效路径设置时，告知浏览器：





语法：

‘’ 默认值。

### 有效域

默认：

Cookie仅仅在当前域下有效。

可以通过设置，使cookie的有效域，扩展到某个一级域名下的所有子域；

Test.kang.com

Shop.kang.com

.kang.com 一级域名

Test.kang.com二级域名

第五个参数可以做到：



### 是否仅安全传输

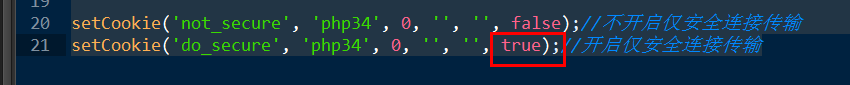
https://

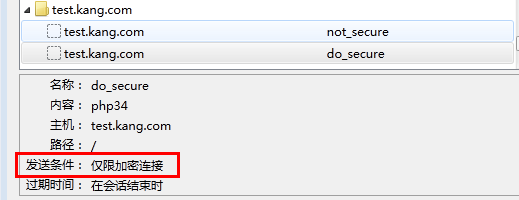
加密的http协议！

默认：

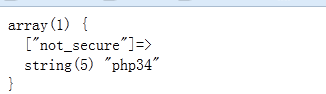
Cookie，不论浏览器发出的是http还是https都会将有效的cookie携带的服务器端。

如果将 第 6 个参数设置为true，表示 激活仅安全连接传输，此时浏览器在向服务器发出请求时，如果请求协议为http，就不会向服务器发送这些设置为仅安全连接传输的cookie数据。





此时发出http非安全连接：仅仅not\_secure 被传输了！



### HTTPONLY

默认：

浏览器存储的cookie是可以被其他脚本所处理的

通过第7个参数，设置为HTTPONLY特性，表示仅仅在http请求中使用！

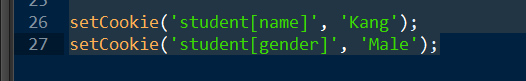


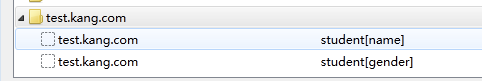
建议，该属性设置为true。

## 语法

Cookie值，仅仅支持字符串类型。

Cookie键，可以写成下标数组形式。



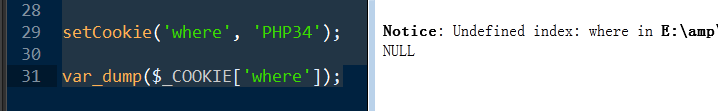


使用时，被整理成数组而已：

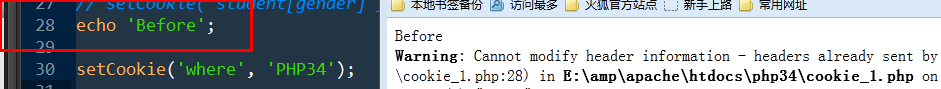


$\_COOKIE仅仅存储是浏览器请求时携带的cookie。

当前脚本周期内，设置的cookie，不能在$\_COOKIE中存在的。



类似于header()，setCookie()前也不能存在任何的输出。



# SESSION

## 基本原理

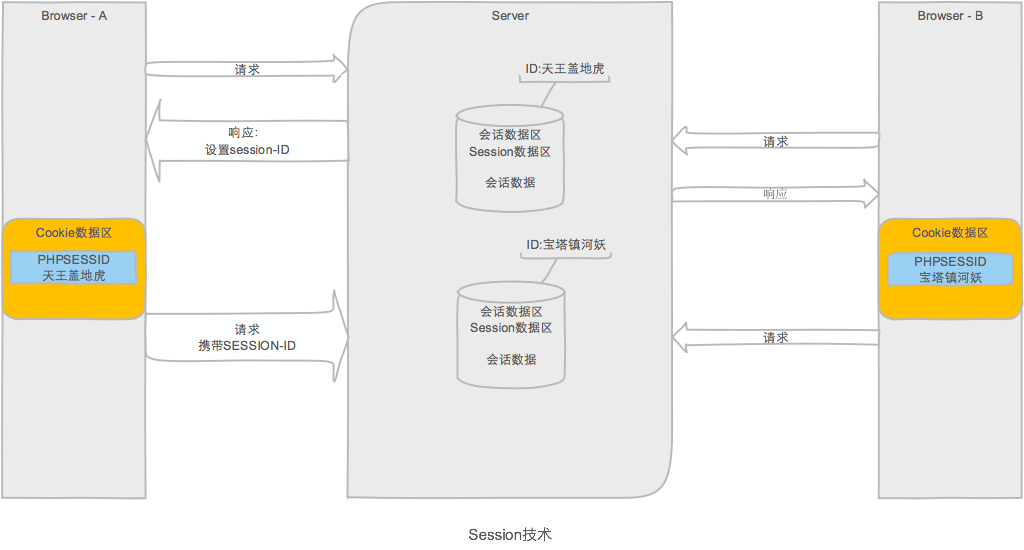
会话数据原文存储于浏览器端，原始数据安全性较低。

如果cookie数据量较大，由于每次请求都要携带，增加带宽使用。

使用session技术来实现：

将会话数据，存储于服务器端！同时使会话数据可以区分浏览器！

为每个会话数据建立独立的会话数据区（来存储当前会话的全部数据），每个会话数据区存在唯一的标志，同时浏览器端存储该唯一标志，做配对使用！



## 基本操作

### 开启session机制

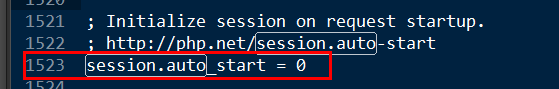
函数可以开启：

Session\_start();



也可以通过php.ini设置为自动开启：

Session.auto\_start

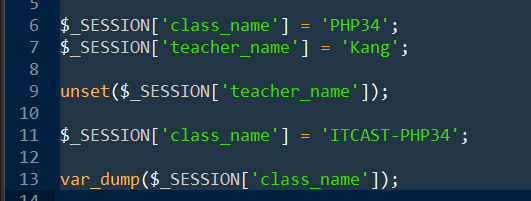


### 利用$\_SESSION操作session数据

增，删，改，查，都是使用$\_SESSION完成。

每个元素，就是session数据。

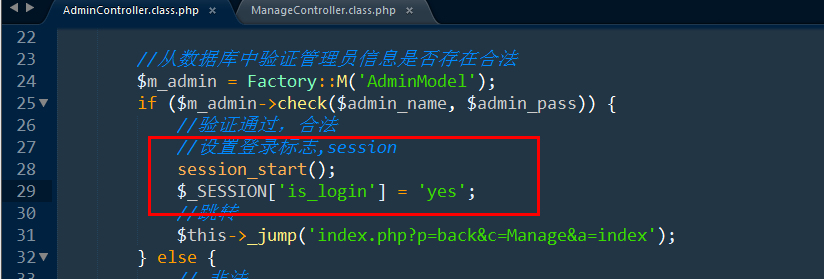
就像操作普通数组一样，操作$\_SESSION数组，就可以完成对session数据的操作！



## Session完成登录状态的存储

分配：

Back/AdminController->checkAction();



判断：

Back/ManageController->indexAction()

