# 总结

## 执行流程

## 语法问题

一：为什么 当前平台 和 路径 应该存储到常量中？

常量，保证在 一个脚本周期内不会被有意或无意的更改。

而，当前平台，和 对应的路径，都是在脚本执行周期内，不会更改。为了保证不会被更改，才选择使用常量。

额外：如果存储到常量中，没有作用域的问题！

二：代码在哪里执行？

内存中执行。

三：那么 php文件中的代码 和 被（在内存中）执行的代码有什么关系？

Index.php php文件，是代码的载体（容器）。



四：加载：include，require，include\_once, require\_once，加载到哪里？

加载是个什么操作。

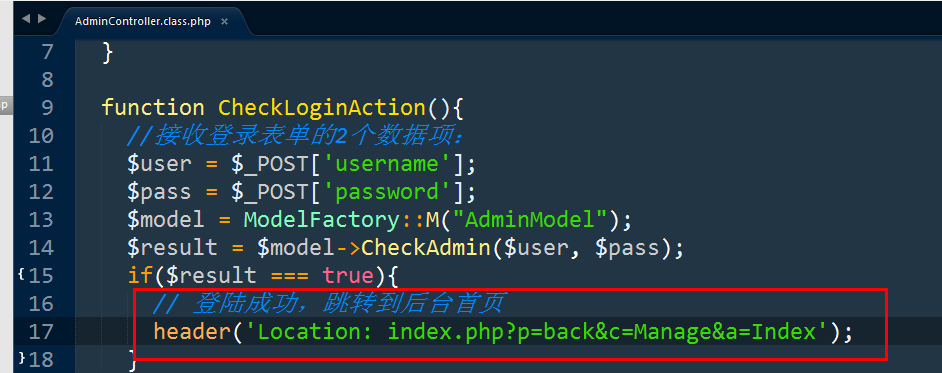
加载到内存中，对源代码进行 解析（分析，编译），和 执行。

# 后台首页[功能]

分析



补充后的 AdminController->CheckLoginAction()



## 控制器-动作

增加控制器类文件：

Application/back/controllers/ManageController.class.php



Tip: public修饰符，最好不要省略。

## 模型

不需要

## 视图

不需要

# 登陆标识的判断

## 分析

流程：



如何判定是否具有登陆标识？

在提供了正确的用户和密码后，分配一个登陆标识。后续的请求携带该标识过来，就可以做出判断了。



如何实现？

登录标识，是什么？就是个特殊的数据而已！

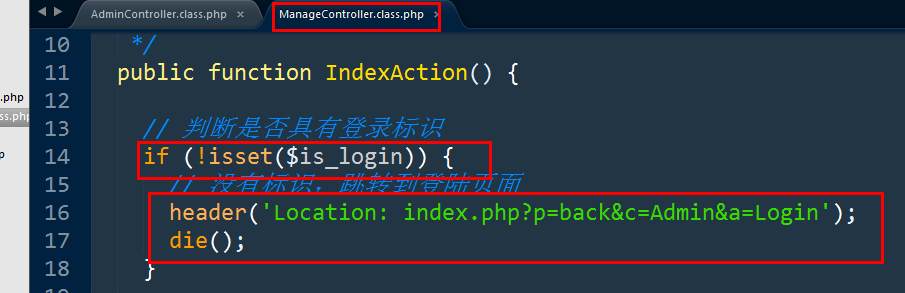
## 该数据如何存储？

变量，常量，数据库，文件，static

### 变量实验

分配标识：AdminController->CheckLoginAction();



判断标识：ManageController->IndexAction(); 

结果：不行。

原因：变量为脚本周期，（跳转时当前周期结束），变量就销毁了。无论如何也不能传输到下个周期（indexAction）. 那 常量呢？static $var呢？

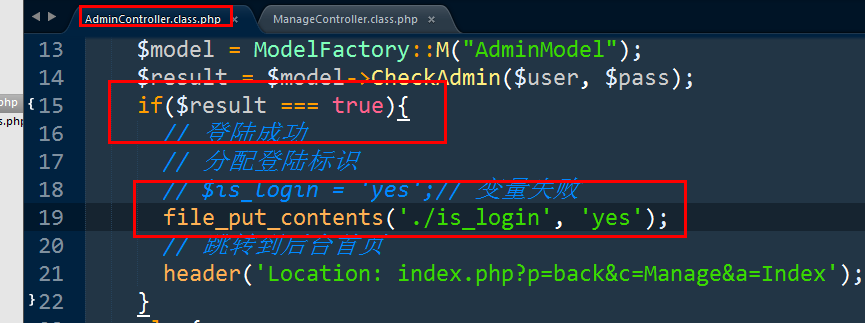
结论：存储的登录标识，不能在脚本周期结束时消失，可以持久性存在一段时间。

### 文件为例

可以持久存在。

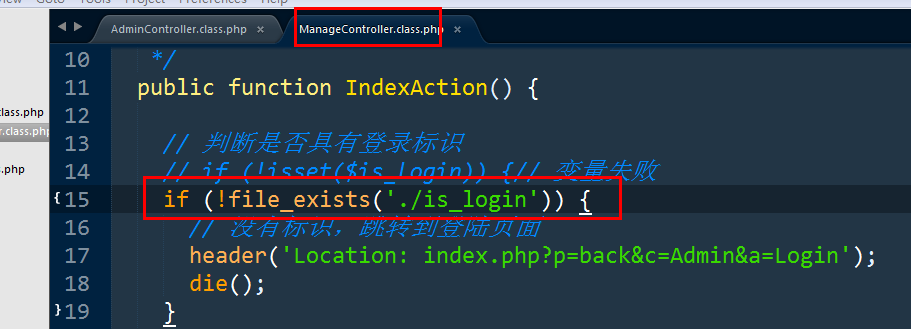
分配标识：AdminController->CheckLoginAction();

File\_put\_contents(),写入内容到文件，如果文件不存在，可以创建！



判断标识：ManageController->IndexAction();

File\_exists(),判断文件是否存在。



结果：失败

原因：不能区分开浏览器。数据库呢？

结论：需要一个可以区分开浏览器的数据。

### 总结

登陆标识的存储：需要在同一个浏览器的多次请求周期内，持久性存在的数据！

应该使用：会话技术来实现。（cookie和session）

# 会话技术

在同一台浏览器，的多次请求周期内，持久性存储数据一种技术！

基础：HTTP的无状态性。

B/S 架构基于HTTP协议进行数据交互。

HTTP请求/响应，是独立的，每次请求响应周期都是完全独立的。

会话：想办法，将数据可以在多次请求周期内，传递存储下来。

两种实现：

Cookie（基础）：Session：

# COOKIE技术

## 基本原理

允许 服务器端程序（PHP） 在 浏览器上存储数据 的一种技术。

浏览器存储数据的技术，浏览器技术。

由服务器脚本（PHP）决定存储内容。

该数据存储好之后，典型的特征：在浏览器向该服务器发出请求时，会携带该服务器所存储的数据。

以上就是COOKIE技术。



## 基本操作

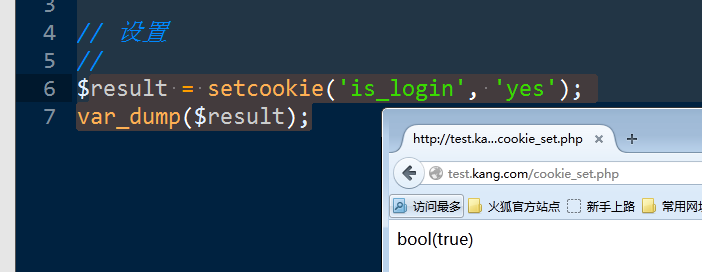
PHP基本操作

浏览器如何存储和如何携带，浏览器自身就实现了！和PHP没关系。

### 设置

函数：

setcookie(键, 值)



C通过浏览器查看到COOKIE数据：



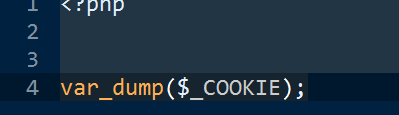
数据是在响应阶段被传输到浏览器端的：

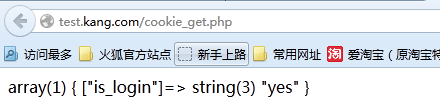


Tip：增加，修改，删除，都使用该函数setcookie()完成！

### 获取

当PHP核心程序，接受到浏览器携带的COOKIE数据时，整理到超全局变量: $\_COOKIE中。





每个元素，就是一个COOKIE数据，元素键，COOKIE名，元素值，COOKIE值！

获取的一定是浏览器请求时携带的：

参考浏览器的请求数据可见：

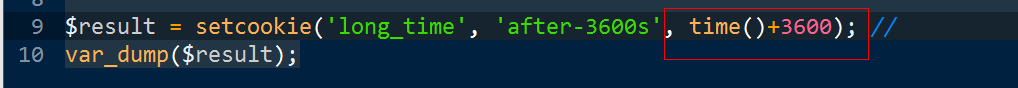


## 属性

### 有效期（重要）

默认：浏览器关闭（会话周期结束）时，COOKIE失效。也成之为：会话COOKIE（临时COOKIE）

同时：允许设置COOKIE的有效期，语法上，使用setcookie的第三个参数进行设置，通过unix时间戳（从1970年到现在的秒数，PHP函数time()来获取当前时间戳。）来设置COOKIE的有效期。



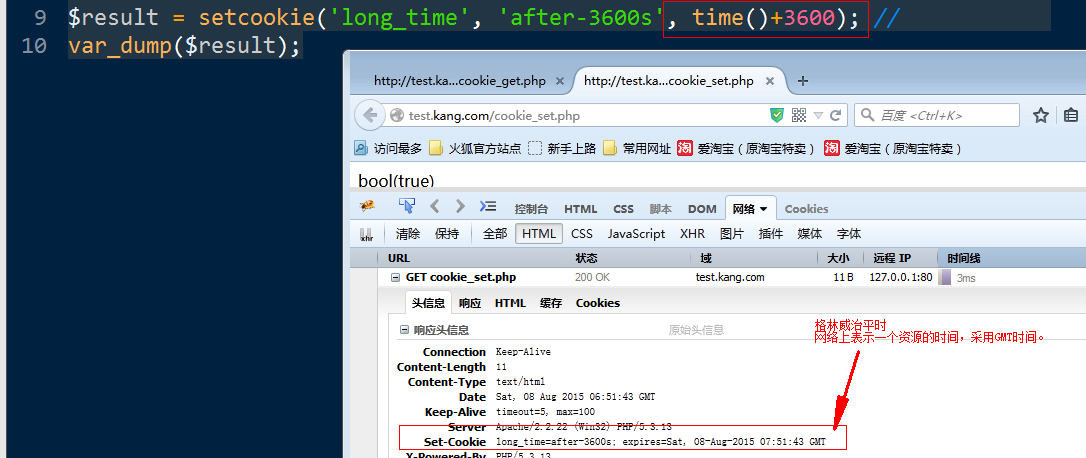
原理：

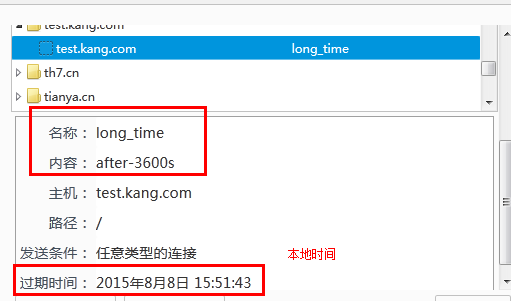
谁来判断该COOKIE是否过期？(是否在有效期内)

浏览器 OR 服务器？

是浏览器判断该COOKIE是否过期的。

在设置COOKIE时，同时将COOKIE的有效期，告知（传输）到浏览器端。





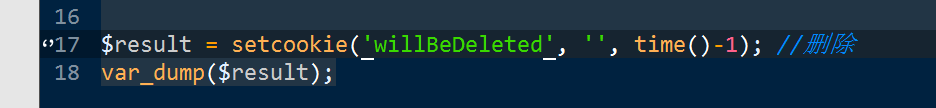
特殊值：

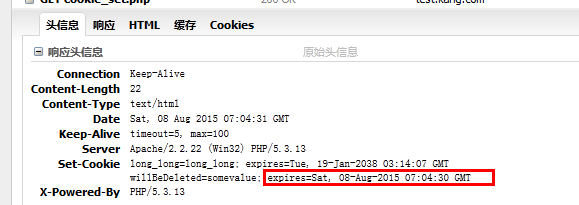
0：默认值，表示临时COOKIE。

PHP\_INT\_MAX常量值：PHP所能表示的最大整型（时间戳也是个整型，也是PHP能表示的最大时间戳）。

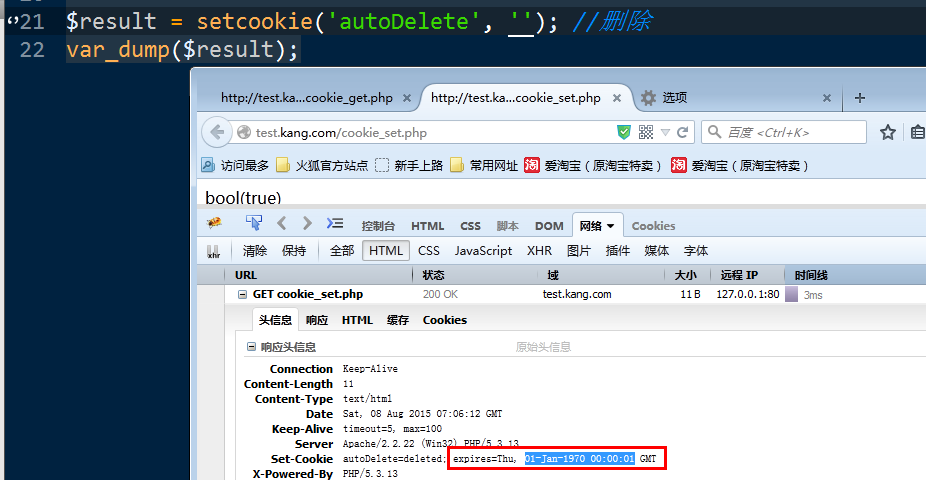


Time()-1: 删除COOKIE通用做法，强制设为过期。





额外的：PHP的setcookie支持将第二个参数设置为空字符串的方式，表示删除。（快捷语法语法糖）



### 有效路径（了解，使用较少）

默认的：

COOKIE：在当前目录及其后代目录有效。

语法使用空字符串表示默认值！

额外的：

通过setcookie函数的第四个参数，进行设置。通常如果需要设置，就是将COOKIE设置为 / 根目录下有效，表示整站有效。



### 有效域（使用频率较高）

默认的：

某个域名下设置的COOKIE，仅仅可以在当前域名下所使用！

额外：

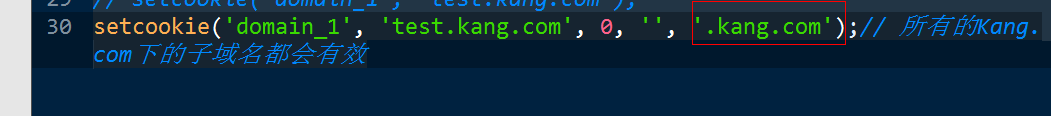
现在稍大点的网站，都会使用子域名（二级域名）的方式划分业务逻辑：

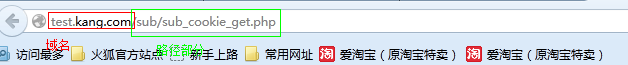
News.baidu.com. music.baidu.com， 就是baidu.com下的子域名（二级域名）

Test.kang.com, shop.kang.com，都是kang.com下的子域名。

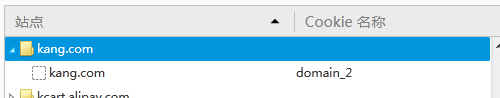
COOKIE，支持在一级域名内（所有的二级域名之间）进行COOKIE数据共享。

使用 setcookie的第5个参数进行设置：





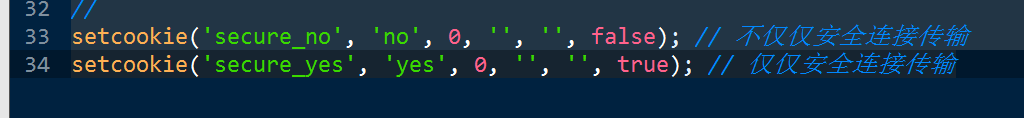
此时，浏览器端，存储方式：

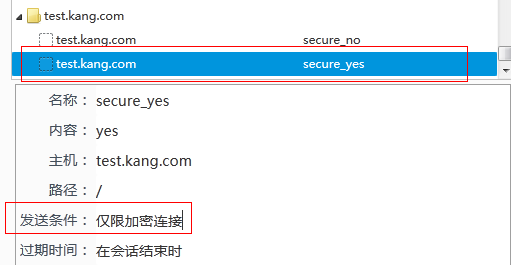


### 是否仅安全连接（https）传输

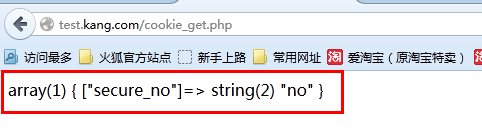
告知浏览器，我们所设置的cookie，是否仅仅在https这个协议下，才会被发送到浏览器端！

是用 参数 第六个完成设置：默认值false，表示都可以（http和https）。





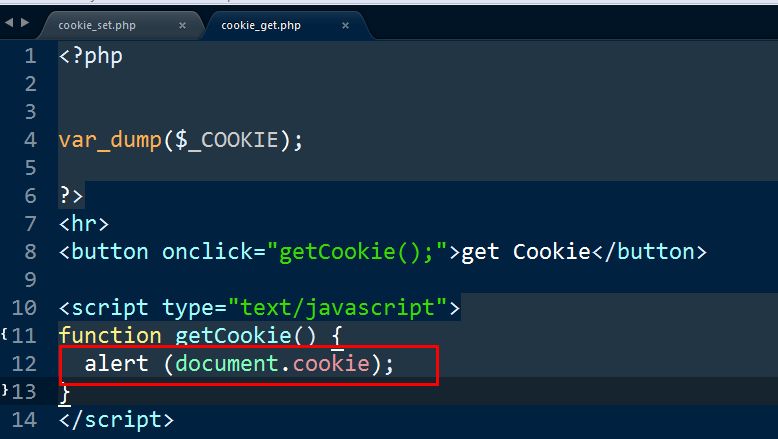
请求时：只有非安全连接才能传输的才会被发送到服务器端！

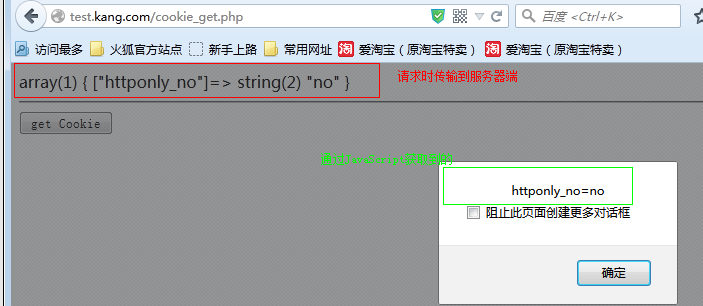


### HTTPONLY

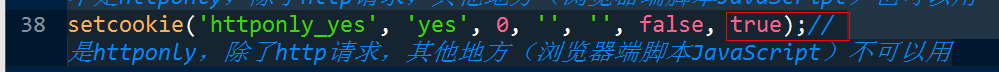
所设置好的COOKIE，是否仅仅在http请求时所使用。

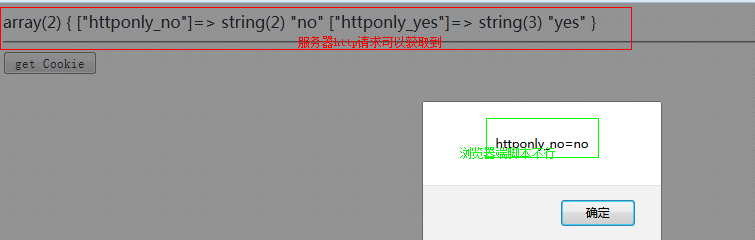
COOKIE还可以通过浏览器端的脚本（javascript）获取到。





如果需要禁止，浏览器端其他脚本使用该cookie，则使用第七个参数即可：





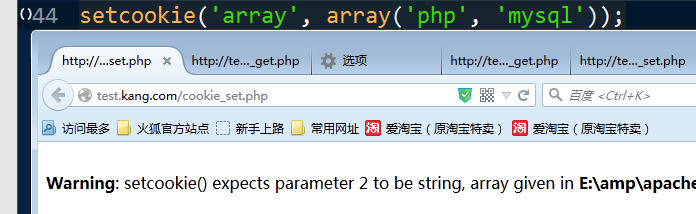
总结

Setcookie(键，值，有效期=0，有效路径=‘’，有效域名=‘’，是否仅安全连接传输=false，是否HTTPonly=false)

## 语法细节

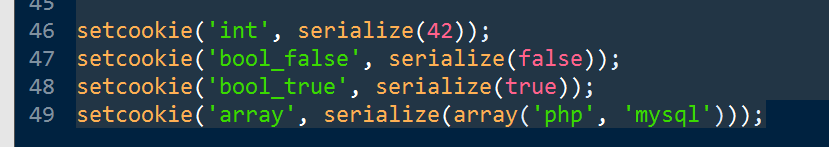
### COOKIE仅仅支持字符串类型数据

可以转换成字符串的数据会执行自动类型转换。

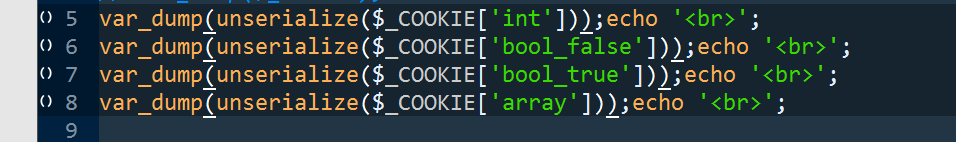


办法：

Serialize()序列化。将某种特殊的数据类型，转换成字符串。

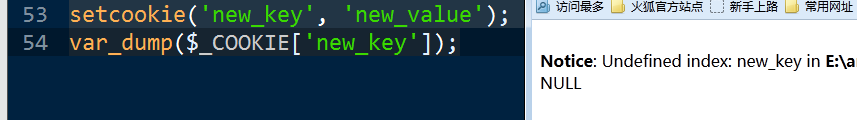


获取后，在反序列化unserialize()：

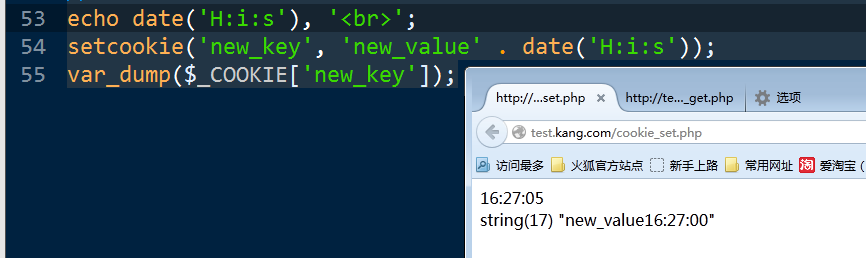


### $\_COOKIE,仅仅用来存储浏览器请求时携带的COOKIE数据

当前脚本周期setcookie所设置的COOKIE变量，是不会出现在$\_COOKIE中！



下载再请求：



## COOKIE的使用

购物车。在未登陆的情况下，用来记录购物车内商品。

多长时间免登陆。



搜索习惯。搜索历史

需要长时间的存储会话的数据，通常使用COOKIE完成。

# SESSION技术

## 基本概念

使用很多的会话技术。

COOKIE的劣势：

会话数据的存储位置：浏览器端。

带来的问题：

会话数据的安全性。

请求（响应）传输的会话数据量大。浏览器都会限制COOKIE的大小和数量。

Session技术：

会话数据的存储位置：服务器端。

那么如何实现，区分不同的浏览器来存储会话数据。

实现方式如下：

在 服务器端，建立很多的会话数据区（session数据区）

为 每个session会话数据区分配唯一标识

将该唯一标识，分配给对应会话浏览器



因此：session技术基于COOKIE技术！

## 基本操作

### 开启session机制

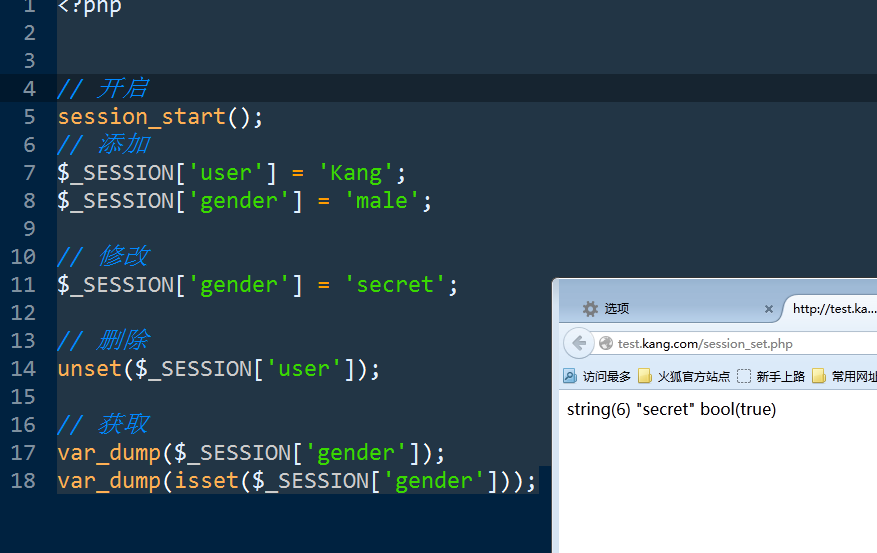
函数  
session\_start()

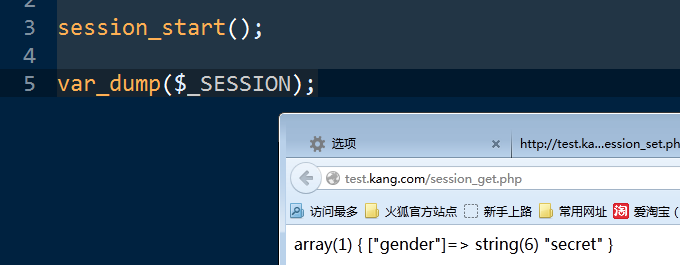
开启后，所有关于session机制（生成session\_id,）都由session机制（PHP提供的session相关功能）完成

### 操作session数据

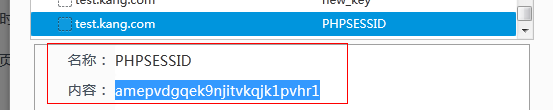
操作$\_SESSION超全局变量即可！

操作常规数组一样，操作$\_SESSION数组即可！





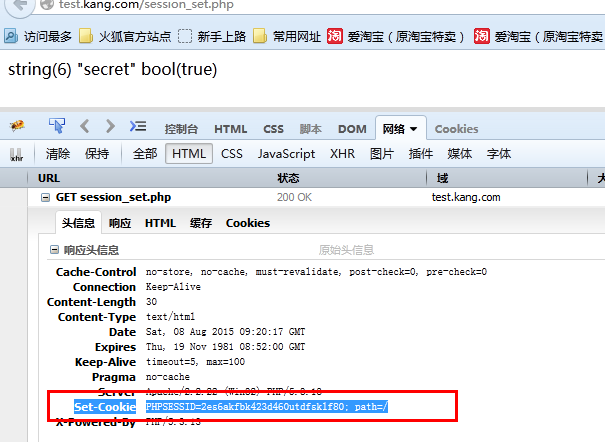
### 浏览器端存储session-ID

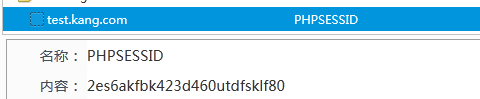


可见，就是一个普通的COOKIE变量，只是名字特别的点：PHPSESSID。值为了保证唯一性：

在：浏览器没有session-id时：浏览器请求服务器端，服务器php开启了session机制，PHP就会生成一个session-ID，以COOKIE的形式设置在浏览器端：

如图：删除PHPSESSID这个COOKIE的情况下，请求：



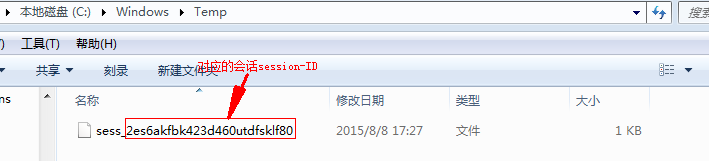


此时浏览器 存在该session-ID，下次请求时，携带该session-ID：

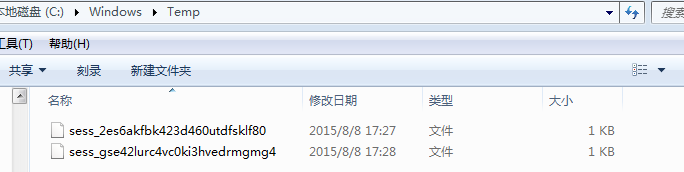


### 服务器端session会话数据区

默认，每个session会话数据区，就是一个独立的文件。存储于 服务器所在操作系统的临时目录中：



如果有多个会话，则会出现多个不同ID为文件名的文件：

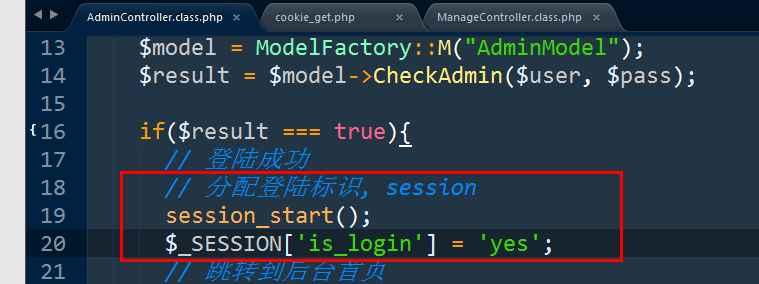


# Session实现登录标识

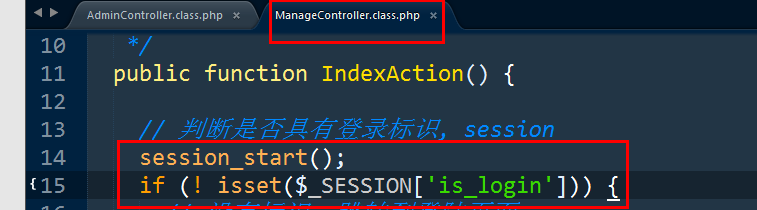
选择使用session的原因：

## 基本：使用 is\_login = yes 完成标志

分配标识：AdminController->CheckLoginAction();



判断标识：ManageController->IndexAction();



完成！

## 实际：使用管理员信息作为登录标识

便于程序中判断是谁登陆：

例如： 欢迎，某某某

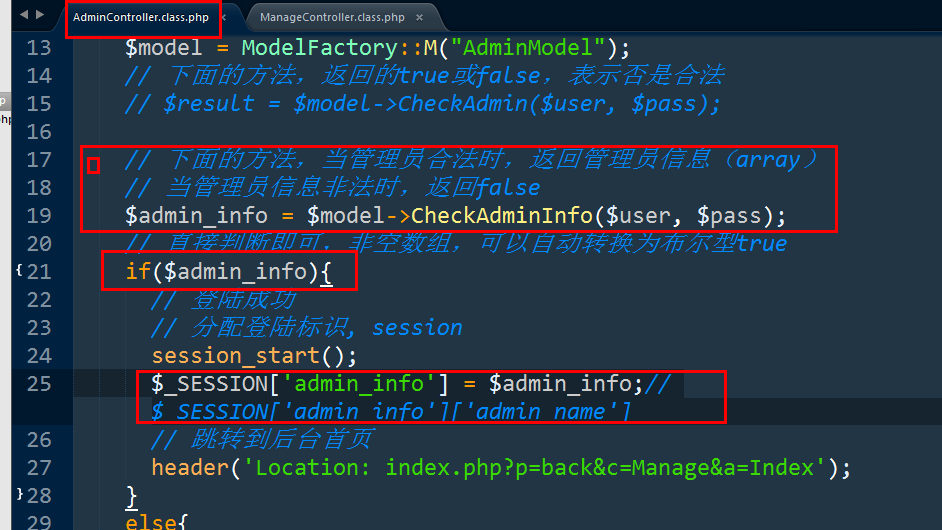
某个用户，对应不同的权限

### 分配标识：AdminController->CheckLoginAction();

#### 控制器：

AdminController->CheckLoginAction()

调用了升级的模型方法，获得管理员信息。



#### 模型：

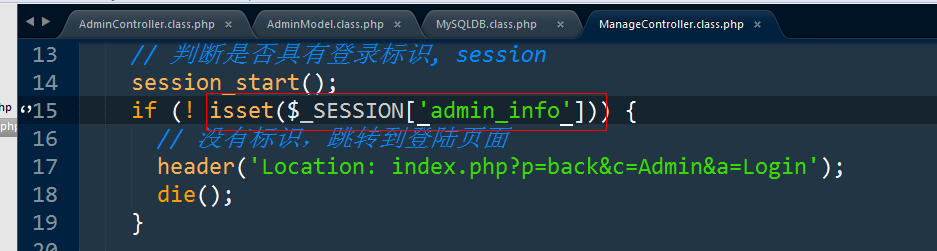
增加back/AdminModel的CheckAdminInfo()方法：



#### 视图：

### 判断标识：ManageController->IndexAction();

ManageController->IndexAction()



### 展示登陆管理员信息

ManageController->IndexAction()

