# SESSION

## 语法问题

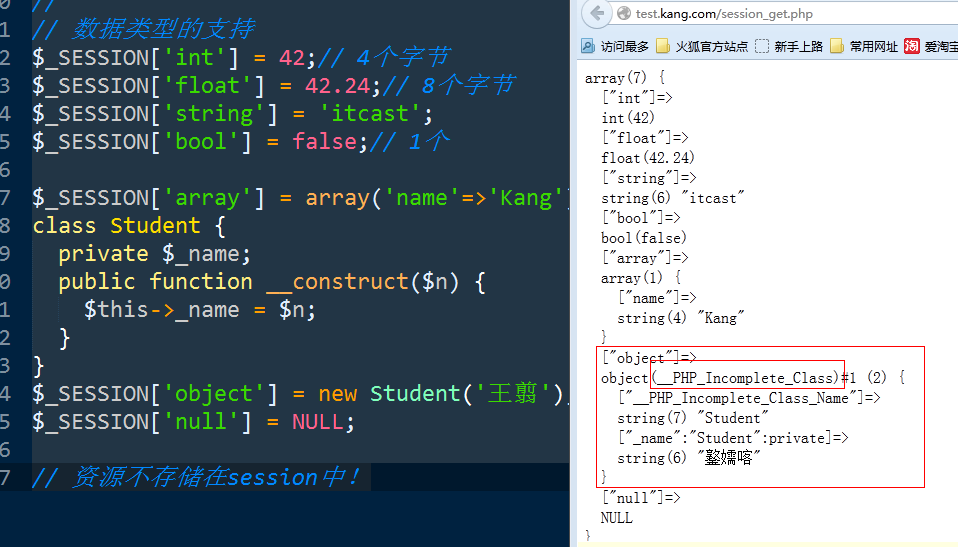
### 多数据类型的支持

字节Byte：1Byte=8bits。存储空间基本概念。最小的存储单位。

位bit：计算单位。

奸商：带宽 10Mbit。

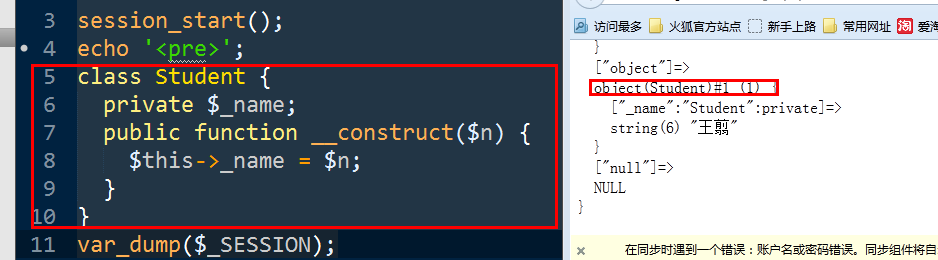
支持多种类型：



注意：对象型数据：

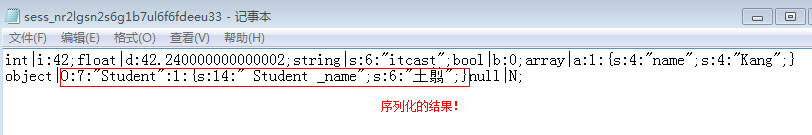
Unserialize()

在session中存储对象时，获取该对象时，需要找到对象对应的类：



Why？

数据在session会话数据区，采用序列化的方式进行存储。

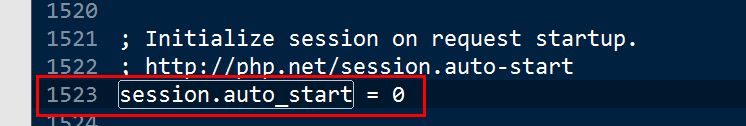


### 开启session

每次使用session，都必须要先开启

PHP支持自动开启session机制

配置 php.ini :



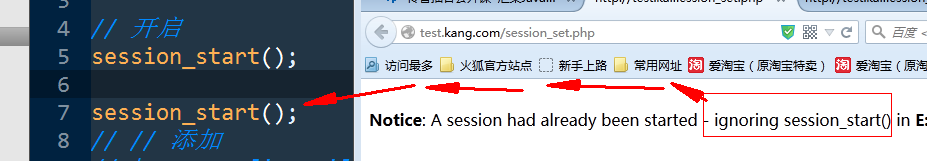
建议：不要自动开启！

重复开启：

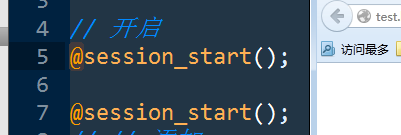
Session不能重复开启。

一旦重复开启，后边的开启会被 忽略！

但是，会触发一个 notice 级别的错误。



通常的解决方案，开启session时，增加@，错误屏蔽！



## Session属性

有效期：

默认关闭浏览器！

有效路径：/

默认整站有效

有效域名：

默认尽在当前域名下有效！

是否仅安全连接传输：

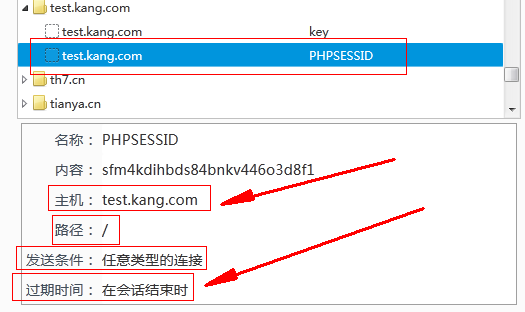
默认为非

是否HTTPONLY：

默认为非；

Session属性的来源？

COOKIE中存储session-ID的属性，决定了对应的session数据的属性。

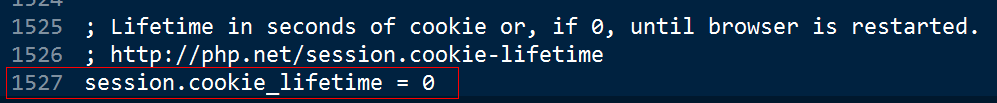


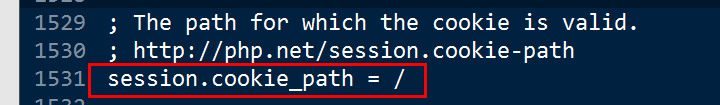
如何设置 session数据的属性？

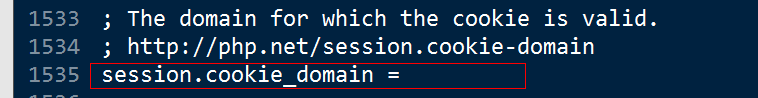
设置COOKIE中session-ID这个COOKIE变量属性即可！

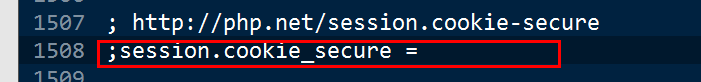
设置方案如下：

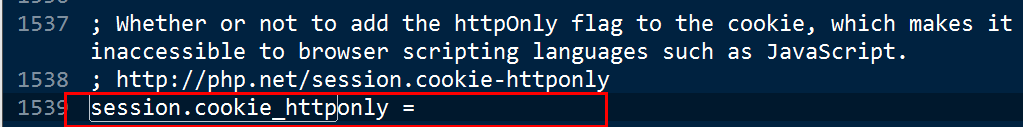
1. 方案一：配置变量php.ini:







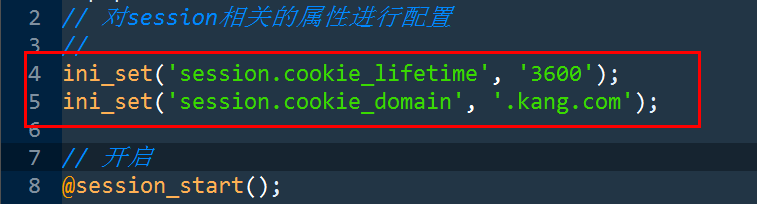


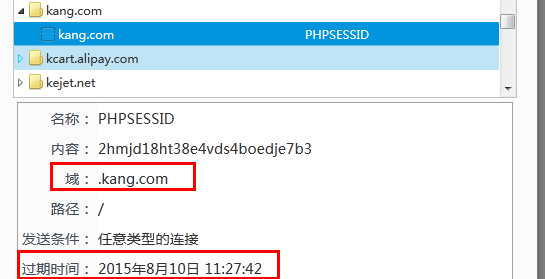


可以更改以上配置，apache httpd restart。

1. 方案二，在脚本中，开启session之前使用函数进行配置（建议）

Ini\_set(配置项，值);用于设置某个PHP配置选项





Session\_set\_cookie\_params(有效期，有效路径，有效域，是否仅安全连接传输，是否HTTPONLY)

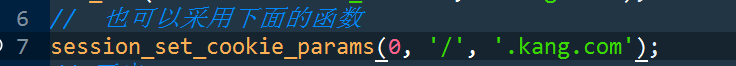


建议使用该方法，仅仅影响当前脚本周期。不影响其他项目！

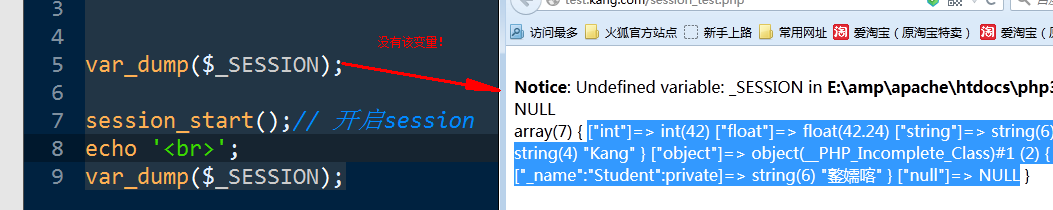
严重建议：

实际环境中，很少该session的有效期。经常改有效域名。





# Session数据区操作



# 重写session的存储机制

Session数据区

默认以 文件的形式存储与服务器操作系统临时目录中！

当 session数据区过多时，文件形式的存储，操作速度变慢。磁盘的读写（IO，input/output）开销是很大的。

实际项目中，都会采用其他的方式更快地存储session数据。典型的办法：数据库，内存。

以 数据库存储为例，讲解：session数据入库！

重写 与 session数据区直接的相关操作即可：

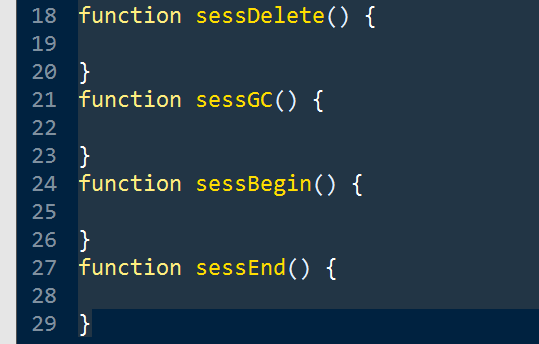
最基本的只有2个：读，写！

一：定义2个可以完成读和写的函数。

二：告知session机制，在需要读写时，使用用户自定义的读写函数完成。

## 定义2（其实共6个需要的相关函数）个可以完成读和写的函数





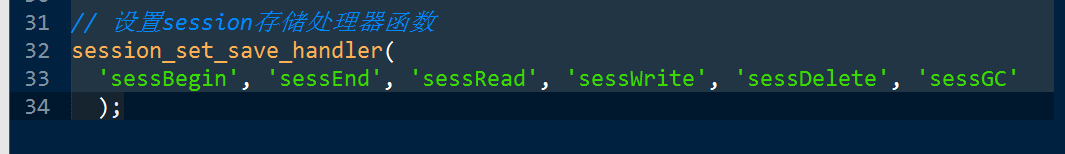
## 告知session机制，在需要读写时，使用用户自定义的读写函数完成

Session\_set\_save\_handler(

开始函数，结束函数，读函数，写函数，删除函数，GC函数

);

用来将用户自定义的函数，设置成session存储相关的函数。



Tip：以上的语法，仅仅是设置告知，不是调用以上6个函数，这六个函数，在session机制运行到某个时间点时，才会被调用！例如，咋开启session时，才需要调用sessRead()

## 常规使用session

开启session机制

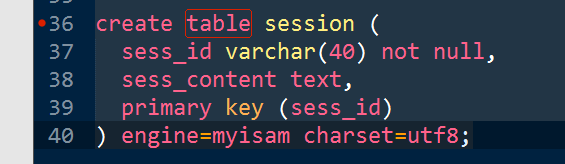
操作$\_SESSION



## 创建session表

该session中，每条记录，就是一个session数据区，相当于原来的一个session文件。

表结构：



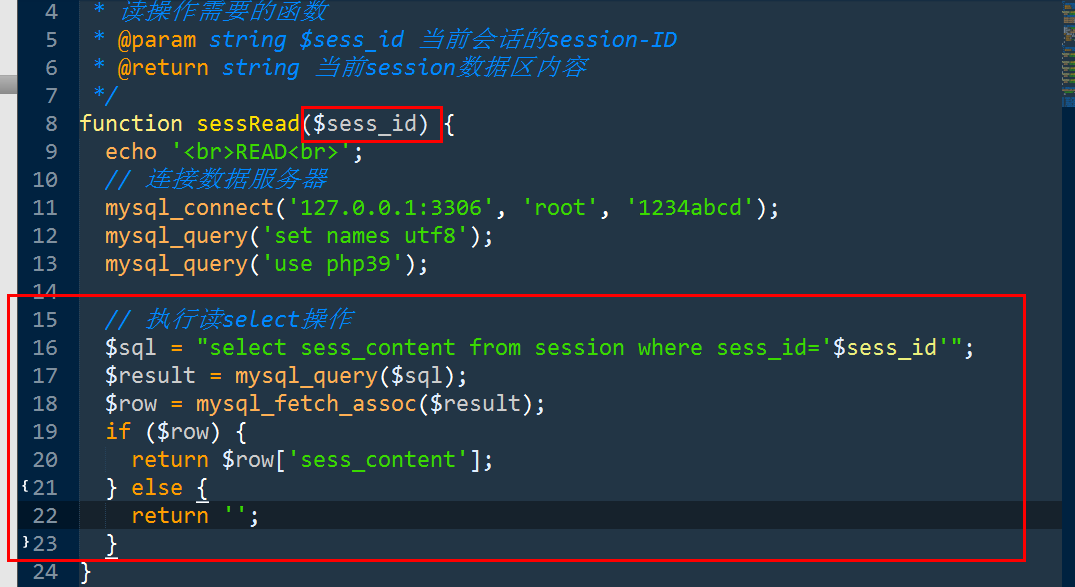
## 读操作：sessRead()

谁调用，谁传参！

在PHP的session机制调用该函数时，会将当前的session-ID作为参数传递到函数中：

因此，需要定一个形参，接受传递的session-ID 参数：

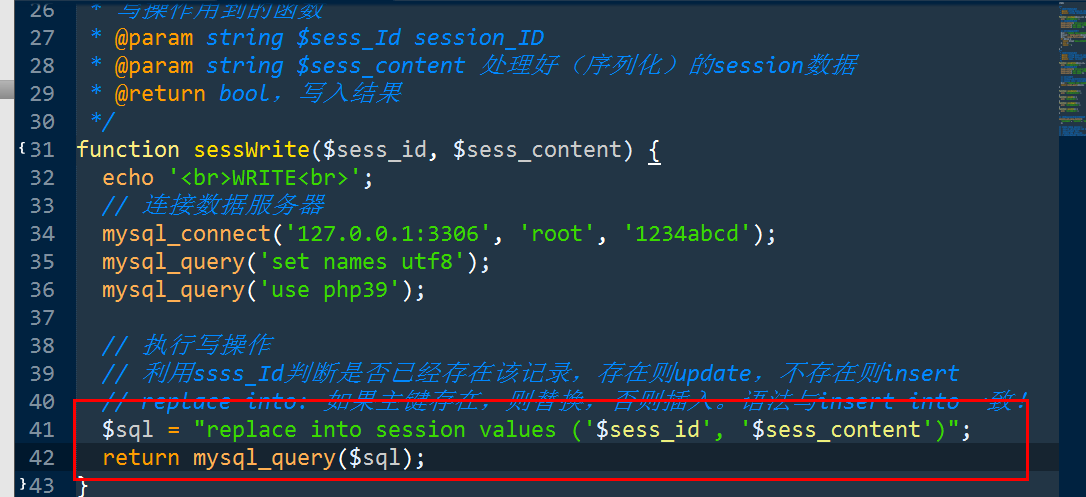
需要返回，读取到的session数据字符串。就是sess\_content字段的内容。如果没有读到，则返回空字符串即可，表示没有session数据。



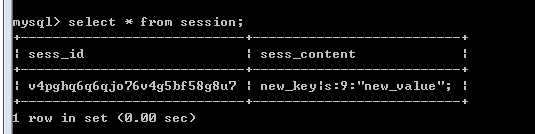
## 写操作：sessWrite()

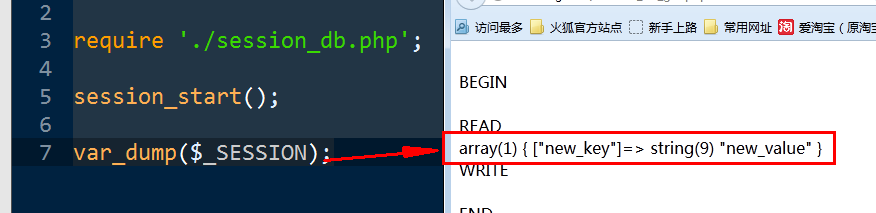
当PHPsession机制调用该函数执行写操作时，会将 当前session-ID和 需要写入的内容（序列化好的）传递到函数！

需要2个形参来接收：



测试：





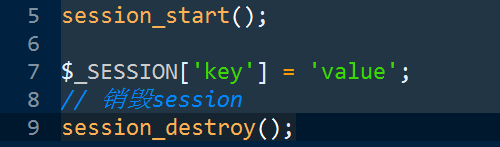
## 删除操作：sessDelete()

销毁session时。

执行了PHP函数：

Session\_destroy();

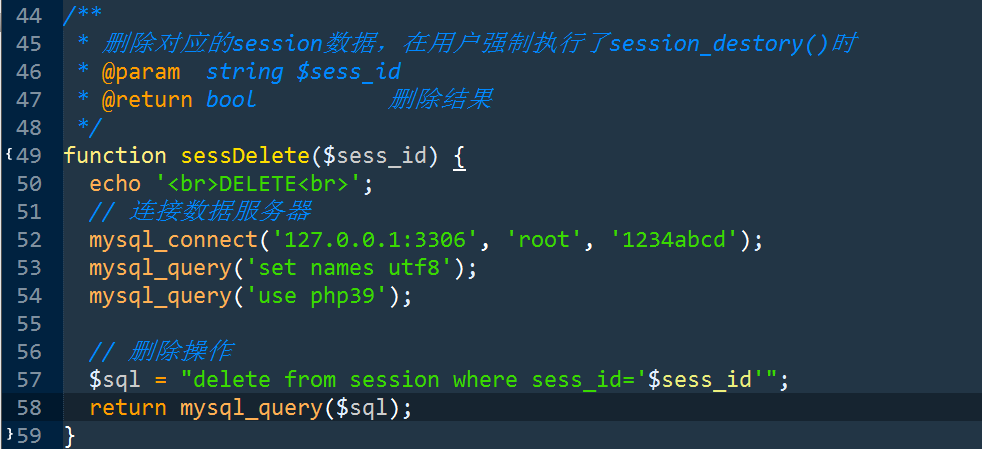
可以销毁session，删除对应的session数据区，同时关闭session机制！



由于需要删除session数据区，需要增加用于删除的方法：

PHP的session机制，在调用sessDelete时，会传递 当前session-ID作为参数：

需要定义形参来接收：



## 垃圾回收操作：sessGC()

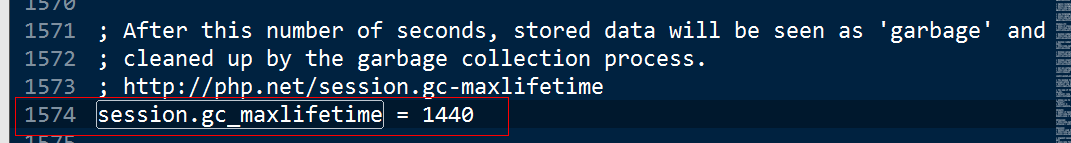
垃圾：

服务器上过时的session数据区。、

### 垃圾如何判定？

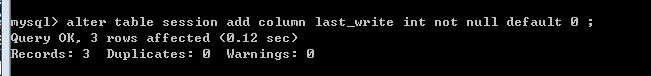
如果一个session数据区已经超过多久没有使用（最后一次写操作）了，就是被视为垃圾数据。

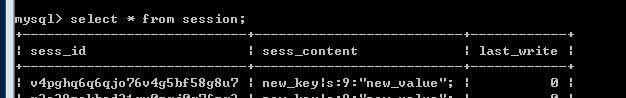
该时间临界点：默认1440s。可以被配置：



配合最后写入时间，就可以断定是否为垃圾啦。

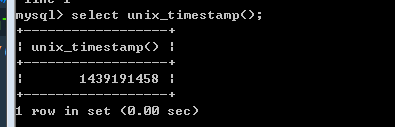
需要增加字段，记录最后写入时间。

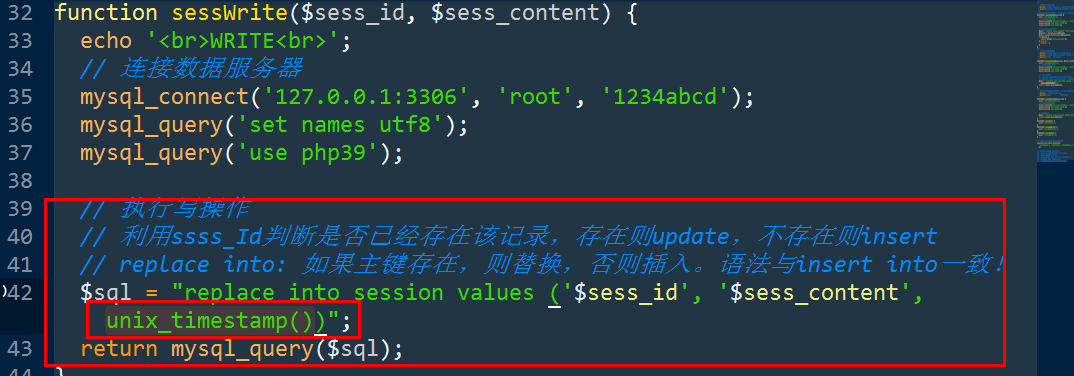




写入时，更新该字段：

sessWrite();





判断条件：过期啦。

Last\_write < 当前时间-1440

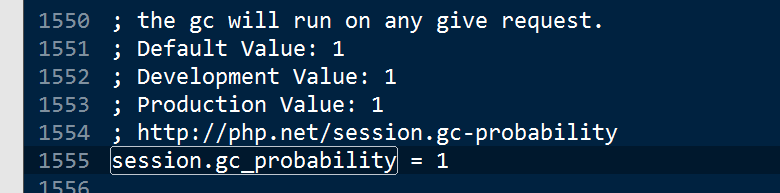
### 如何删除？

在 session\_start()过程中，开启session机制过程中：有几率地执行 垃圾回收操作。一旦执行，就会删除所有的过期的垃圾数据区。

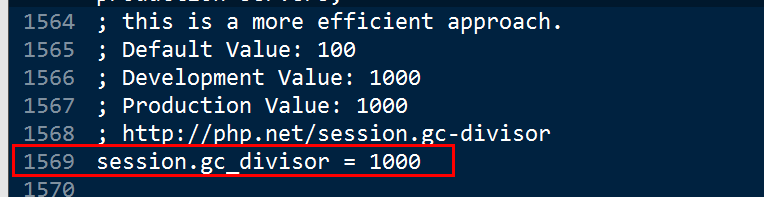
默认的概率为1/1000。

可以设置该几率：

可能性：

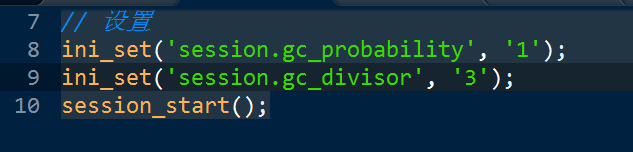


基数（除数）：



调整几率测试：

建议在脚本周期调整，使用函数ini\_set(),在开启session机制前完成：



实现 sessGC()

PHP的session机制将 最大有效期作为参数，传递过来！



## 开始操作sessBegin():

初始化工作

可保证在第一个执行。将初始代码，在sessBegin完成：

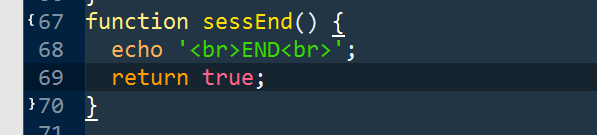
例如初始化数据库连接：



## 结尾操作sessEnd()：

收尾性工作

Return true;



## 语法细节

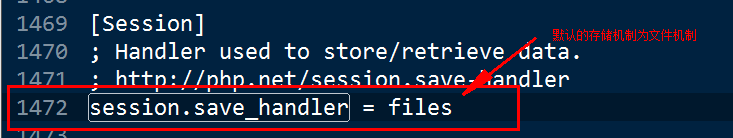
### 先设置在开启session机制

Session\_set\_save\_handler()先于session\_start()被调用。

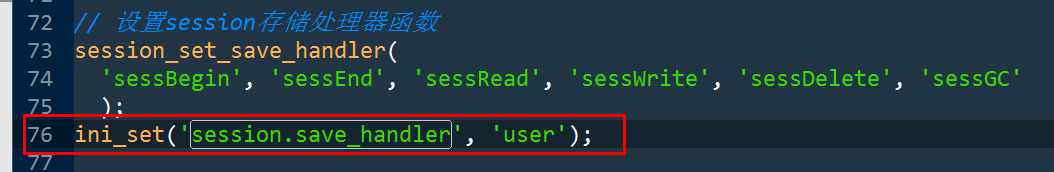
不要自动开启session！php.ini: session.auto\_start = 0

### PHP配置项：session.save\_handler

PHP所使用的存储机制：



建议，将以上配置改为 user： 表示用户自定义！



# 会话技术总结

## 相关相关配置

Session.save\_handler 存储处理器： files|user

Session.save\_path 存储地址。

Session.cookie\_XXX （lifetime，path，domain，secure，httponly）存储session-ID这个COOKIE变量的属性

Session.gc\_maxlifetime

Session.gc\_probability

Session.gc\_divisor

## Session，COOKIE联系和区别？

联系

都是会话技术。

Session基于COOKIE，session-ID存储于COOKIE中！

区别：

Cookie session

存储位置 浏览器端 服务器端

安全性 低 高

大小限制 有 没有

数据类型 字符串 全部

有效期使用 长时间存储 几乎不做持久化

## Session如何持久化？[理论]

Session-ID要持久化：session\_set\_cookie\_params(3600);

服务器session数据区有效期修改：ini\_set(‘session.gc\_maxlifetime’, 3600);

## 浏览器禁用COOKIE，session是否可用？[理论]

COOKIE被禁用，session-Id不能存储和传输。

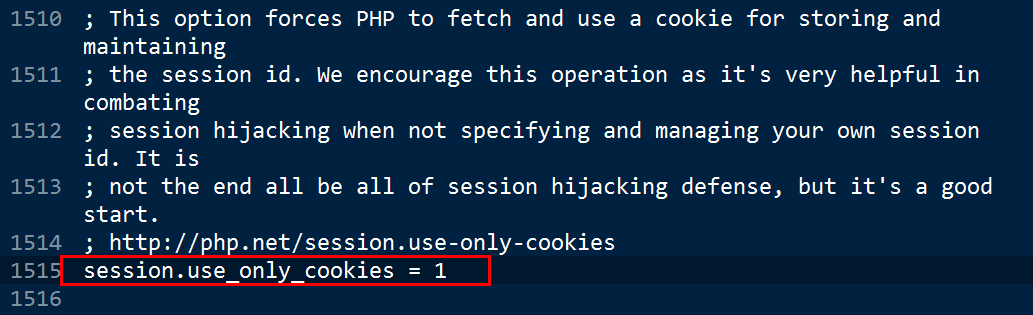
不可用！

理论上的解决方案：

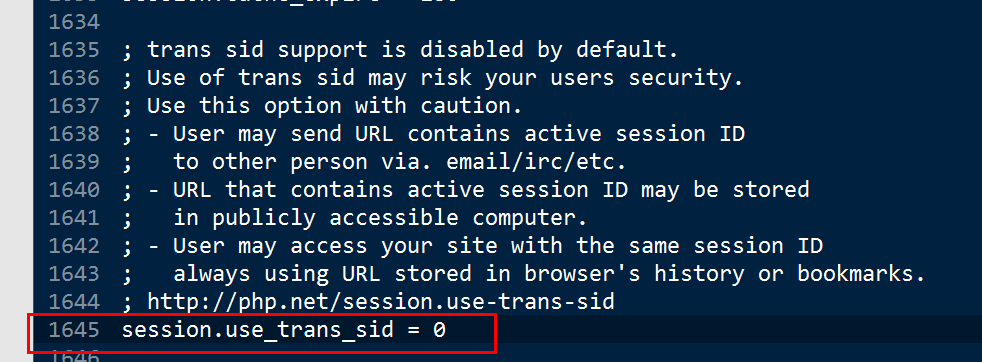
通过 URL， 或者 POST数据数据向服务器端，每次传输session-ID！

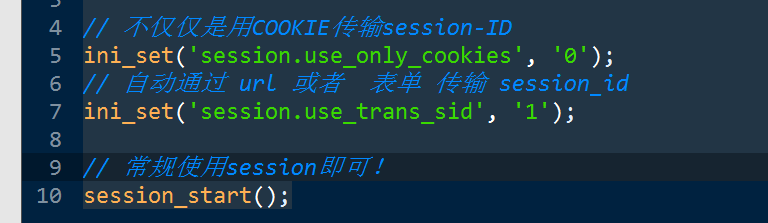
例如下面的配置：php.ini

Session是否仅仅是用COOKIE完成传输session-ID：



是否通过其他方式自动传输session-ID

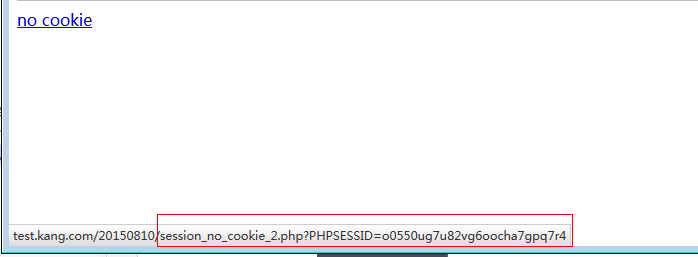




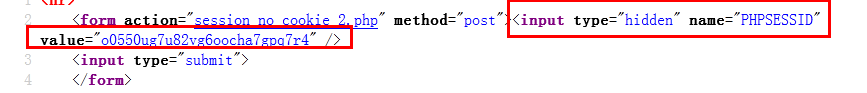
测试：



链接get：



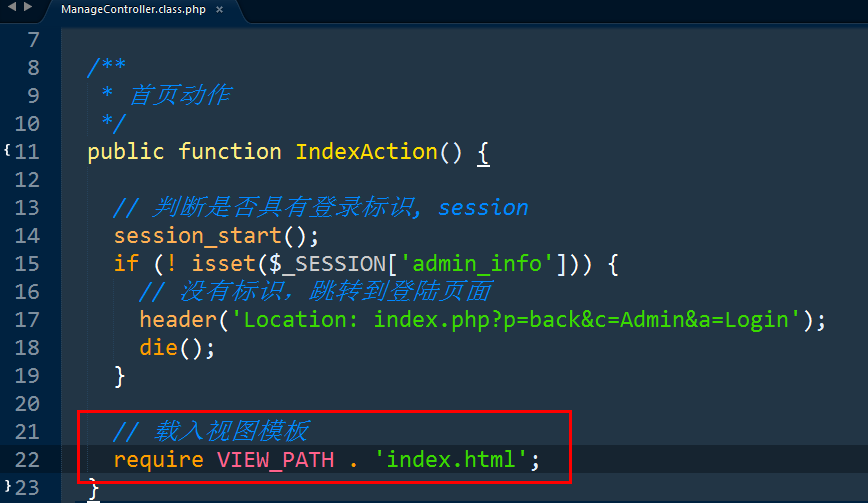
表单post：



# 后台首页[功能]

## 控制器

Back/ManageController->indexAction();



## 模型

不需要！

## 视图

新建模板：

Application/back/view/index.html

采用框架集布局！

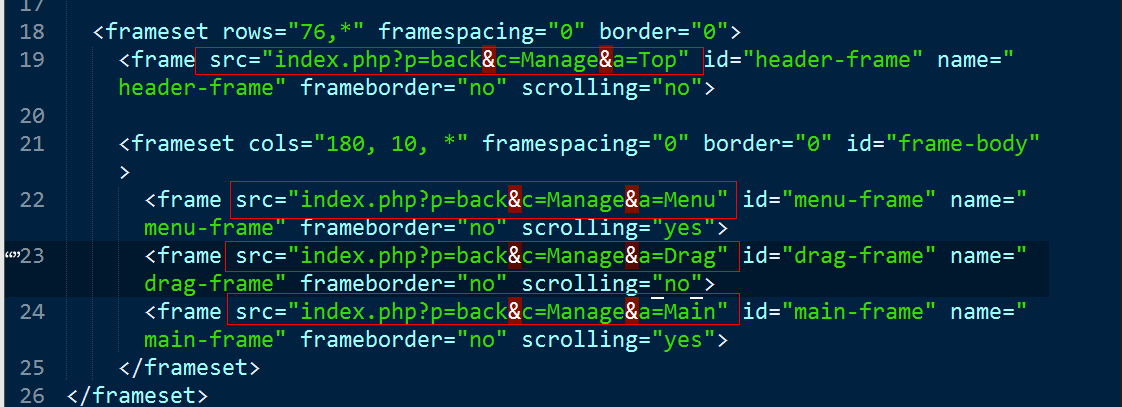


同时提供 4 个动作，完成每个frame（框架）部分的实现：

由 动作 完成，不要直接写模板地址！

项目中任何的功能，都是由动作提供的！

如下：

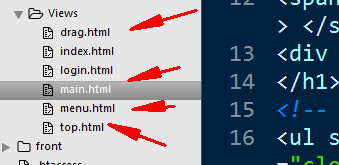


## 提供4个[功能]

Back/ManageController:



## 提供4个模板



# 后台集中的登陆验证[功能]

后台，几乎所有的动作需要先登陆再执行！不仅仅是首页indexACtion().

在何处完成该登陆验证？

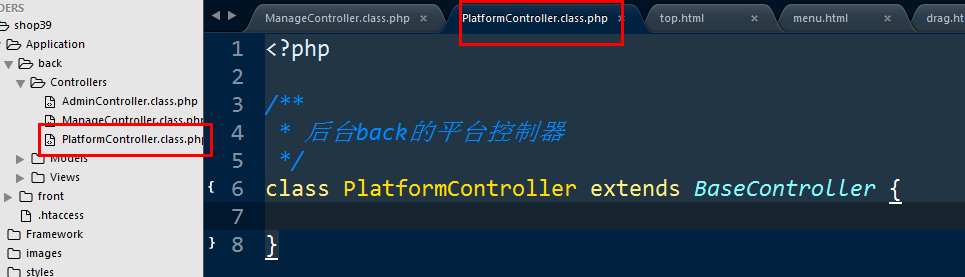
在后台公共的控制器位置！

使用平台控制器完成：



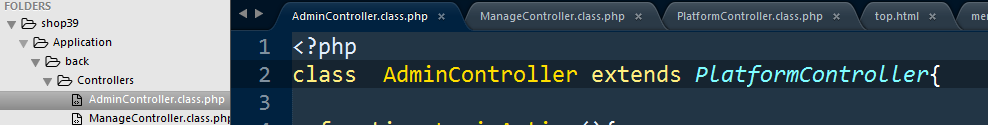
编码实现：

## 继承链结构



具体功能控制器，继承自 平台控制器：

后台的所有控制器。





## 后台实现集中校验

Back/controllers/PlatformController.class.php

定义：



何时调用？

业务逻辑（我们要干什么）：在执行后台几乎所有的动作之前，都需要先调用该方法，校验是否登陆。

可见，在实例化后台控制器对象对象时，完成该方法的调用，完成是否登陆的校验。在构造方法中完成。

后台平台控制器构造方法中：

Back/controllers/PlatformController:



Tip:别忘了调用父类被重写构造方法

测试结果：

请求后台p=back&c=Manage&a=Index



可以肯定：已经完成校验并跳转到登陆动作：

但是，出现了错误，不能显示登陆界面。

原因是什么？

在执行Admin/LoginAction()时，也需要先实例化AdminController，此时又会调用构造方法同时完成登陆校验。未登录，又跳转到LoginAction(). 没完没了.

如何解决？

将后台中不需要校验的动作，以特例的形式跳过即可！

## 特例实现

一：在\_check()中，先判断为是否特例，如果为特例，不予校验



二：获取当前的动作和控制器

在入口文件请求分发器index.php中，完成对 c和a的处理，存储在常量中，在一个脚本周期内不会（不应该）被修改

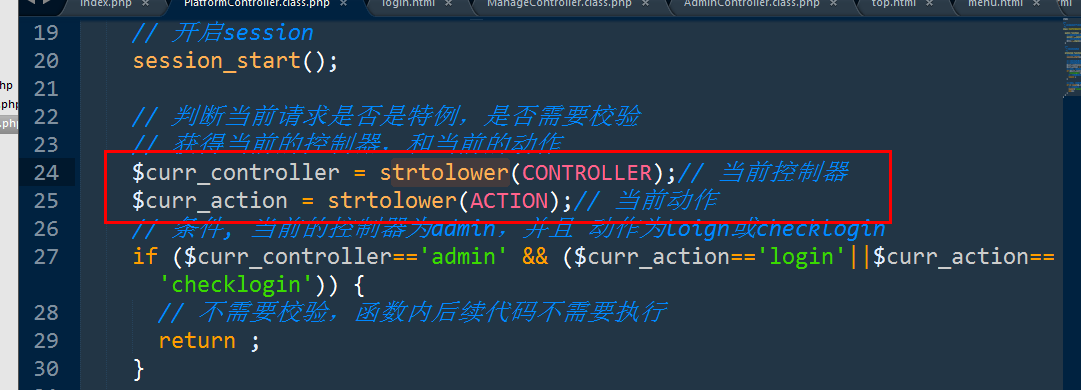
Index.php



后台平台控制器中，直接使用即可：

Back/controllers/PlatformController.class.php

\_check();



提示：转成小写比较，可以避免大小写问题！