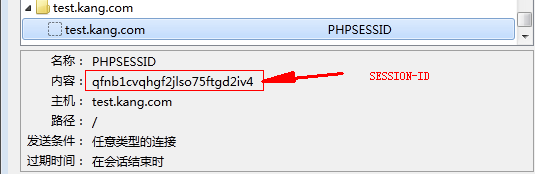
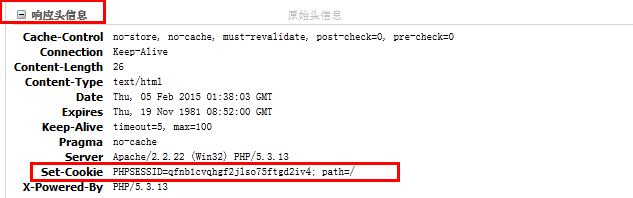
# SESSION

## Session原理

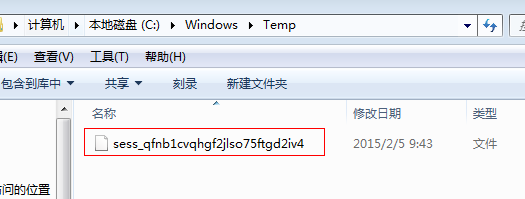
存储于浏览器端Cookie中的session-id，就是一个普通的cookie变量（在session机制中尤其特殊的含义）



也是在响应时将session-id cookie变量PHPSESSID设置好：



每个会话，所生成存储于服务器端的session数据区  
默认的，以文件的形式，存储于服务器端操作系统的临时目录中：



## Session数据属性

### 有效期

会话周期结束：session id会在会话周期结束后过期，再访问会产生新的会话id

### 有效路径

整站有效

### 有效域

当前域

### 是否仅安全连接传输

否

### 是否仅为HTTP使用

是

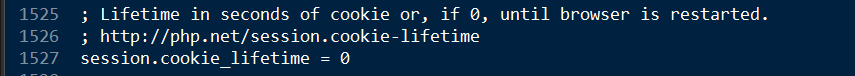
### 属性的管理

以上session数据的特征，都是由，浏览器cookie中所存储的session-idcookie变量的特征所导致的。

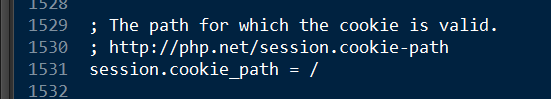
可见，如果需要更改session数据的属性，则需要更改存储session-ID的cookie变量PHPSESSID的属性：

Php.ini中存在该属性的配置：

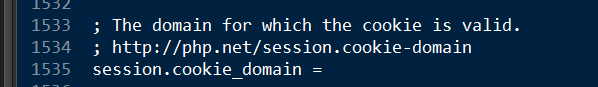
有效期：



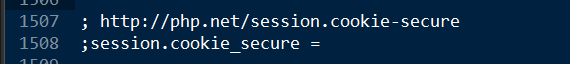
路径：



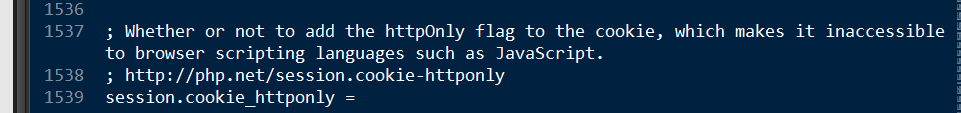
有效域：



仅安全连接传输：



HTTPONLY：



如果需要对默认属性加以修改，办法如下：

A计划：

更改php.ini配置文件即可（不建议）

B计划：

通过在脚本中，使用函数ini\_set()来进行配置的修改，仅在设置后的脚本周期内有效，要保证在开启session前设置完毕(可以选择)

Ini\_set(‘选项’, ‘值’)

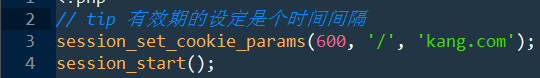
C计划：（推荐）

使用特定功能函数:

session\_set\_cookie\_params(有效期, 有效路径，有效域，是否仅安全传输，是否HTTPONLY)

完成设置。

在开启session前设置完毕



注意：

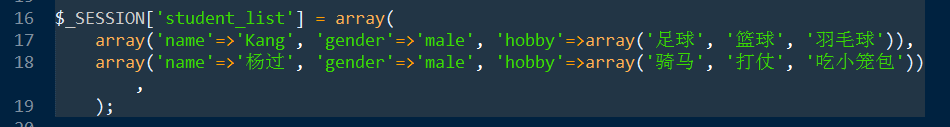
选项的设置是针对cookie中的sessionID，因此是针对所有session数据的。

Tip：

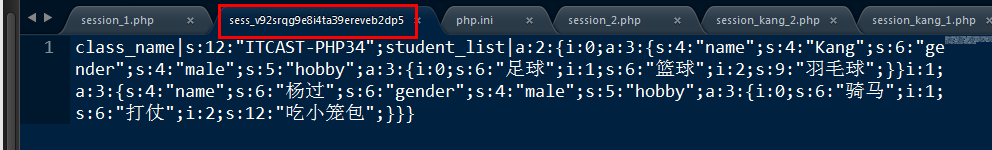
Session属性通常都会保持其默认值，不建议修改！

## Session使用语法问题

Session数据可以是任意的数据类型



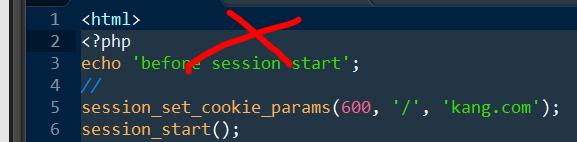
就是因为，session数据区，内的数据是序列化后才存储的！



$\_SESSION数组元素的下标，仅仅可以是字符串类型



Session\_start()，类似于header函数，前不应该有输出！



## Session数据区

在脚本周期外，持久存储当前会话session数据的区域。

在脚本周期内，使用$\_SESSION这个变量管理的会话session数据。

观察：$\_SESSION与session数据区的交互情况：

## Session销毁

使用函数

session\_destroy()

完成。

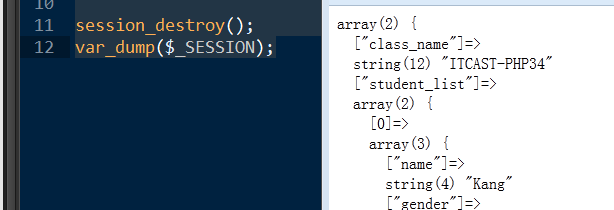
销毁：删除当前session对应的数据区，关闭session机制！



关闭session机制后，导致余下的session操作都不处理！

注意：

$\_SESSION 变量，在销毁session后，是不会自动消失，但结束不完成写操作。因此下次脚本周期，就不能获取到存储的session数据。



如何完整删除与当前session相关的全部数据？

Session\_destroy(); 数据区

Unset($\_SESSION); 销毁变量

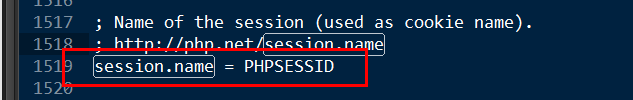
//setCookie(‘PHPSESSID’, ‘’, time()-1), 销毁cookie中的session-iD

setCookie(session\_name(), ‘’, time()-1), 销毁cookie中的session-iD

tip：

PHPSESSID，称之为session.name 可以被配置的php.ini

使用函数session\_name()来获取当前的值！



如何清空session数据？

不要 unset($\_SESSION)

需要：$\_SESSION = array();

## 重写session的存储机制

目的：

1. 便于管理大量的session数据
2. 便于web服务器集群共享session数据。

方案：

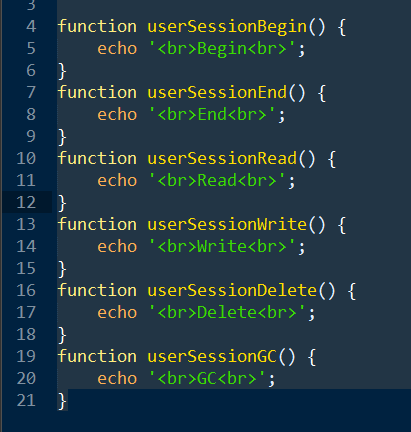
入库，入内存

实现过程：

定义（实现）自定义的相关的存储处理函数

将其设置为session机制需要的存储函数(告知给session机制，使用我们的函数完成存储处理)

Session机制，共需要6个存储处理函数



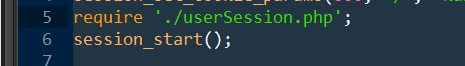
设置session的存储机制函数：

使用PHP函数：

Session\_set\_save\_handler(开始处理器，结束处理器，读处理器，写处理器，删除处理，垃圾回收处理器)



常规使用session机制即可：



### 建立session表

表中的记录，对应session文件。

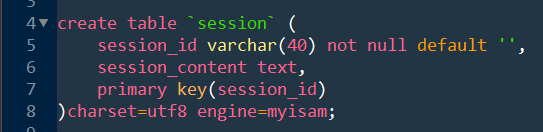
create table `session` (

session\_id varchar(40) not null default '',

session\_content text,

primary key(session\_id)

)charset=utf8 engine=myisam;



### 读操作



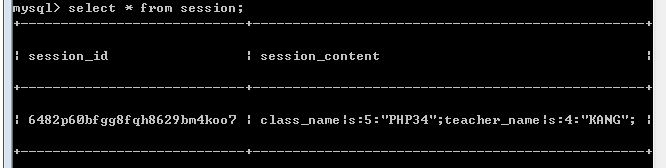
### 写操作

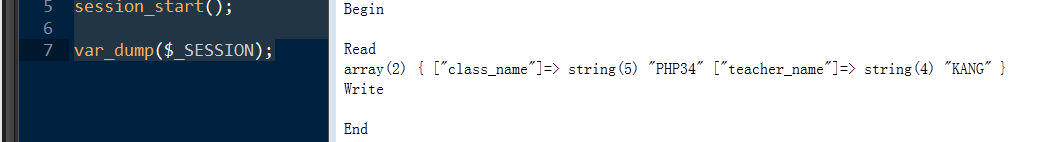
Insert into … ON DUPLICATE KEY UPDATE

REPLACE 语法与insert 一致，存在则替换，不存在则插入。

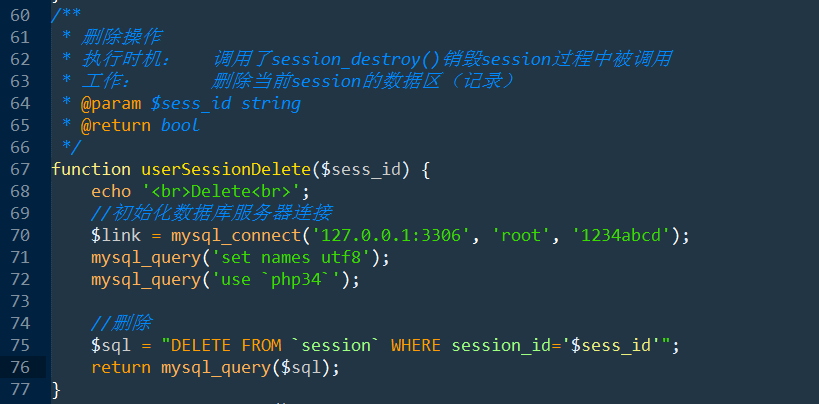


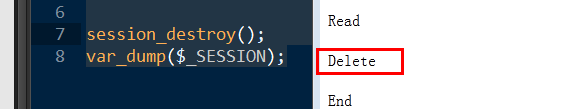
测试：





### 删除



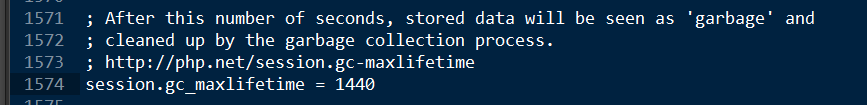


### 垃圾回收操作

如何识别垃圾数据区？

PHPsession机制设置session数据区的最大有效期，某条session记录（数据区），在最后一次处理后，如果超过了多久之后没有被使用，则被视为垃圾数据。

该时间默认为1440s，可以被配置：



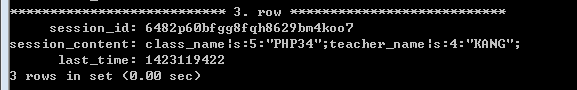
同时需要记录每个session数据区的最后处理时间。

增加字段，记录最后处理时间：



在写操作时，将其更新：





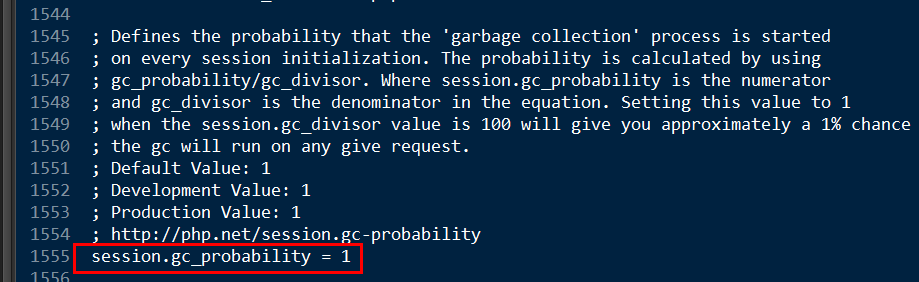
如何执行删除？

在开启session机制过程中，有概率的执行垃圾回收操作。

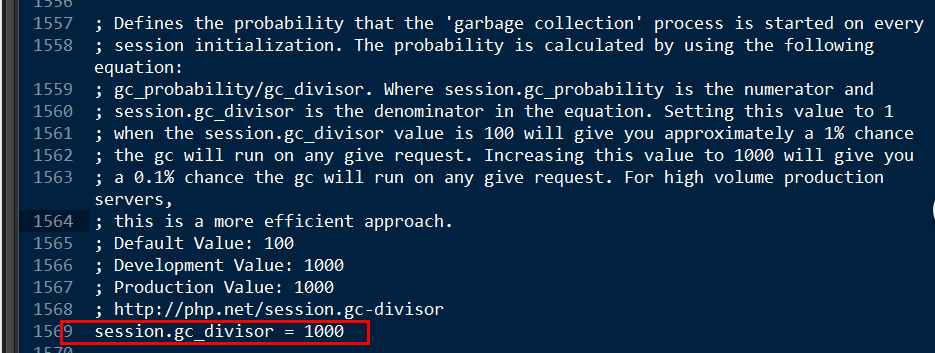
默认的概率为1/1000

可以配置：

可能性：

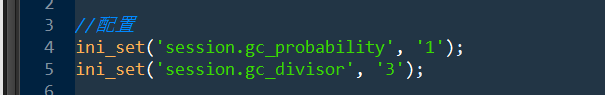


除数基础：

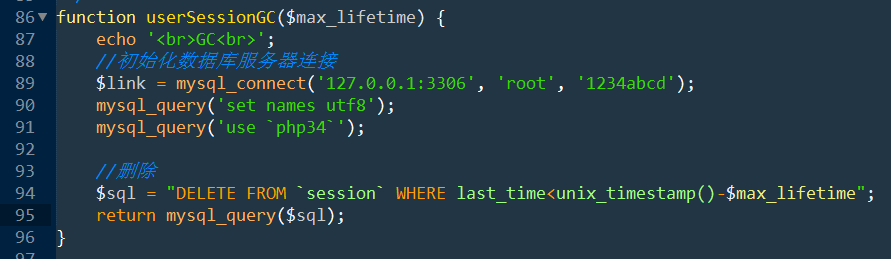


修改：

Ini\_set();



实现该操作：

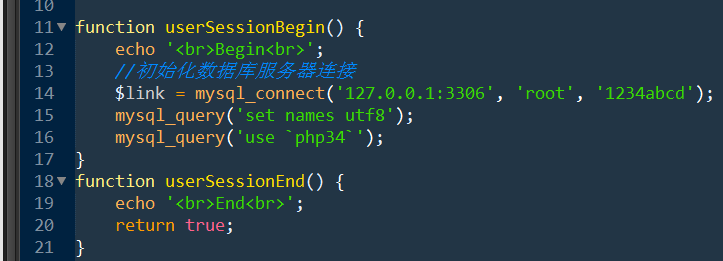


### 开始

Session开启时，最早执行的一个存储机制相关方法，用于初始化存储操作的相关资源。

### 结束

在session机制关闭时，执行的方法，最后一个执行的存储相关操作，用于收尾性工作！

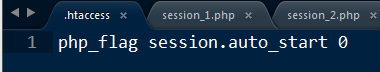


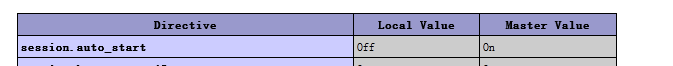
### 补充

先执行 session\_set\_save\_handler()在session\_start()

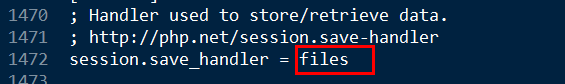
保证session不自动开启

可以通过.htaccess配置需要的选项：



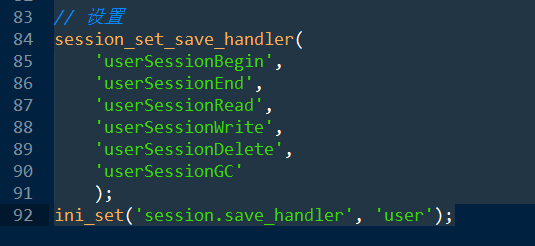


配置 session.save\_handler



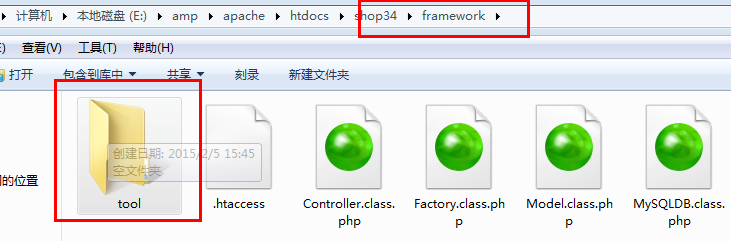
FilesPHP内置的session存储处理器。

建议：重写session存储机制，应该将其改为user表示用户自定义！



# 项目中的session入库

以框架基础代码中 扩展工具的角色 出现项目中！



增加相应的目录常量：

Index.php



采用面向对象的编程思想完成：

工具类。

要求：

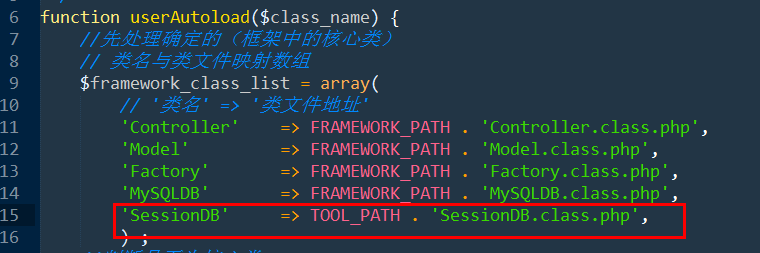
增加session入库工具类（完成其自动加载）

入库操作由工具类对象的方法充当。

在实例化该工具类对象时，完成设置session处理器，并开启session。

增加session入库工具类（完成其自动加载）



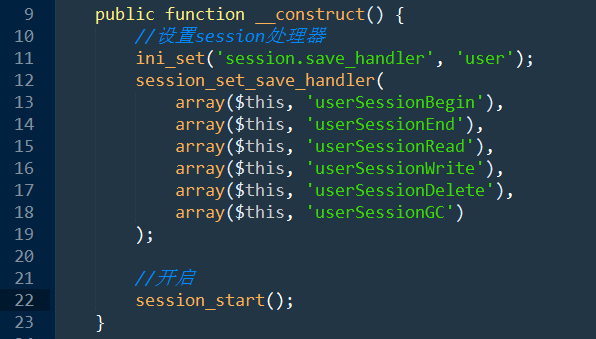


入库操作由工具类对象的方法充当

要求使用项目中统一的DAO完成数据库操作。

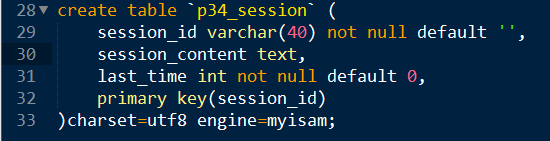
在实例化该工具类对象时，完成设置session处理器，并开启session。

由构造方法完成：

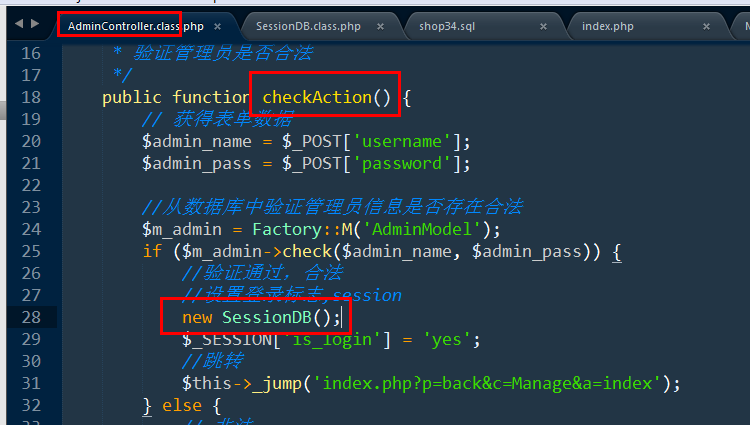


使用：

创建session表：



项目中，原本session\_start()，由new SessionDB即可！





# 会话总结

## Session pk cookie？

联系：

都是实现会话的方法。

Session基于cookie。

差异：

|  |  |  |
| --- | --- | --- |
|  | cookie | session |
| 会话数据存储位置 | 浏览器端 | 服务器端 |
| 安全性 | 低 | 高 |
| 数据传输量 | 大 | 小 |
| 支持会话数据量 | 有限制4k，20个 | 无限制 |
| 支持数据类型 | 字符串 | All |
|  |  |  |
|  |  |  |

## Session数据持久化？

浏览器端session-id

Session\_set\_cookie\_params(3600);

服务器端session数据区

Ini\_set(‘session.gc\_maxlifetime’, ‘3600’);

Tip:session很少做持久化，如果需要持久存储的会话数据，就选择cookie。

## Cookie禁用，session是否可用？

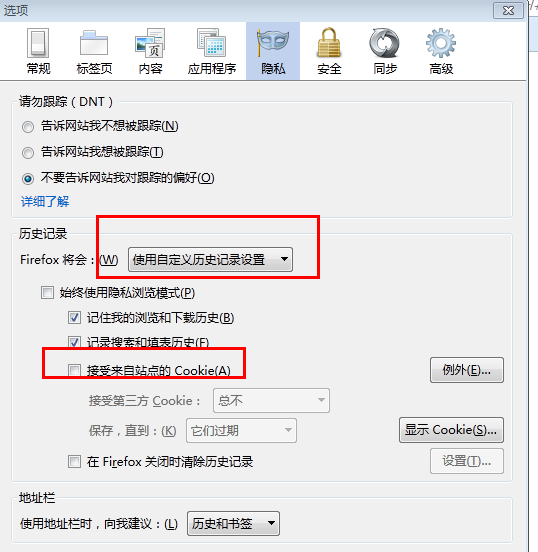
常规：

Cookie禁用，session不能用！

技术上，可以使用：

Cookie用来存储session的 ID，每次请求时携带。

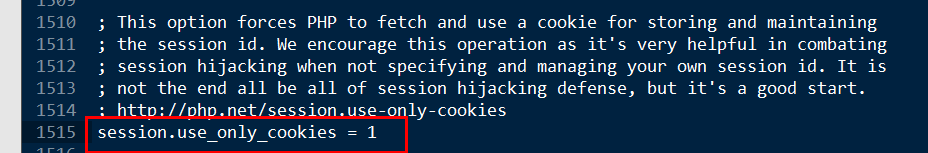
通过 get，post，方式向服务器传递session-ID。



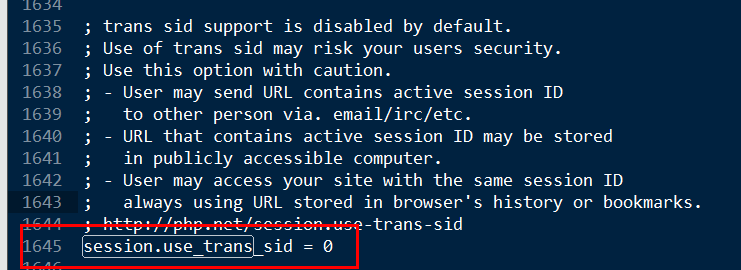
一下操作仅仅针对内网系统：

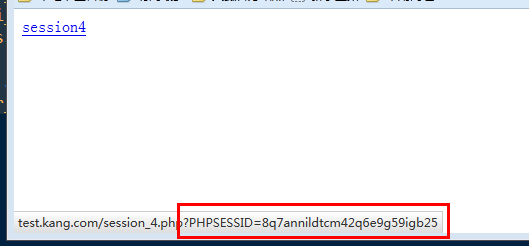
PHP提供如下的设置，帮助在cookie禁用的情况下，通过get或post方法传输sessionid：

是否仅仅使用cookie来传输session-ID：



是否自动采用其他方式传session-id





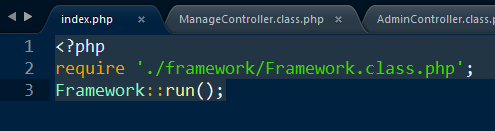
# 框架类（框架初始化类）

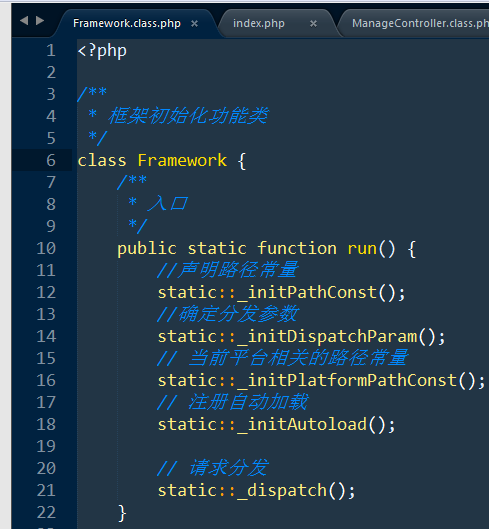
将原来入口文件中功能，放在该类中完成，入口文件变得简单，轻量！

将入口文件中的各个功能，由框架类的各个方法，完成：

为了简单化，使用纯静态的类。（看成功能的集合）

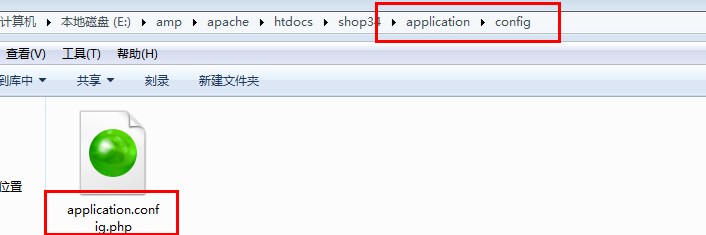
此时入口文件：





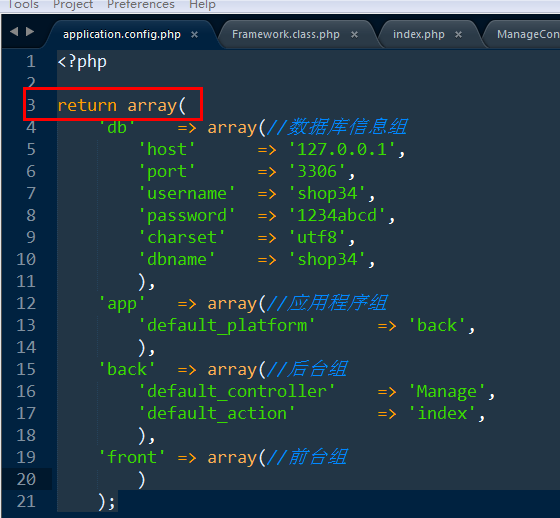
# 配置文件

在application目录中，增加一个子目录config用于管理项目中所出现的配置文件：



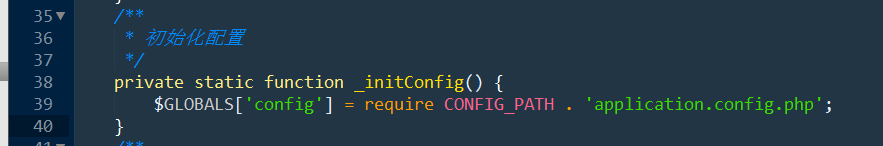
设计，配置文件存储数据的格式：

对配置进行分组管理：

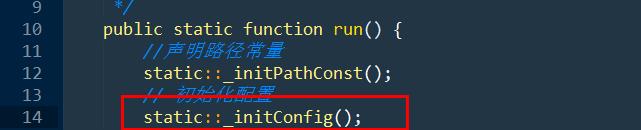


初始化项目时，载入该配置文件，获取配置信息

在框架类中，增加载入配置的操作：

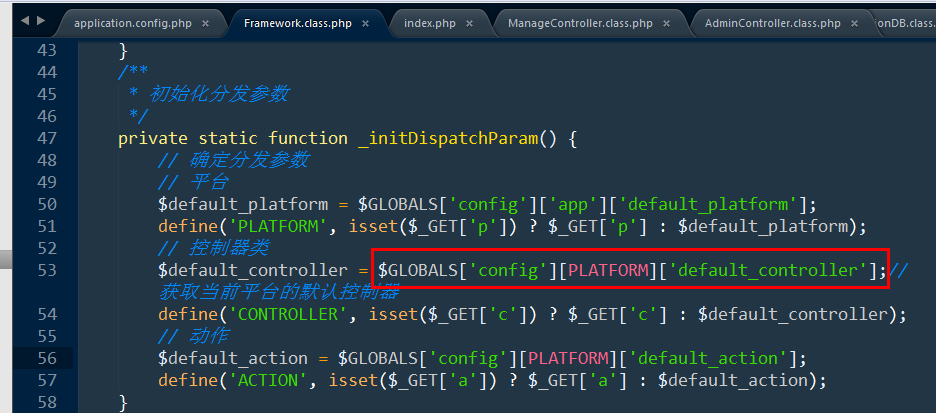


在初始化完路径常量后，就可以完成对配置的初始化：



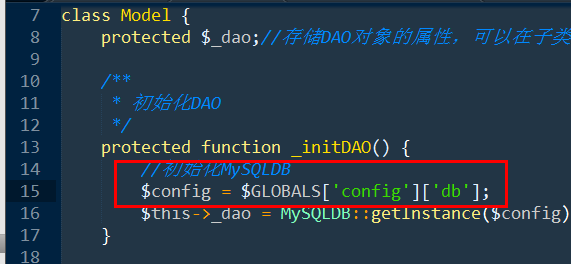
使用配置完成：

分发参数：



数据库服务器信息：

Model：



SessionDB：

