# Cookie练习 – 记录登陆状态

COOKIE中存储了可以认定当前浏览器处于登陆状态的信息才可以完成！

通常保证存储的信息具有以下特点：

可验证性。

保密性（安全性）。

通常可以采取，存储 加密处理 后的 管理员ID 和 密码 的组合！

如下：

## 记录

### 控制器

Back/AdminController->CheckLoginAction()

Checkbox: 如果选中，则提交数据为 checkbox的value值。

如果没有选中，不会提交该数据！

Back/controllers/AdminController.class.php

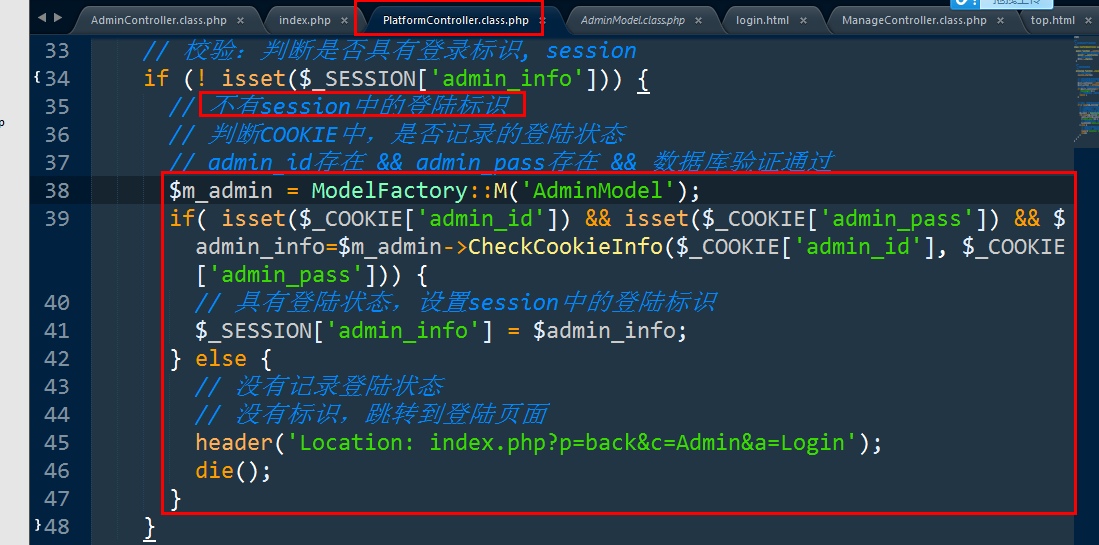


### 模型

## 验证

### 控制器

Back/PlatformController->\_check();



### 模型

使用AdminModel的CheckCookieInfo()方法：

Back/AdminModel->CheckCookieInfo();



测试：

# 验证码 - 绘图技术

作用？

防治计算机程序，向服务器发出请求从而达到恶意的目的：

例如：

登陆界面：暴力破解！

论坛：灌水，刷帖！

被采集：

特点，快！

提升 计算机请求的成本（时间，准确率）。

计算机技术不能成本很低从 图片上识别信息！

使用 表单中 增加验证码的方式，方式 计算机恶意请求服务器！



实现技术：

生成图片技术！

验证时：将码值存储与session中！

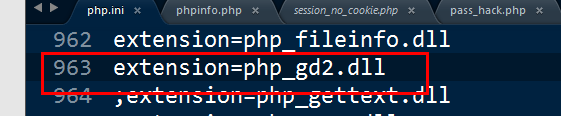
测试：

# PHP绘图技术：图像处理技术

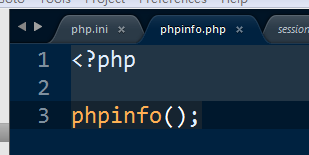
使用 GD 扩展完成！

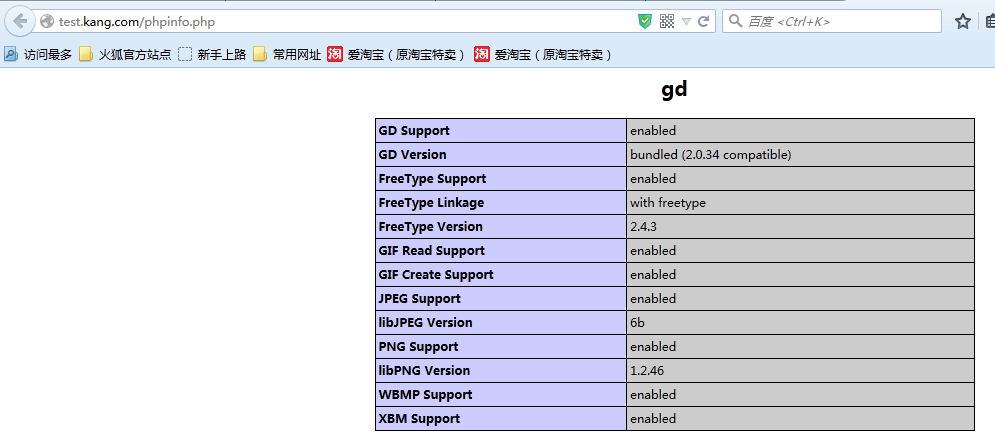
## 加载扩展

Php.ini



Apache httpd restart





处理图像的基本步骤。

例如：制作500\*250 的 蓝色图片！

## 创建画布资源

创建新的画布：

画布资源 = Imagecreate(宽，高);创建基于调色板画布（支持的颜色少）

画布资源 = Imagecreatetruecolor(宽，高);创建正彩色（支持的颜色多）

基于已有图像创建画布：

画布资源 = Imagecreatefromjpeg(图片地址);

Imagecreatefrompng();

Imagecreatefromgif();

测试：



## 操作画布

分配颜色:为某张画布分配某种颜色。

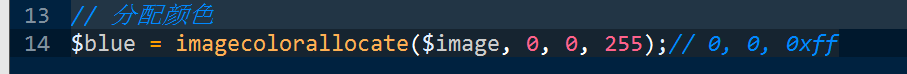
颜色的表示方式：RGB颜色。

颜色标识 = Imagecolorallocate(画布，颜色R， 颜色G， 颜色B)

#ff0000 红色

#000000 黑

#ffffff 白



填充画布：使用某个颜色，在画布的某个位置进行填充

Imagefill(画布，位置X，位置Y，颜色);

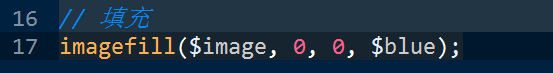
位置，采用坐标表示。

原点：左上角，0, 0

向右，X 增加

向下，Y 增加

右下角坐标：（width-1, height-1）



## 导出（输出）

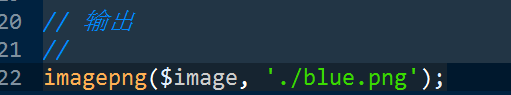
是函数：

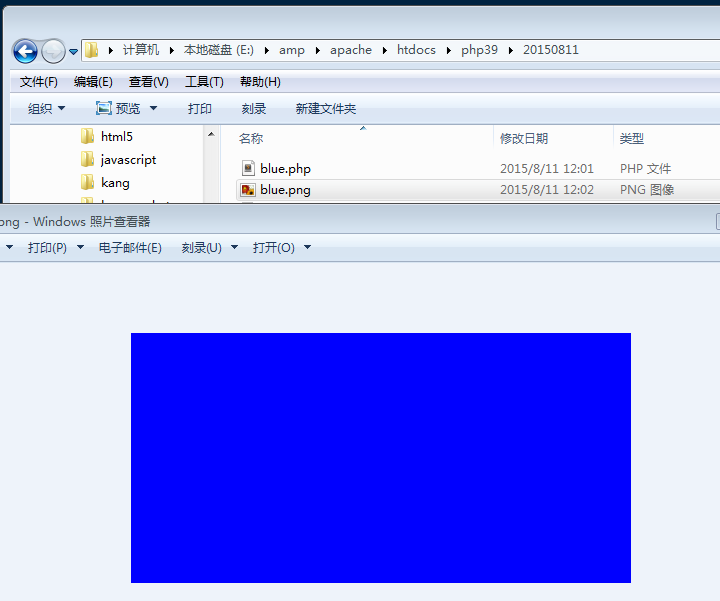
Imagepng(画布， 图片地址) 输出为PNG格式

Imagegif();

Imagejpeg()

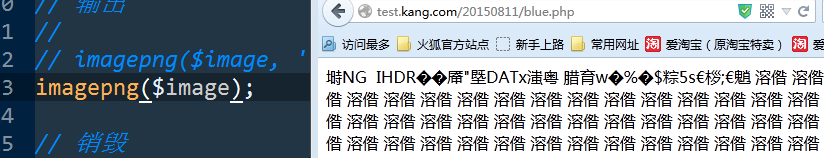
完成



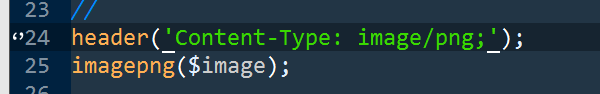


直接输出（到浏览器端）

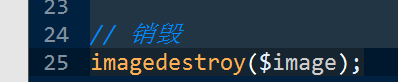
如果不使用个第二个参数，表示直接输出



需要，告知浏览器，所输出的为图片，否则浏览器会将所有的响应内容当作html代码字符串处理！



## 销毁资源



# 仿照ecshop完成验证码

特征：

点击更换

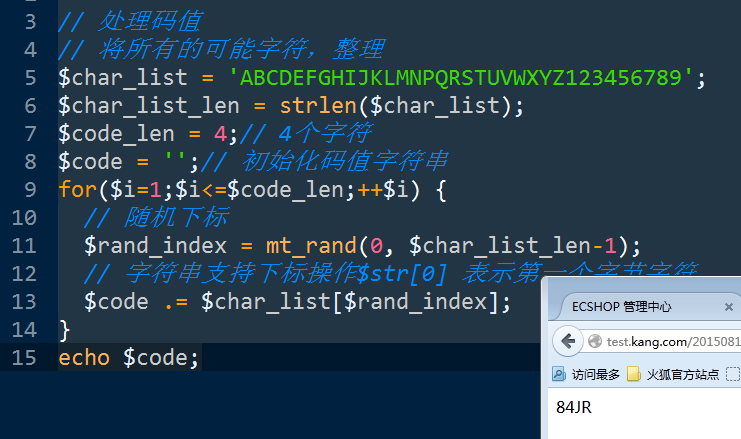
图片背景更换（在有限几张背景图片中选择背景）

码值：大写字母+数字 随机组合！4个长度

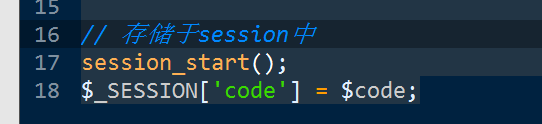
文字颜色：白色或者黑色

文字，居中显示

## 处理码值

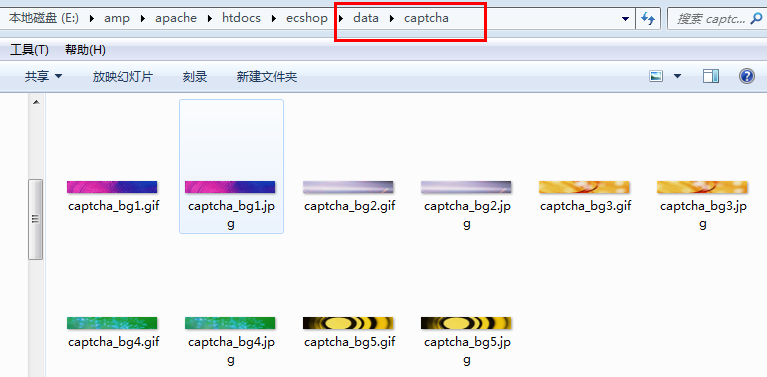


## 存储在session中

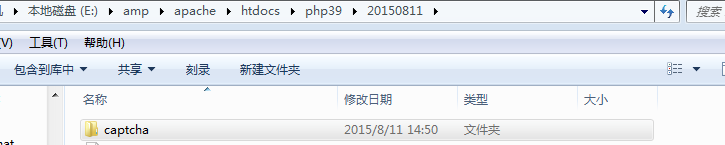


## 形成验证码图片

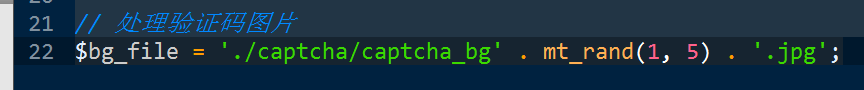
获取需要的背景图片：



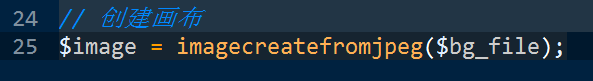
拷贝项目中：



获得随机背景图片：



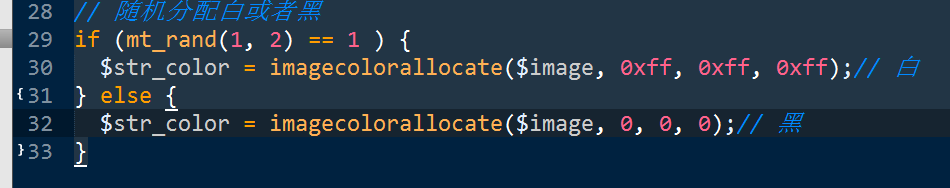
创建画布：



操作画布：

分配字符串颜色：

Imagecoloarallocate(画布，R, G, B);

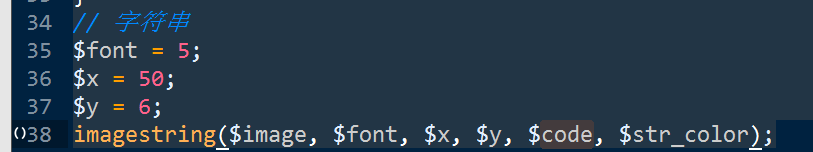


将字符串写到画布上！

Imagestring(画布，字体大小，位置X, 位置Y，字符串内容，颜色);

字体大小：该函数使用内置字体。通过1-5表示字号。1小5大。

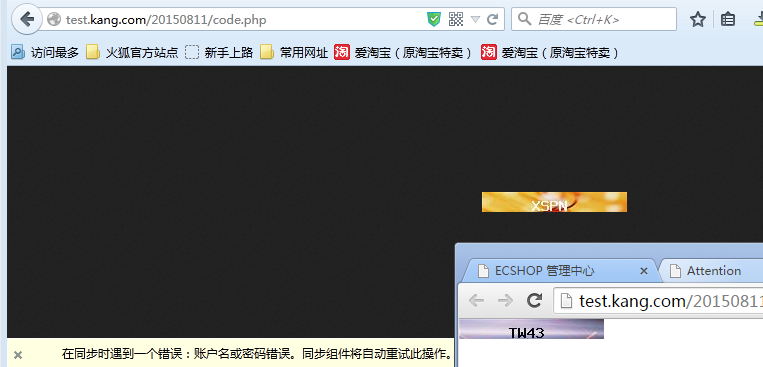
位置坐标，字符串左上角的坐标！



输出导出：



测试：



### 补充文件居中

图片宽高：

宽 = Imagesx(画布);

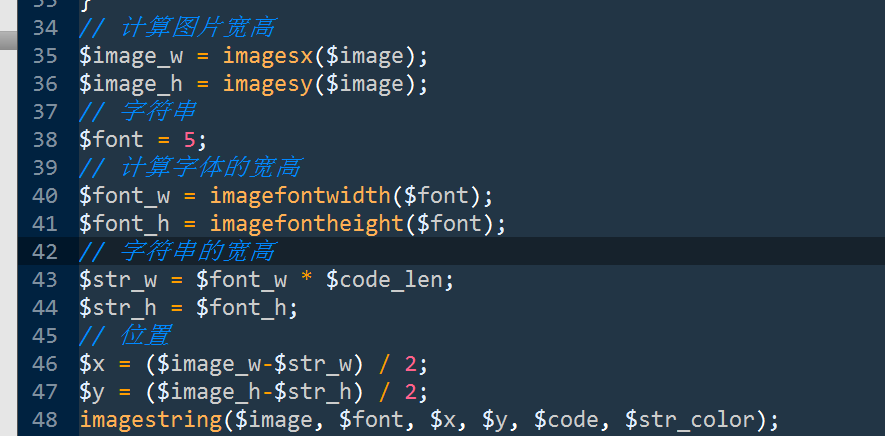
高 = Imagesy(画布);

内置字体的宽高：

imagefontWidth(字体);

imagefontHeight(字体);

字符串宽高：



# 项目中使用验证码

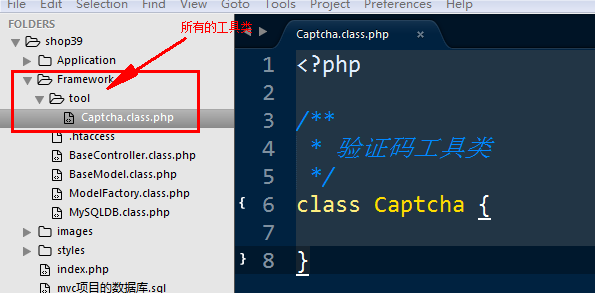
考虑到验证码功能，不仅一个项目一个功能要用，几乎所有的项目都要使用！

将验证码处理的代码，放置 项目框架目录中framework！

定义 验证码相关功能 的 工具类（扩展功能类）。

工具类：专门用于完成某一个功能类封装！

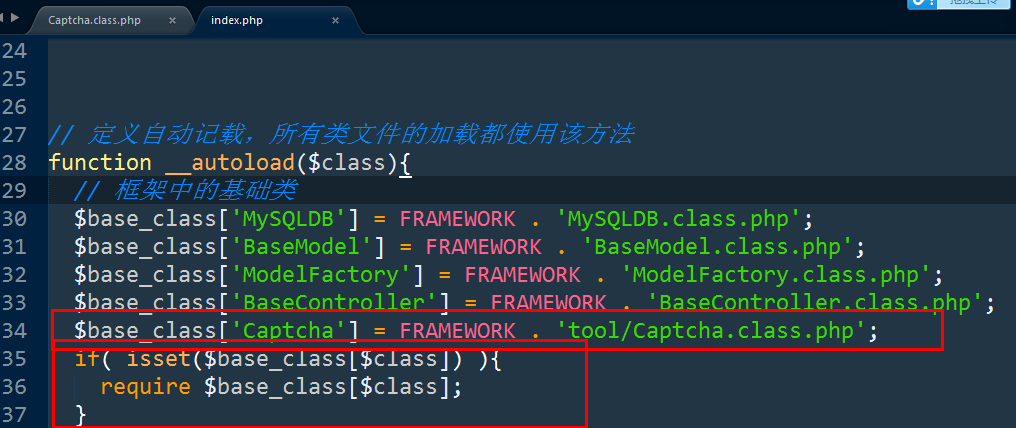
## 在framework建独立的目录存储工具类代码



## 处理该类文件的自动加载

处理 自动加载，将所有的框架基础类，与 类文件地址，做一个映射对应！

Index.php



## 实现生成验证码图片的方法

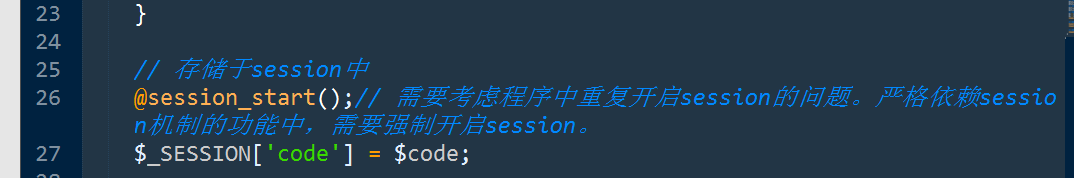
Framework/tool/Captcha.class.php

拷贝刚刚的练习：

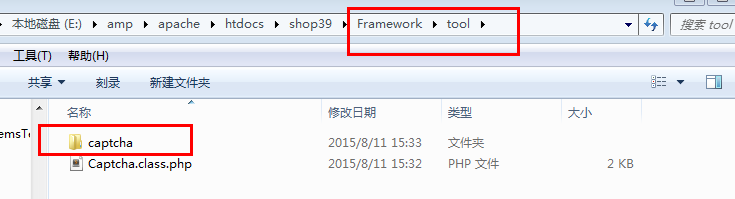
去掉长度，使用参数来充当：

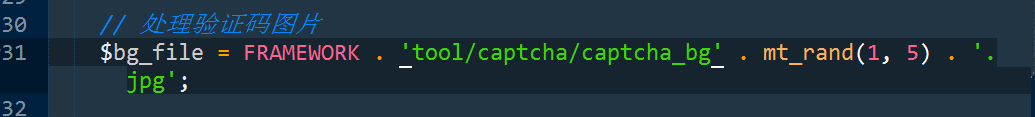


Session开加@



图片地址问题：存储与工具目录！





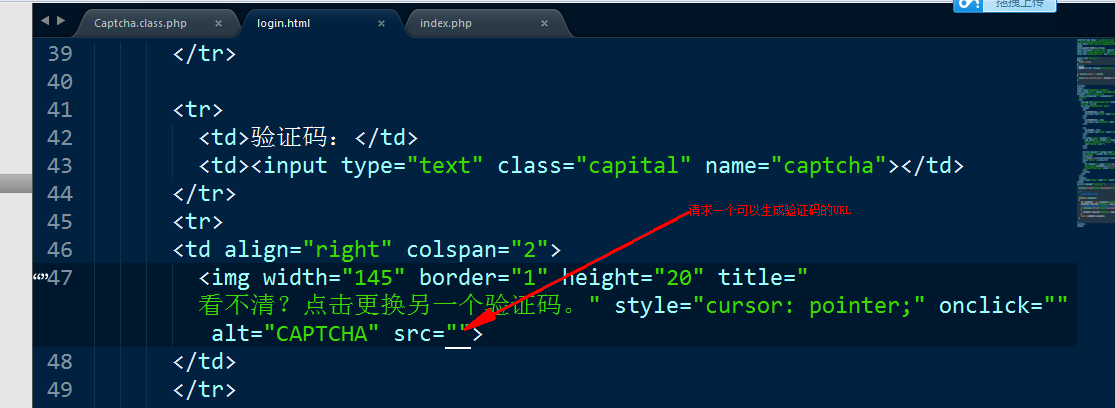
Tip：

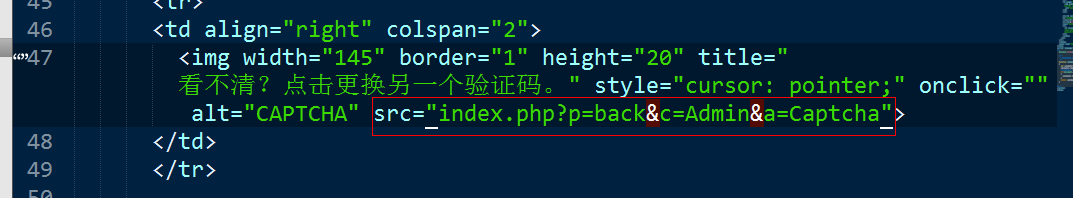
CAPTCHA项目是Completely Automated Public Turing Test to Tell Computers and Humans Apart (全自动区分计算机和人类的[图灵测试](http://baike.baidu.com/view/94296.htm))的简称

## 使用工具类生成登陆页面验证码

在登陆表单模板中，使用img标签，请求可以生成验证码图片的URL地址，生成验证码！

Back/views/login.html

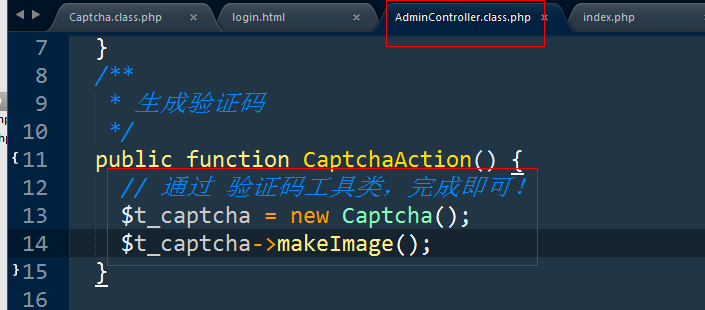




提供一个动作，处理该验证码图片请求：

完成该动作：

Back/AdminController->CaptchaAction();



测试：



处理错误后：



## 错误的处理

图片程序很多时候不会报告错误！

解决方案：

一：直接请求生成图的URL

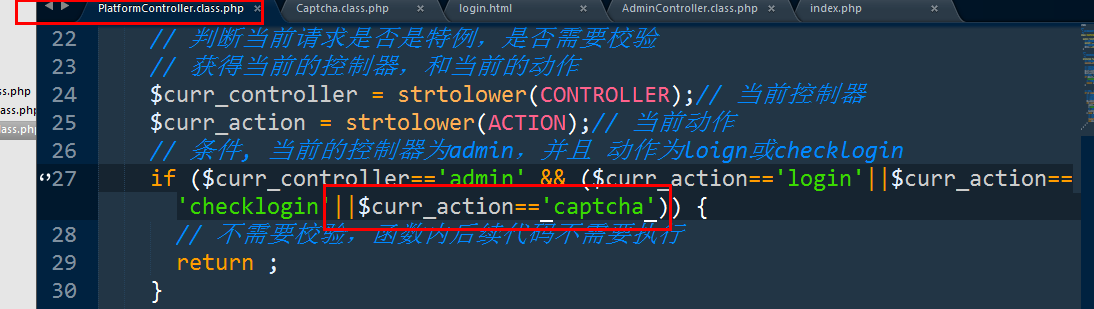


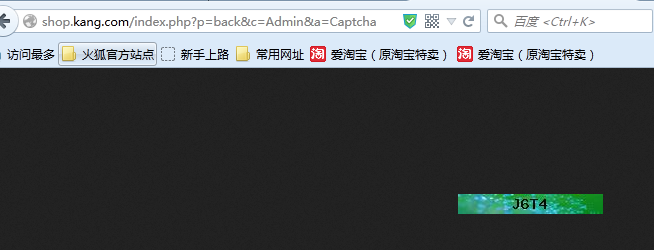
结果，跳转到了Login动作，可以，判断了没有登陆而跳转：

可见：该动作也是特例：

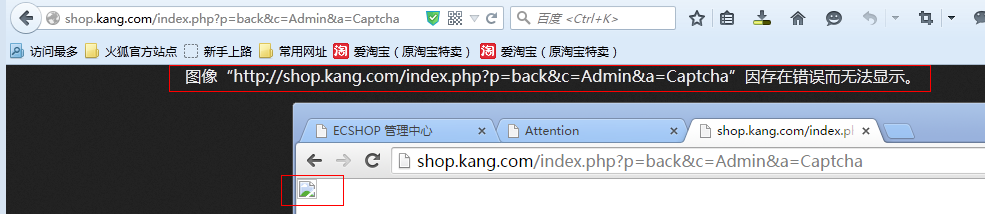
解决：

Back/PlatformContorller->\_check();

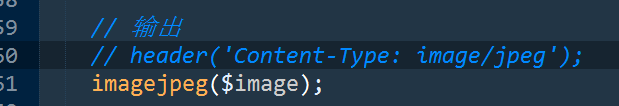




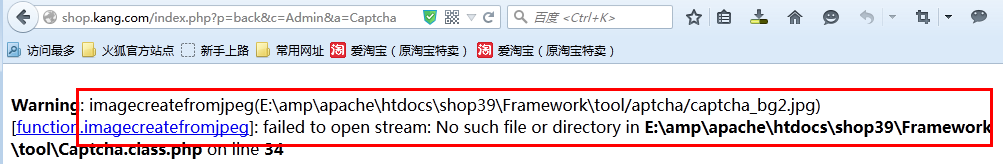
二：如果图片请求出现：



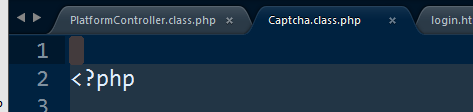
将 header(‘Content-Type: image/jpeg’)注释上！

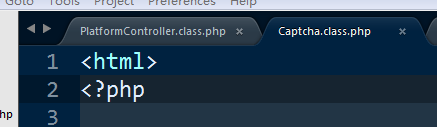


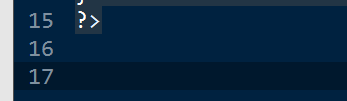
此时：



三：编码不规范，导致存在额外的输出



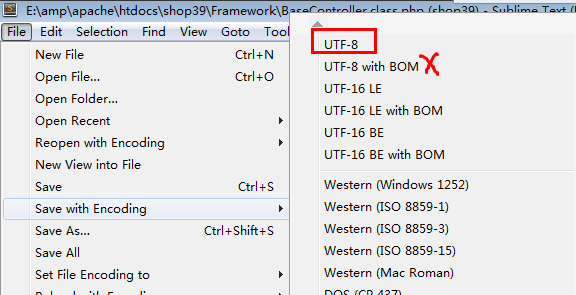




建议（要求）：写PHP程序，所有的程序文件，保证PHP标记要开头！而且同时不要写结束标记！

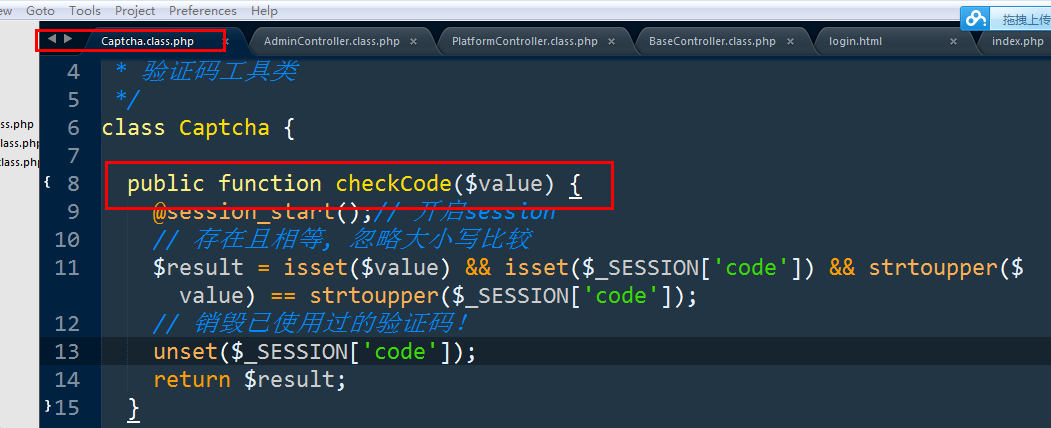
四：BOM，取消掉

UTF-8文件的签名。



## 实现验证码工具类验证码校验方法

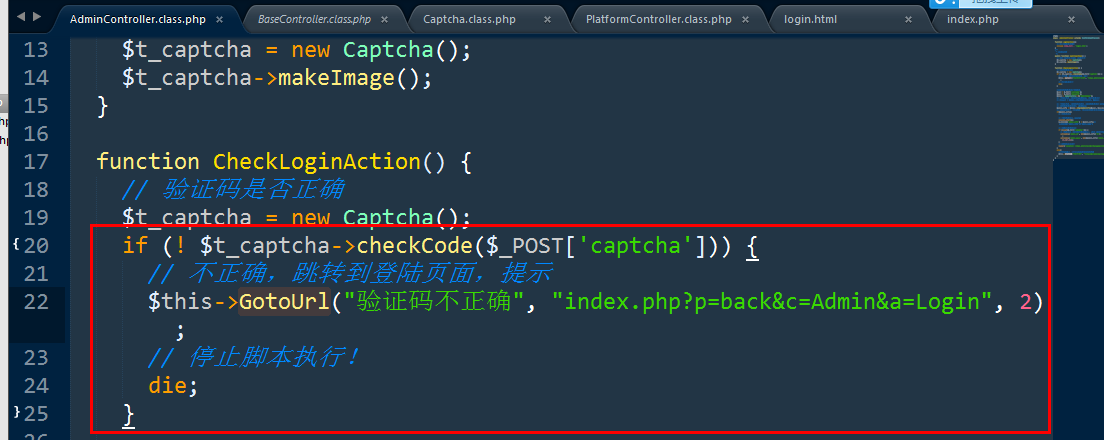
Framework/tool/Captcha.class.php

验证码使用完要销毁！切记！

## 校验验证码是否正确

提交登录表时，back/AdminController->CheckLoginAction();

通过调用验证码工具类的校验功能，得到校验结果！



注意：跳转后一定要强制停止脚本。

## 点击更换验证码

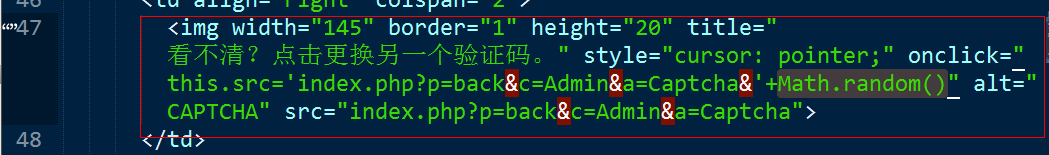
在点击图片时，重写请求生成验证码的动作。

语法实现：

Onclick：事件

执行更改当前图片img元素的src属性，此时浏览器就会发出新的请求。

Back/view/login.html



# 文件上传

上传： 文件从浏览器端传输到服务器端。

浏览器与服务器的交互存在几种方式：请求，响应！

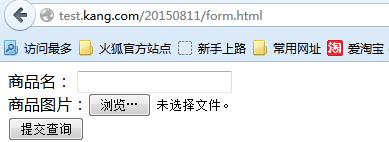
数据由浏览器端传输到服务器端，发生在哪个阶段？ 请求阶段。

结论：文件上传是发生在请求阶段。

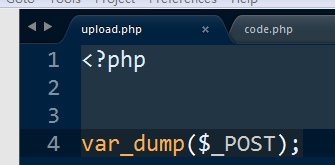
请求是浏览器发出的，浏览器将请求数据传输到服务器端。因此，文件也是由浏览器上传到服务器端！

表单：



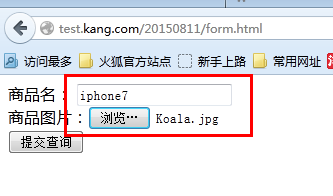


服务器端处理脚本：



典型的表单提交流程：

添加表单数据



提交到服务器端：



以上过程如下：

当服务器PHP程序，接收到浏览器post请求时，将所有接受到的字符串数据，存储到$\_POST数组中，使用！

## 当浏览器需要上传文件时

表单中的数据类型共两种：字符串型和文件型。

但是：默认情况下，两类会将所有的表单元素，视为 字符串型！

如果需要上传文件，告知浏览器，表单中的数据存在多种类型（字符串和文件型）：通过在form元素中增加属性：

Enctype=’multipart/form-data”

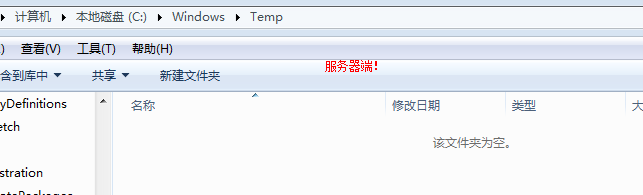
编码类型 = 多部分/表单-数据





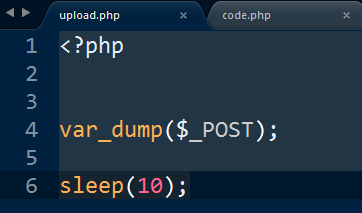
当服务器脚本PHP，接收到文件类型的post数据如何处理？

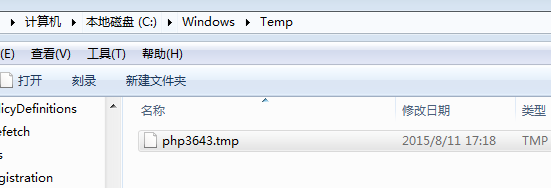
将接收到的文件，存储在 上传临时目录中。默认为 服务器所在操作系统的临时目录。



该临时文件有效期为脚本周期！（时间短）

通过延长脚本周期，从而测试：





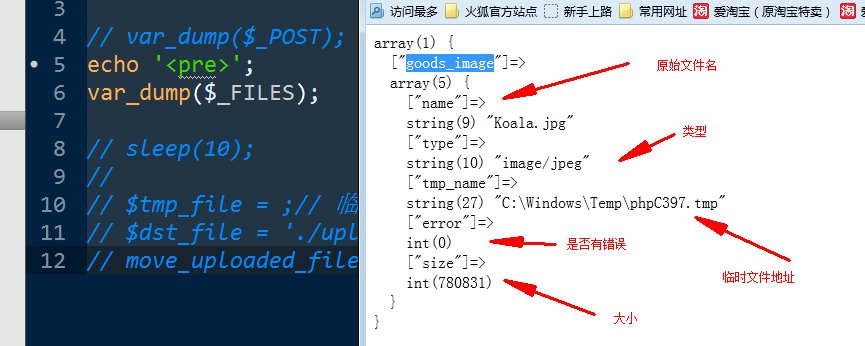
## PHP需要做的工作？

在临时文件未消失前，将其持久性存储！

通过函数：移动已经上传的参数

Move\_uploaded\_file(临时文件地址，目标文件地址);

通过 预定义变量$\_FILES获得所有的上传文件的信息。



结果！

