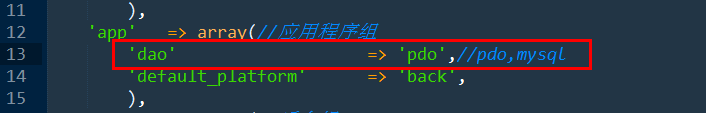
# 完成PDODB

功能与MySQLDB一模一样的DAO类！

项目中使用！

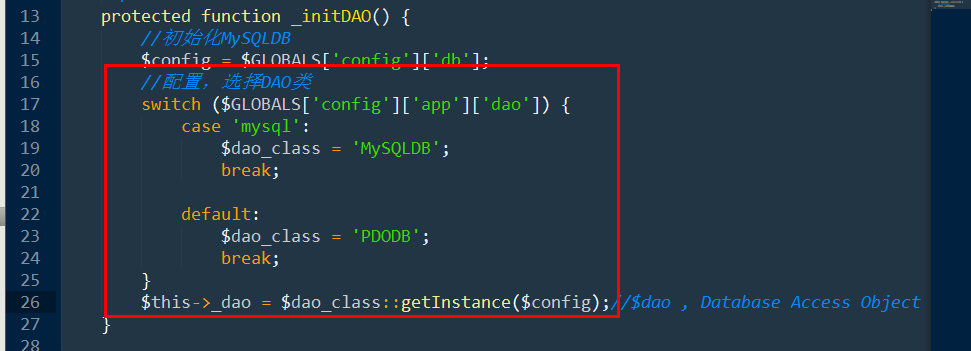
增加配置开关

Config/application.config.php

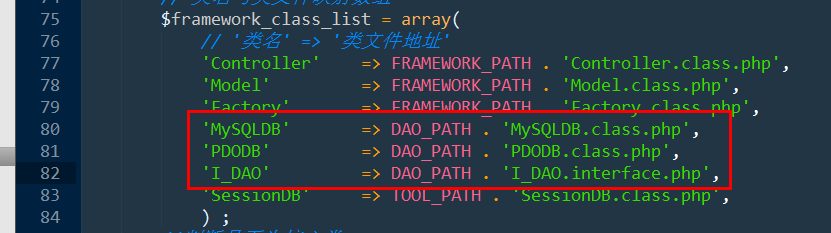


基础模型在初始化DAO时，利用配置，选择合适的DAO：

Model.class.php



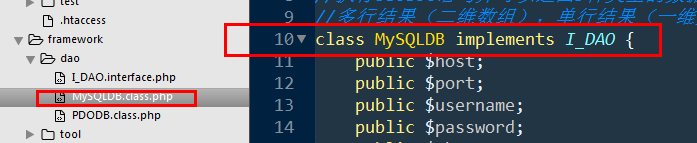
配置需要的I\_DAO接口，PDODB的加载情况：



路径常量：



将原来的MySQLDB，也实现与I\_DAO，并放置在dao目录中：



# 实现仿照ecshop后台验证码

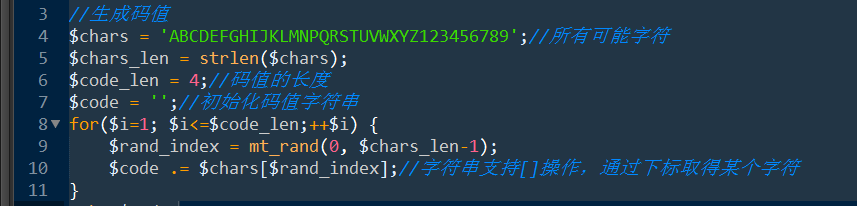
分析：

码值：大写字母&数字随机选择，位置居中，黑白随机

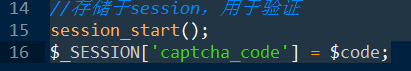
背景：固定的背景图片见随机。

支持点击更换。

码值：



存储于session：



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)的简称

利用ecshop提供的验证码背景图：

选择背景图：



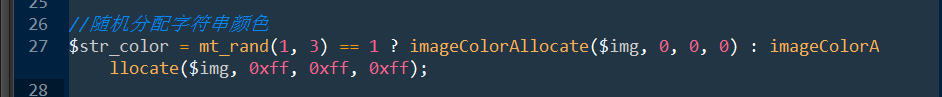
创建画布：



操作画布：

分配文字的颜色：

imageColorAllocate();

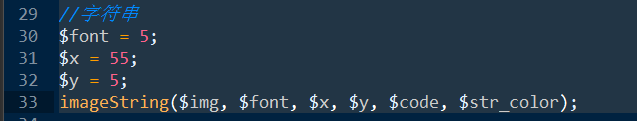


将字符串写到图片上：

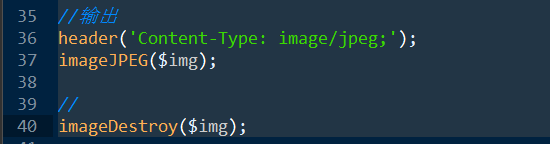
imageString(画布，字体，位置X，位置Y，字符串内容，颜色)

字体：该函数使用的内置字体，1-5表示。差异1小5大

位置X,Y：左边字符的左上角的位置来确定。



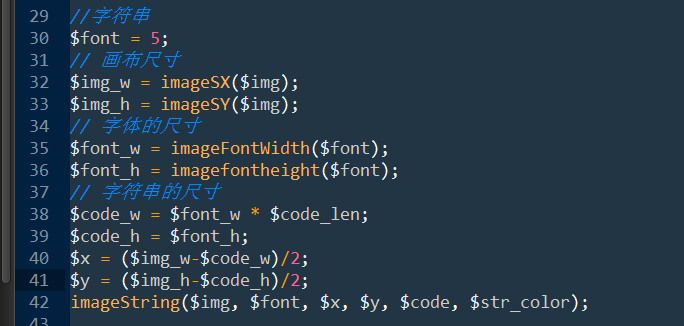
输出，销毁画布



计算字符串的居中：

画布大小：imageSX(画布)宽, imageSY(画布)高

内置字体大小：imageFontWidth(字体号)宽，imageFontHeight(字体号)高;

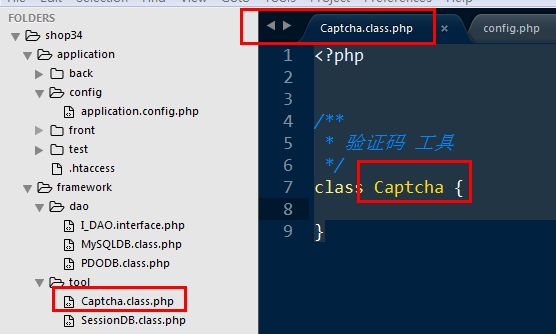




# 项目中实现验证码

## 验证码工具类

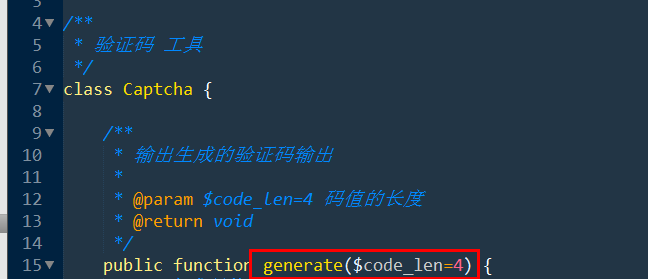
Framework/tool/Captcha.class.php



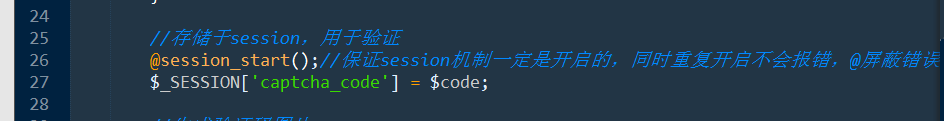
自动加载：



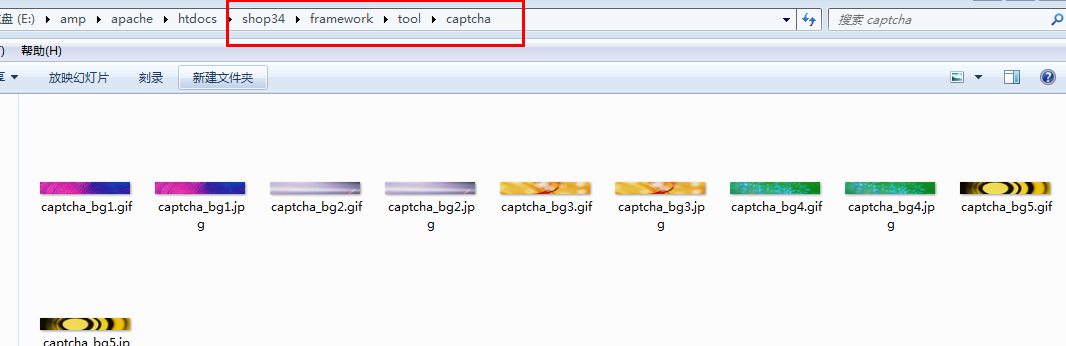
生成验证码图片：

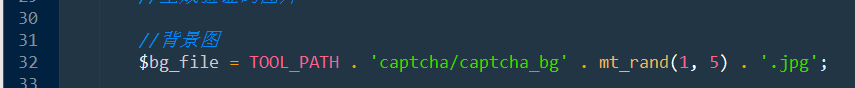


Session开启重复避免：



背景图位置：

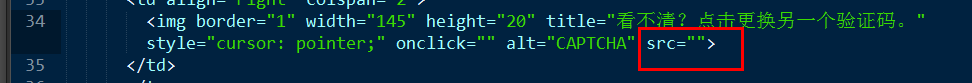




## 登录表单时展示

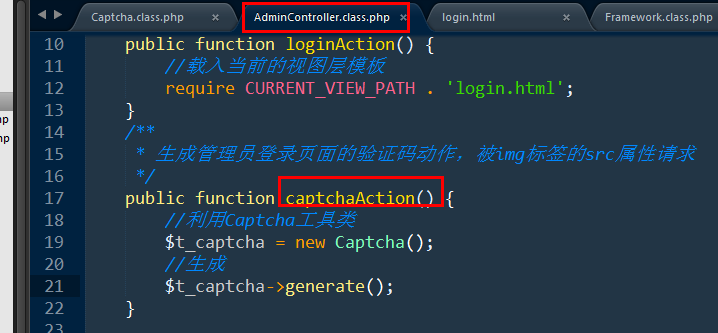
在登录表单页面，通过IMG标签，请求生成图片的URL地址，展示图片。

<img src=”URL” />

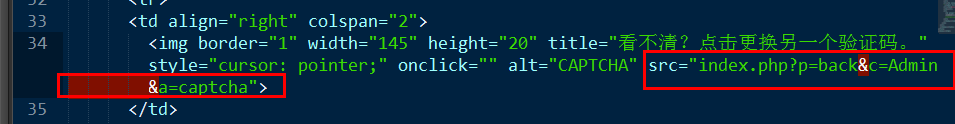


增加一个动作，用于完成生成后台登录界面的验证码图片：

Back/AdminController->captchaAction();



页面的img标签的src属性，请求该动作：

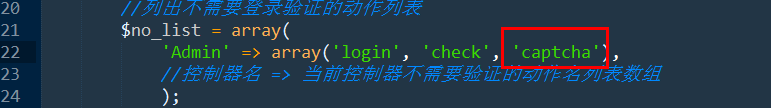


Tip：如果图片显示失败，如何调试？

应该直接请求生成图片的地址，而不是在img标签中查看。

我的程序，应该将captchaAction()也列为登入认证例外：

后台的平台控制器：



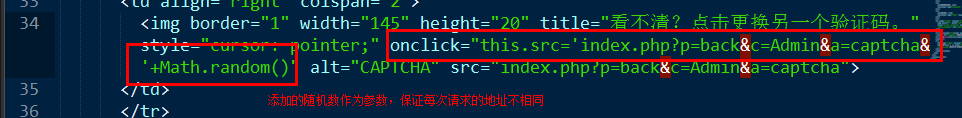
结果：



点击更换：

通过点击事件，对生产验证码图片的地址重新请求，形成新的验证码，进行展示！

如果img标签的src属性被更改，则浏览器会对新的url进行请求，展示在img图片区域内。



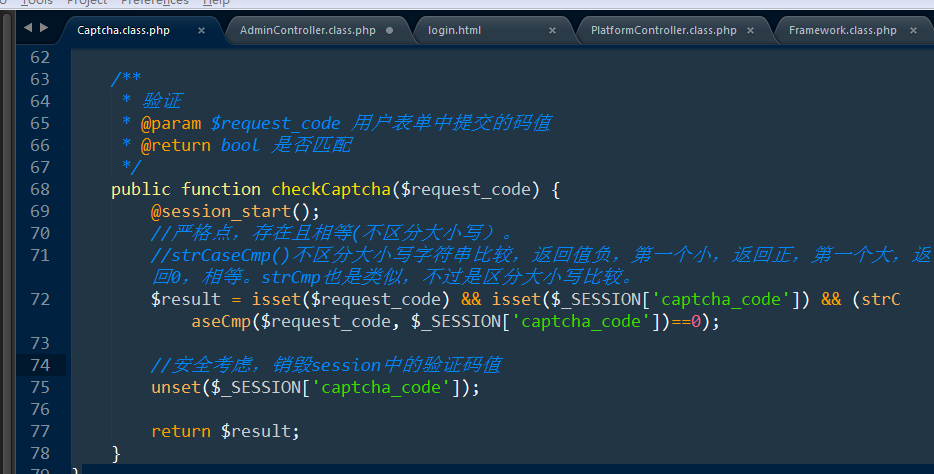
## 验证管理员信息前验证

在验证管理员信息的合法性前，对验证码作出验证：

Back/AdminController->checkAction();

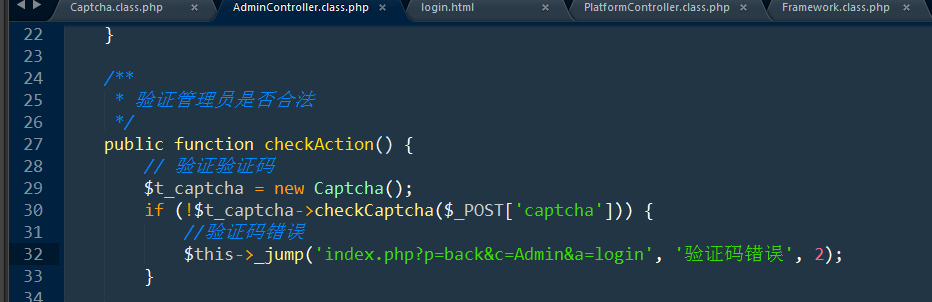
增加验证码工具类Captcha的方法

checkCaptcha()



Back/AdminController->checkAction();

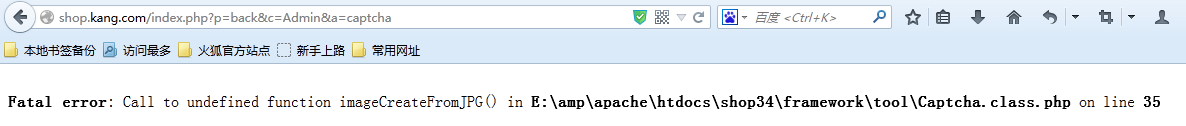
利用上面的方法完成验证：



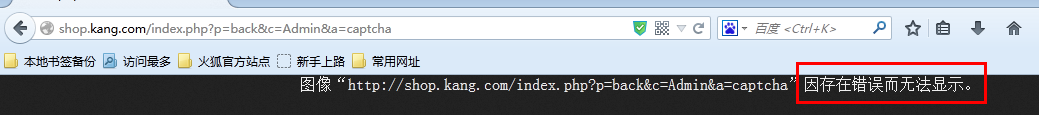
# tip：图片程序错误解析

step1： 请求生成图片的URL

语法错误：



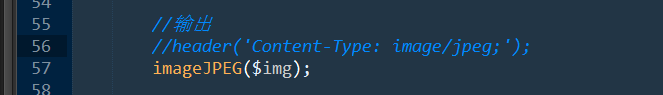
逻辑错误：

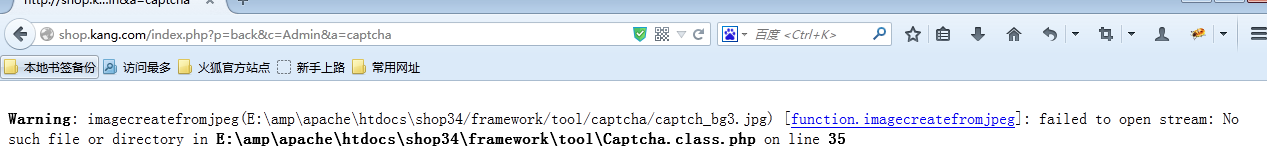


此时，如果出现语法错误，页面会有相应的提示。

但是没有语法错误，就如上面的结果，提示图片内部错误无法显示：

Step2：将header(‘Content-Type:image/jpeg’)注释





解决即可！

图片不能展示的原因：额外输出



BOM：utf-8编码文件的签名，三个字节的隐藏字符。也是输出的！