# 典型的上传实现

去判断 临时上传文件，是否满足 我们的业务逻辑需求！

满足，再移动！

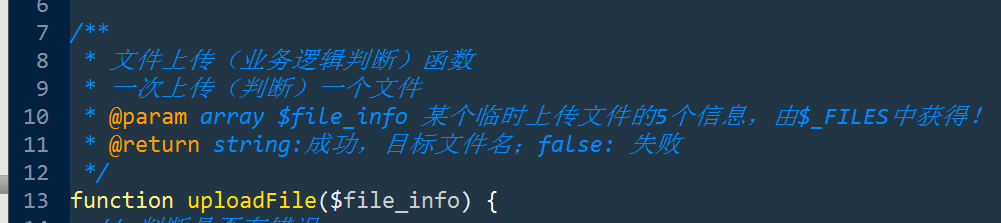
需要判断：

是否存在错误，类型是否符合要求，大小是否符合要求

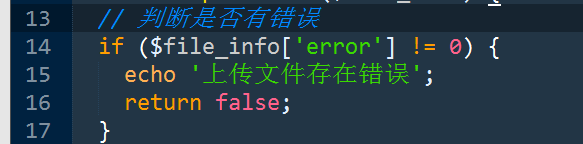
目标文件生成合理的名字

编码实现

## 当前上传临时文件信息作为参数



## 判断是否有错误



## 判断文件类型

两种表示方式：

后缀名(.jpg) 和 MIME(image/jpeg)

后缀名： 在文件系统中，表示文件类型的方式！当文件存储在磁盘上时，表示文件类型的方式！

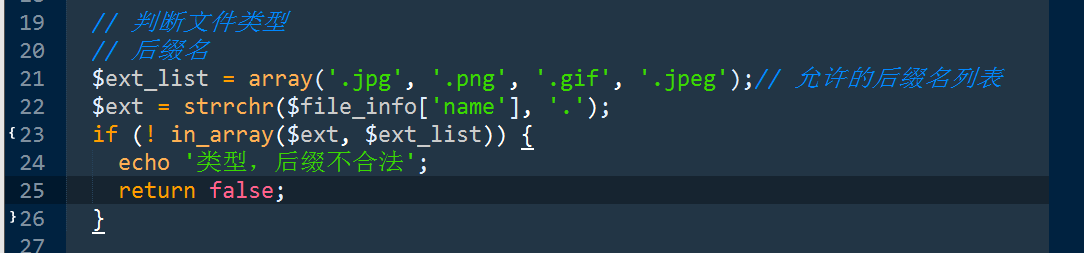
MIME: text/html, image/jpeg, 在互联网上传输文件内容时，表示该内容的方法！MIME(Multipurpose Internet Mail Extensions)多用途互联网邮件扩展类型。

判断类型时，通常两种方式都需要判断：

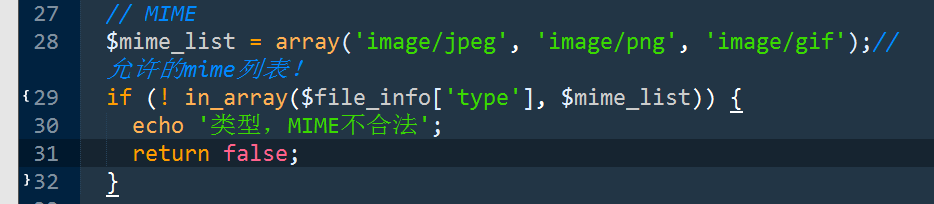
一：后缀名： 在原始文件名中截取

strrchr(); 字符串函数，查找某个子字符串在字符串中最后一次出现的位置，并返回从该位置到字符串结尾的全部内容！

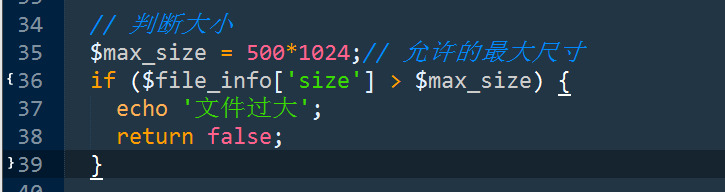




二：Mime： 使用type元素值！



## 判断大小

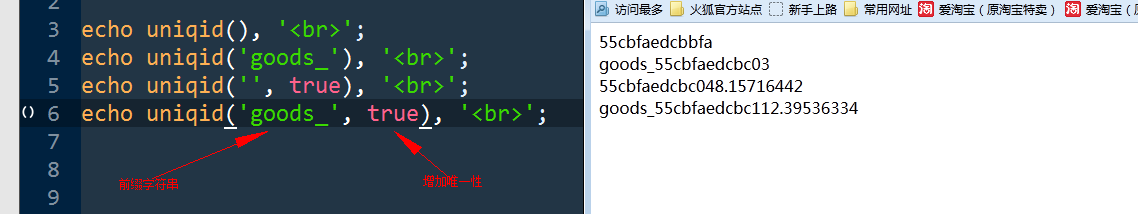


## 设计目标文件位置

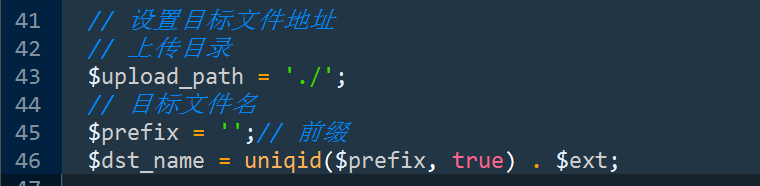
目标文件名：

不能重名，不能存在特殊字符，具有逻辑含义，保持原有后缀。

建议使用 函数：uniqid(), 生成唯一字符串！



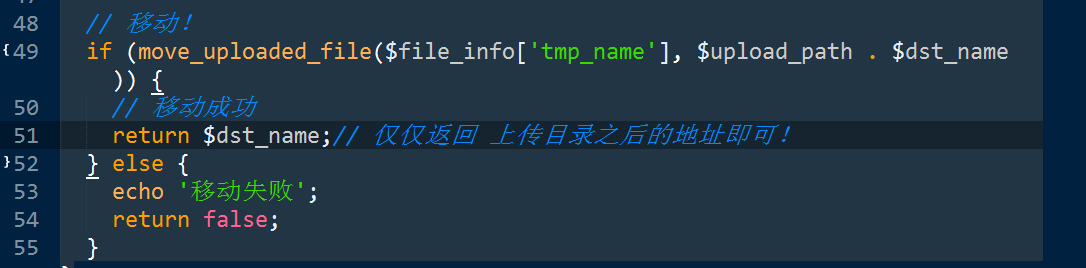
使用：



## 可以移动

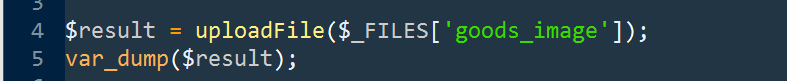
如果移动成功，应该返回，目标文件名！

为了后续处理该文件。否则文件名时临时生成的，不容易确定！

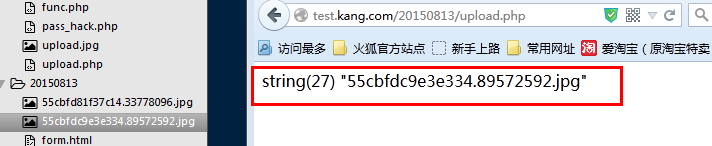


## 测试

失败：



成功：



## 分子目录存储上传文件

为了减少 一个目录中文件过多！

划分成不同的子目录：

划分方案：

按照业务逻辑： 商品图片，用户logo。指定不同的upload\_path即可

文件数量： 每个目录存储2000个文件，数量达到，则创建新子目录！

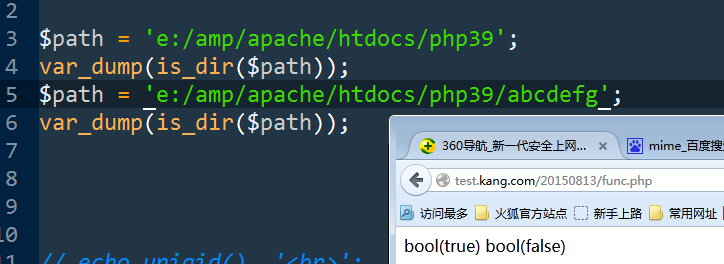
时间划分： 每个月（星期）使用一个上传子目录。

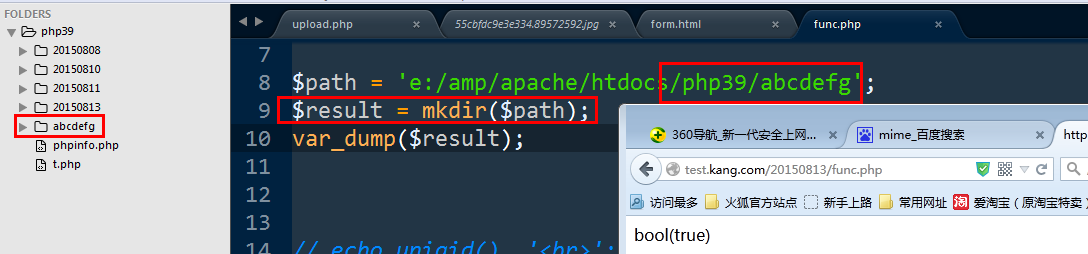
时间划分：

获取当前应该使用的子目录名：date()

判断需要的目录是否存在：is\_dir()是否为一个目录

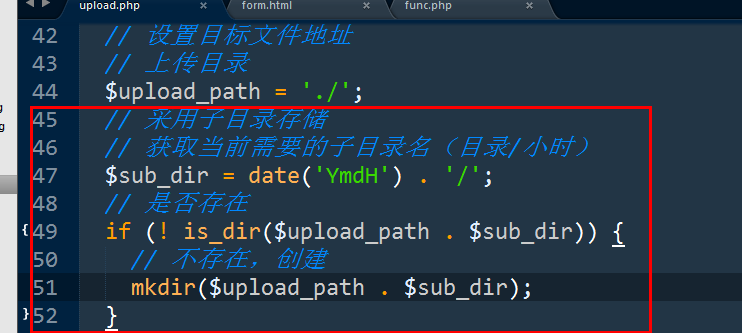
创建需要的目录：mkdir() 创建目录



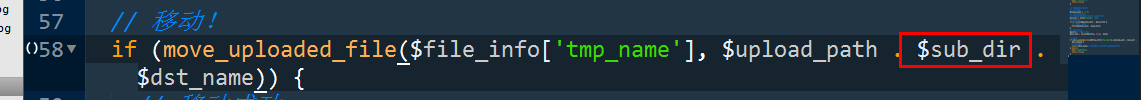


实现：

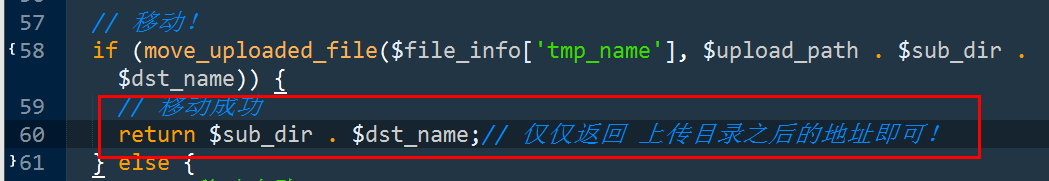
### 创建子目录



### 移动到子目录



### 返回文件中包含子目录



测试：



## 安全性

### 对类型的限制

$\_FILES中的type的信息，不是PHP检测出来的，而是浏览器提供的。

因此：PHP也需要检测文件类型才可以，不能完全信赖浏览器数据！

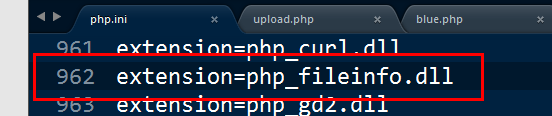
PHP检测文件的MIME类型

（文件内容中，存储相关的类型）



PHP提供 fileinfo 相关函数（方法），来完成

#### 开启fileinfo扩展

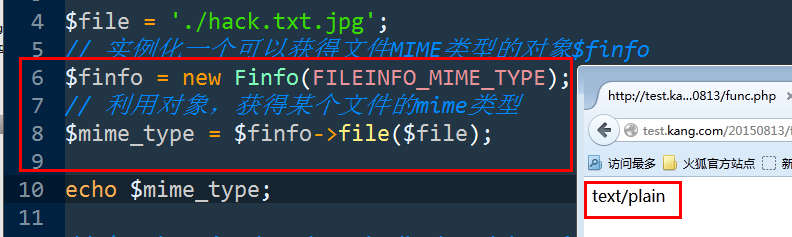


Apache httpd restart

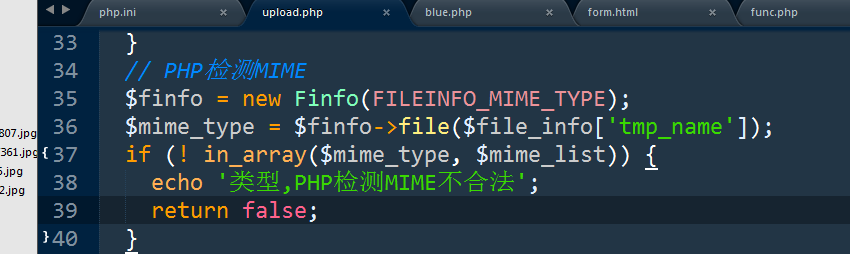
提供了面向对象的语法：

#### 获取MIME类型

过程如下：



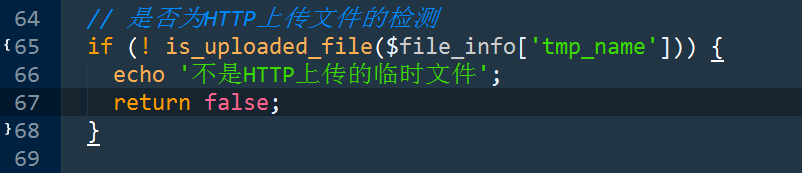
#### 用在上传方法中



### Is\_uploaded\_file();

判断 临时文件，是否为真实的上传文件！

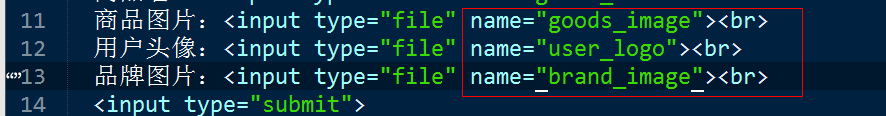
在移动前，判断下：



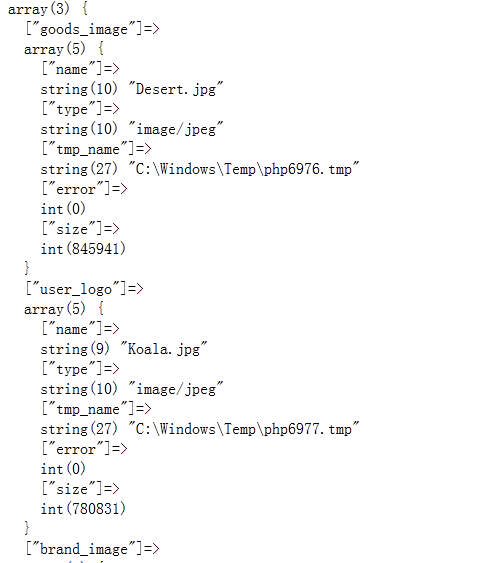
# 多文件上传

$\_FILES结构。

## 情况一表单文件域元素name值不同！

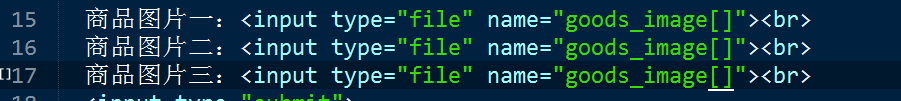


在$\_FILES中，每个元素，对应一个5个元素的数组：



此时，需要处理那个，就使用哪个数组即可！

## 情况二：文件域name相关，数组方式命名

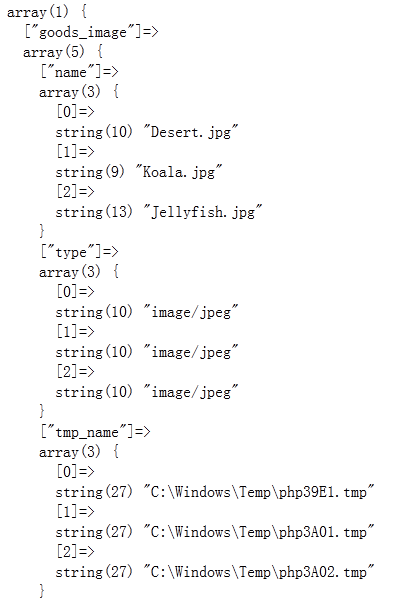


此时$\_FILES的结构如下：

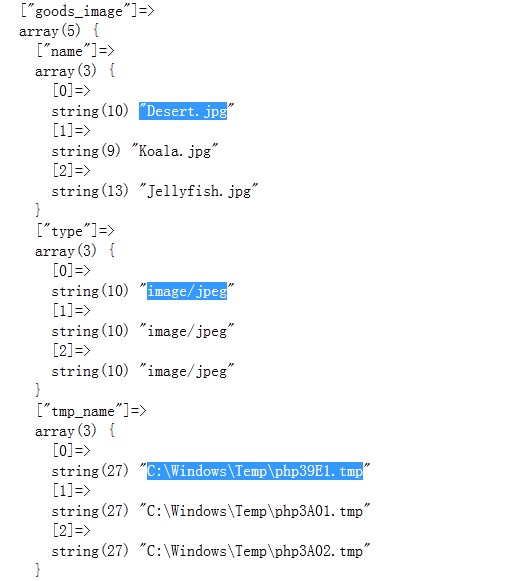
将 相关的整理到一个元素中，该元素对应 5个元素(name, type, tmp\_name, error, size)的数组。

每个name，对应三个值

每个type,tmp\_name, error, size都是如此：



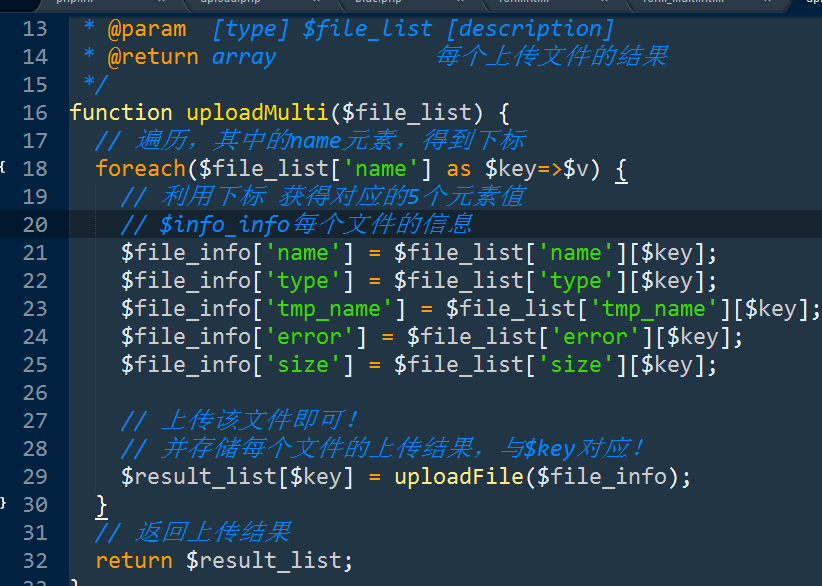
此时，分别整理出来每个文件的5个信息，将每个文件都上传。



下标一致的属于一个文件，只要获得每个下标，就可以通过下标获得对应的5个元素值！

编码实现

定义一个可以完成多文件上传的函数，针对于该情况：



测试

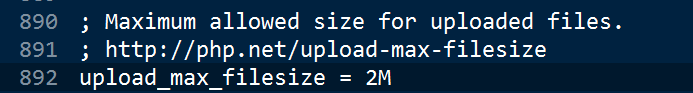


# 错误类型

Error，表示错误

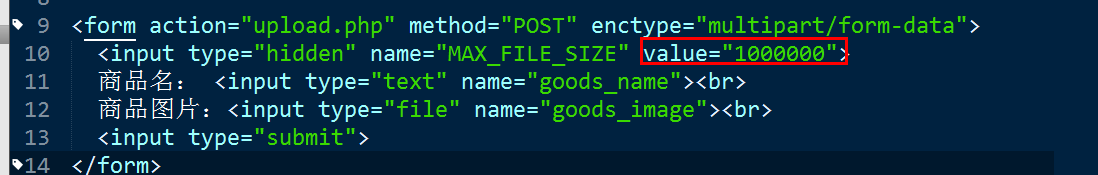
## 0：没有错误

## 1：文件超过了PHP中对上传文件大小的设置



## 2：文件过大，超过表单中元素

MAX\_FILE\_SIZE



精确到每个表单层面！

## 3：文件没有上传完

## 4：没有上传文件

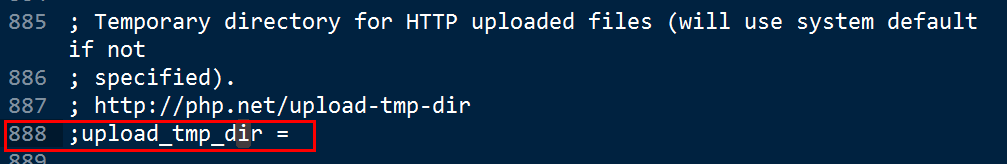
## 5：上传的文件大小为0，上传的为空文件

约定，不是错误！

PHP 没有定义 5 这个错！

## 6：临时上传目录未找到

PHP可以更改临时目录：

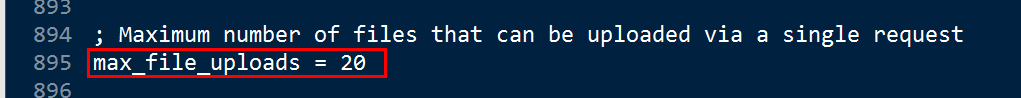
Php.ini  
 

## 7：临时文件写入失败

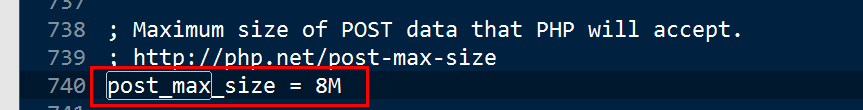
磁盘空间不足，权限不允许！

# 相关配置

## 上传文件数量



## POST数据大小



Post数据总大小。

超过总大小，PHP不处理post数据，$\_POST$\_FILES就是空数组！

Tip：文件仅仅可以以post的方法提交！

HTTP不适合上传超大文件！

# 商品添加表单[功能]

## 控制器

后台，新建 控制器 GoodsController 完成所有商品相关的操作！

Application/back/controllers/GoodsController.class.php



增加商品添加表单：



## 模型

不需要模型层！

## 视图

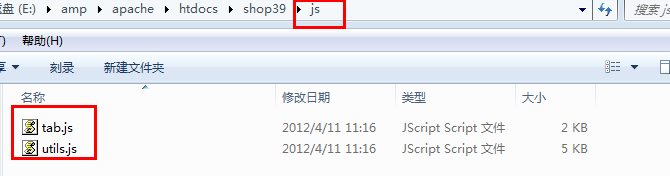
增加模板：

Application/back/views/goods\_add.html

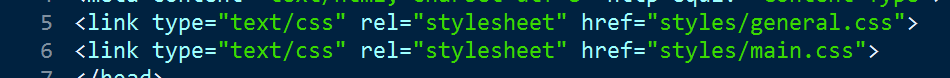
载入需要的js：



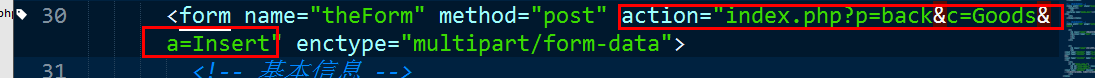




注意 css样式，和图片的位置！



Form表单的提交方法和位置：

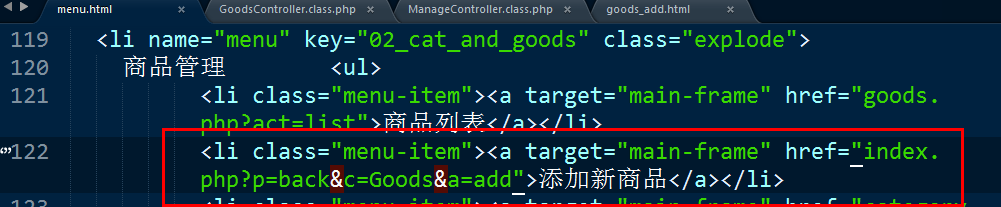


## 测试

修改 左侧菜单链接地址！

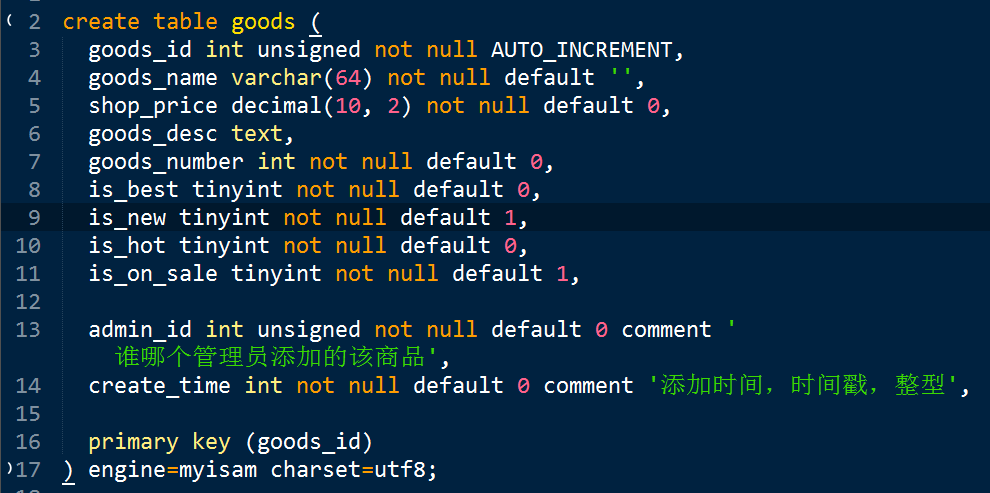
模板：

Back/view/menu.html





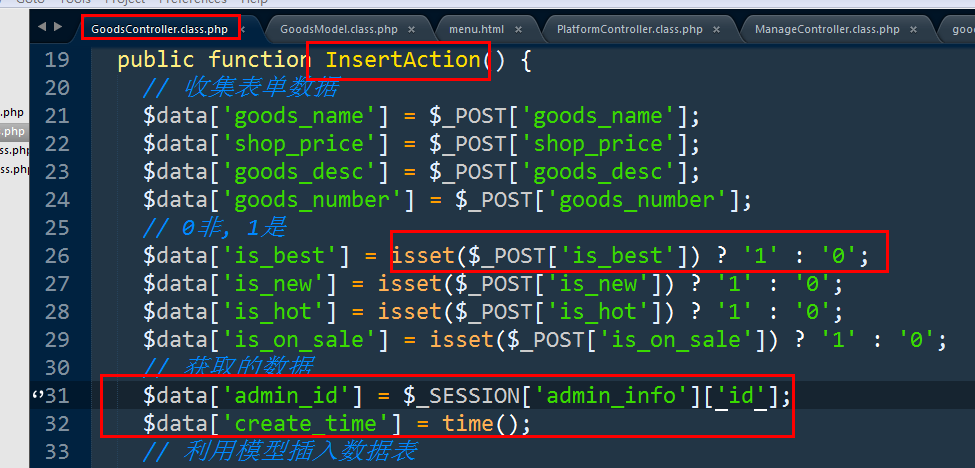
# 创建商品表



# 处理商品数据[功能]

## 控制器

收集数据：



模型插入：利用插入结果跳转！



## 模型

需要增加goods操作模型，

Application/back/models/GoodsModel.class.php

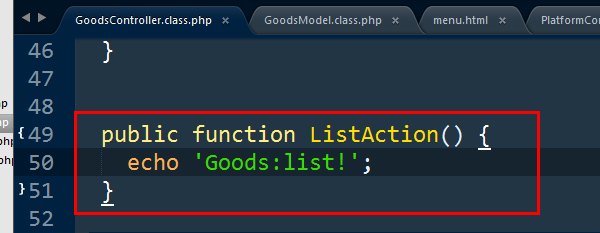


## 视图

不需要！

## 测试

增加简单的列表动作！



插入成功：



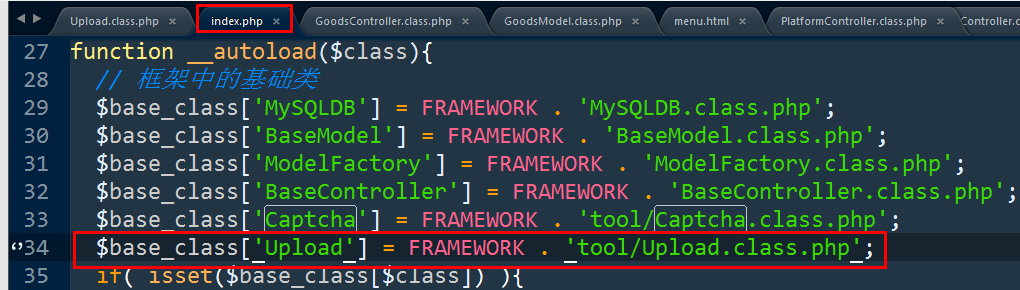
# 商品图片的处理

## 上传工具类

封装成类！实例化使用！



完成自动加载：index.php



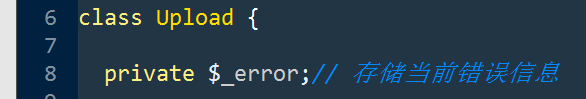
## 实现上传的方法

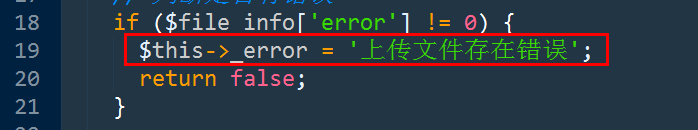
上传单个文件，上传多个文件！

将定义好的方法，在上传类中实现：需要更新的地方：

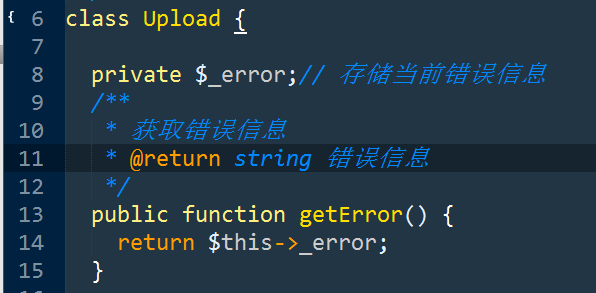
### 一：错误信息

增加属性存储错误信息



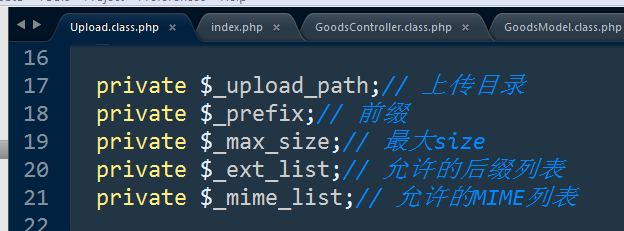


增加一个方法获取错信息：

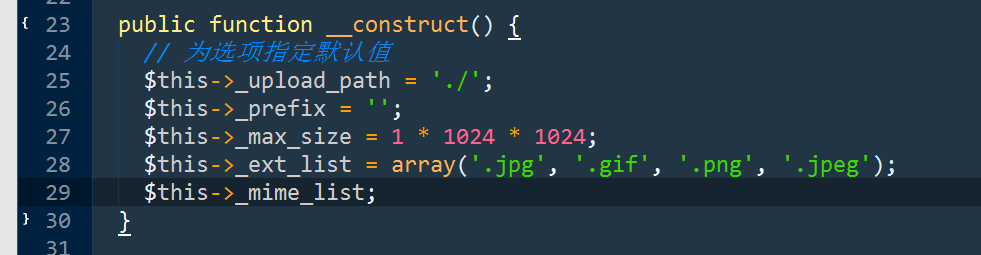


### 二：将可变的做成配置

使用属性来表示！



通过 构造方法，进行默认值初始化

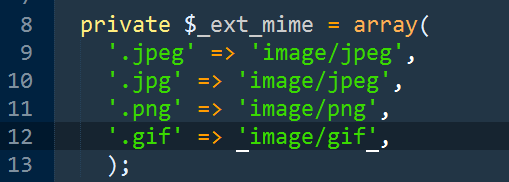


注意 后缀名和mime的关系：

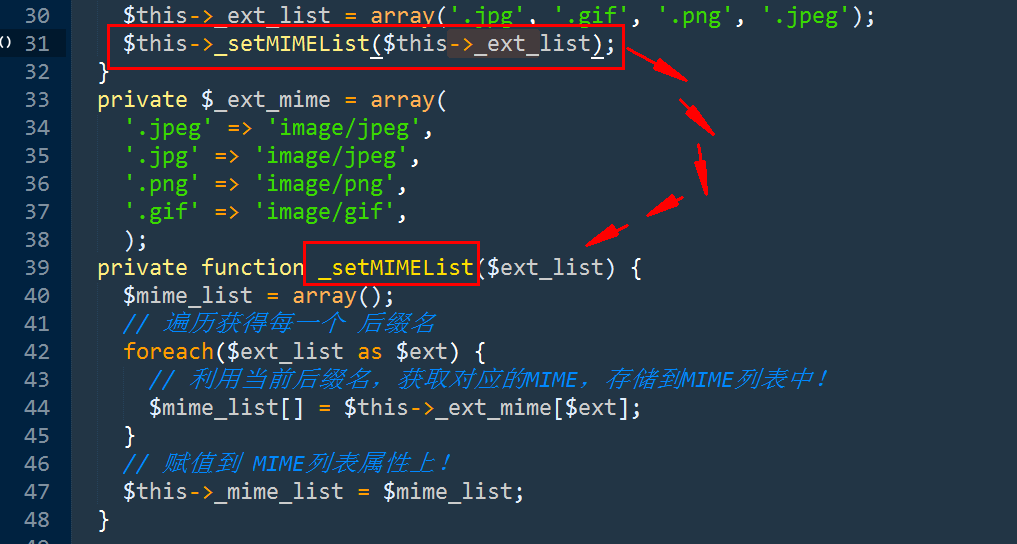
要求用户指定 后缀名，我们帮着推算出来mime：

参考： 后缀名 与 MIME 映射关系！

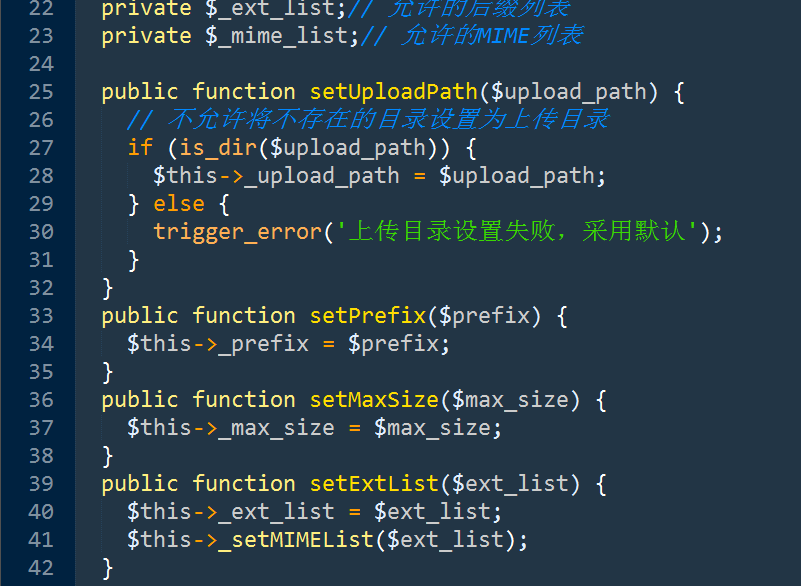
通过提供对照表：提示，实际中所有的类型最好都由对照！



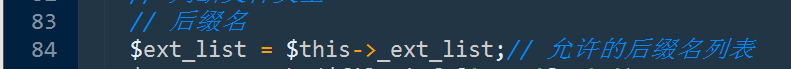
完成映射查找：

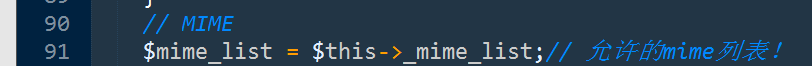


提供修改这些配置属性的方法：



需要时，使用这些属性：







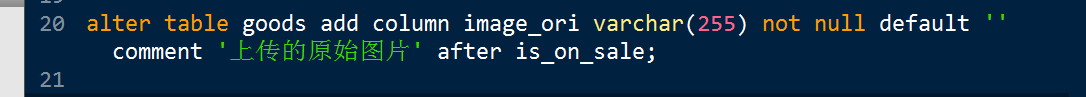


## 使用工具类完成商品图片上

如何将商品与图片关联上？

增加一个字段，存储商品图片地址。（地址由上传的结果）



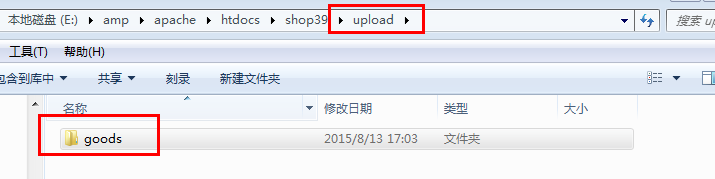


上传

Back/GoodsController->InsertAction()

完成上传

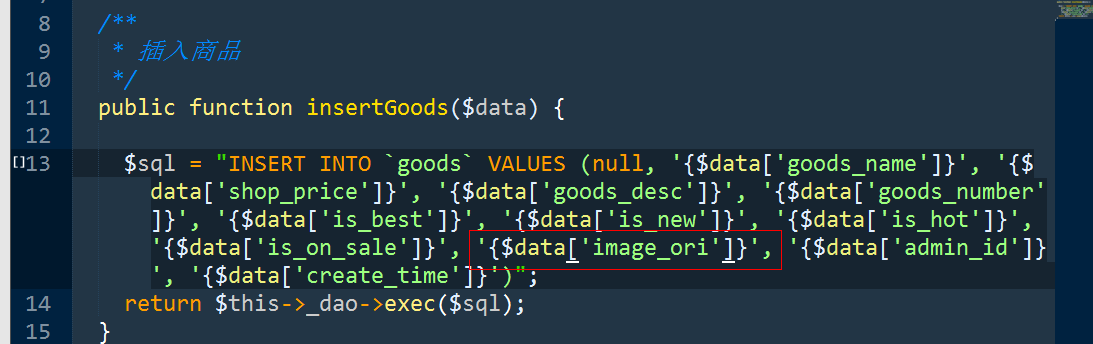
选择上传目录地址：设计如下



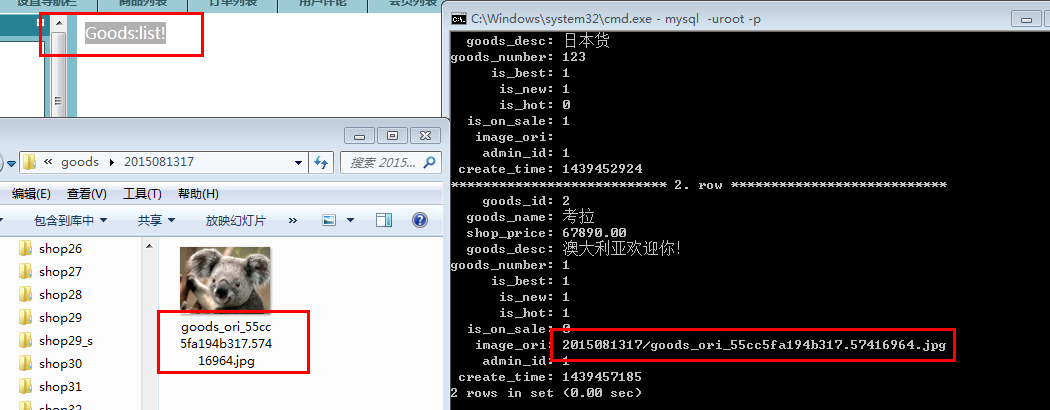


模型中，增加对上传地址的处理：

Back/models/GoodsModel->InsertGoods();



测试



# 目录操作

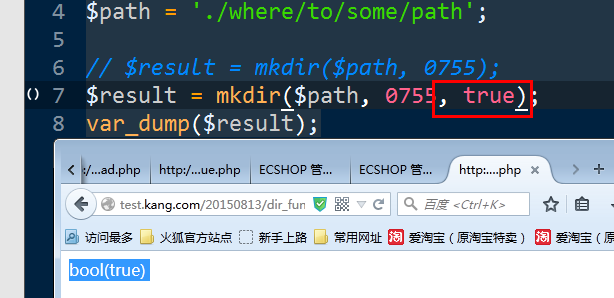
## 创建

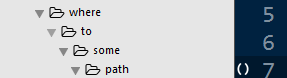
Mkdir(目录地址, 权限, 是否递归创建)

默认不支持递归创建：



第三个参数写true：





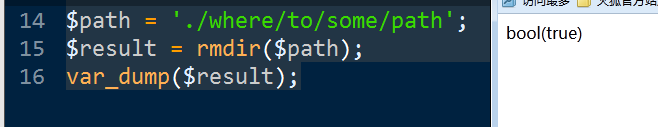
Tip：最近PHP增加的参数。

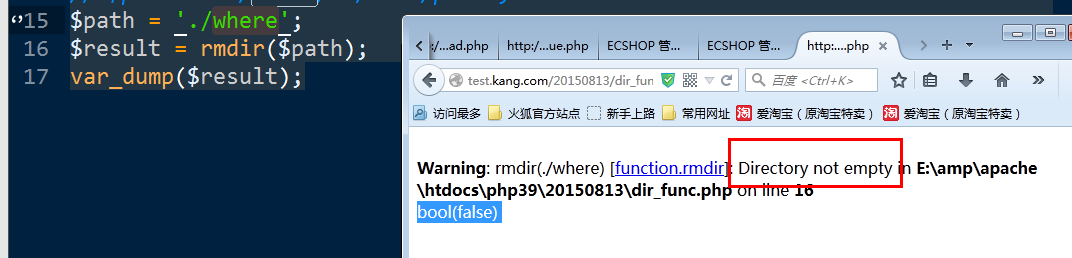
第二个在windows被忽略！没用！

在linux有用！

## 删除

删除结果= Rmdir(目录地址)





不允许删除非空目录！

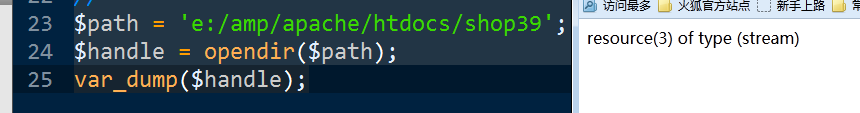
## 获取目录内容

### 句柄 = Opendir(目录地址);

打开目录句柄。

句柄：PHP程序与文件系统 数据流通道！

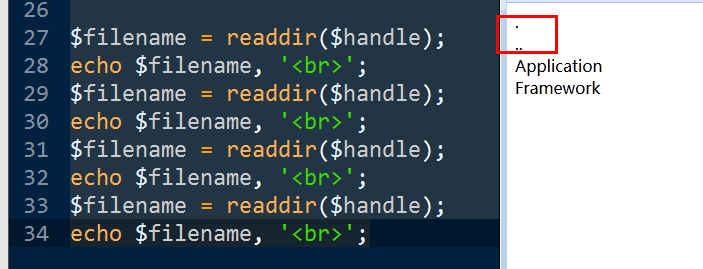


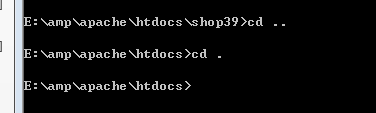


### 文件名 = Readdir(句柄);

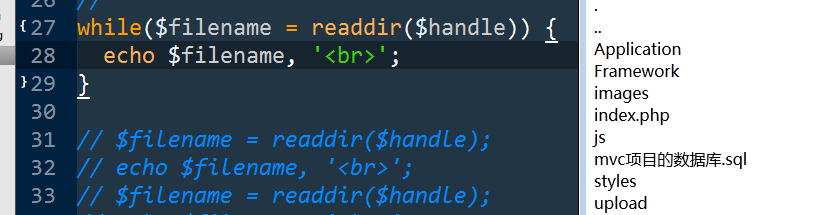
通过句柄，从目录中读取一个文件（包括文件和子目录），一次读取一个文件，并向下移动文件指针！

注意，每个目录下都存在 ., .. 虚拟目录。表示 当前目录，和 上级目录。

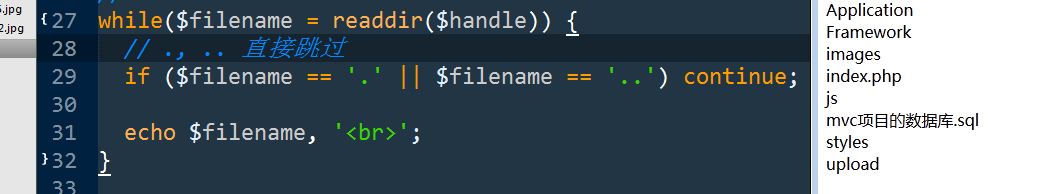




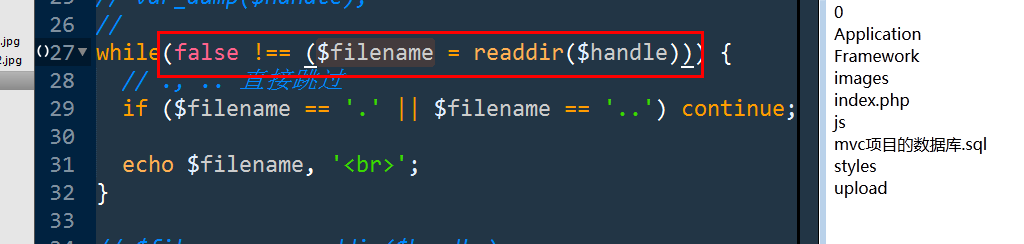
配合循环结构就可以获取目录内的全部内容：



通常要 处理 .,..:



0文件名：



### Closedir();

关闭句柄



## 重命名（移动）

函数：

Rename(原始地址, 目标地址)

针对文件和目录！

