面向对象关键字：

class

public,protected,private

static

const

self,parent,static

extends

implements

abstract

interface

final

new

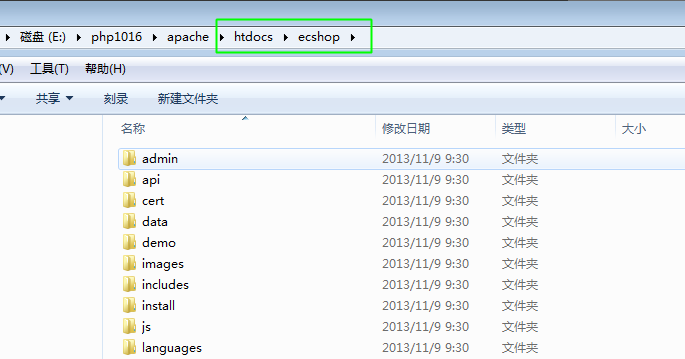
clone

$this

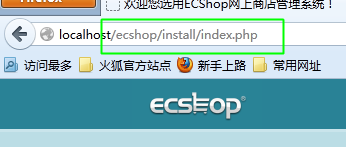
# 介绍项目开发

## 安装ecshop

解压压缩包后，获取其中upload子目录的所有文件，将其放置到可以被浏览器请求到某个目录（建议改名）



请求，index.php即可！



会自动跳转到安装页面（或者可以请求install/index.php）

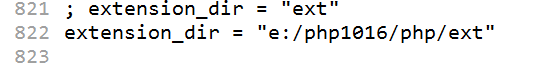
**配置php**

ecshop需要mysql扩展与gd扩展（图片处理扩展）支持！







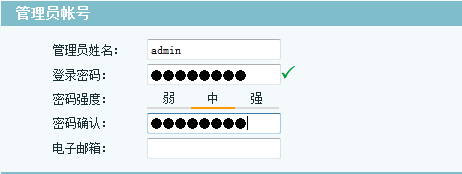


**配置系统**

配置数据库信息

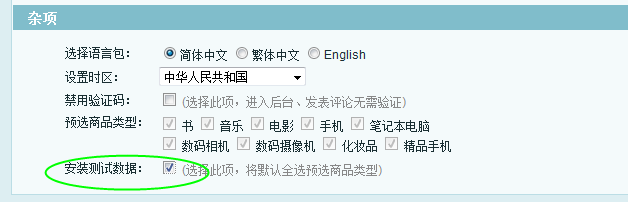


配置ecshop管理员的信息



其他配置：

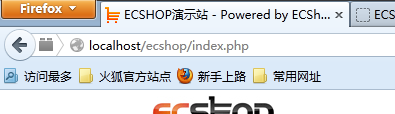
建议安装测试数据



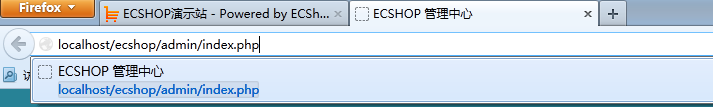
安装即可

安装成功：

前台地址：



后天地址：



## 项目的开发周期

### 需求分析

### 设计

编码设计，应该如何实现！

两大方向：

1. 二次开发，基于已有产品，做后期自定义处理！产品，已经实现了主体业务逻辑功能代码！

商场：ecshop，ecmall，shpex,zen-cart，es-commerce

CMS，内容管理，博客，wordpress，dedecms，drupal，

论坛，discuz，phpwind

1. 基于框架开发

框架，只有基础公共代码，而没有业务逻辑的产品，就是框架！

Thinkphp，YII，Zendframework，symfony，ci，fleaphp，quephp，

### 编码实现

### 测试，调试，修改

### 上线，生产阶段

### 维护，升级，改版

# 代码的进化流程

## one，混编

嵌入式脚本语言PHP

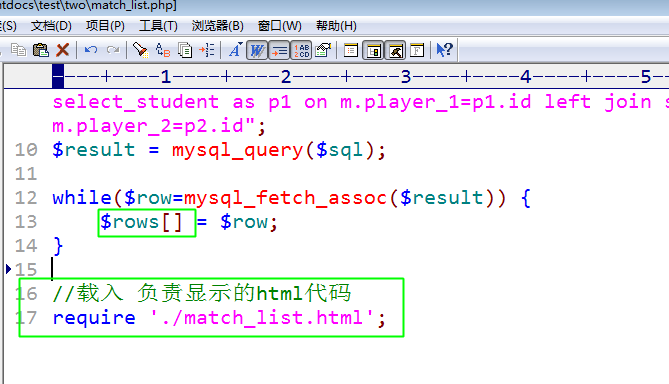
html与php混编的编码方式

## two，显示和逻辑相分离

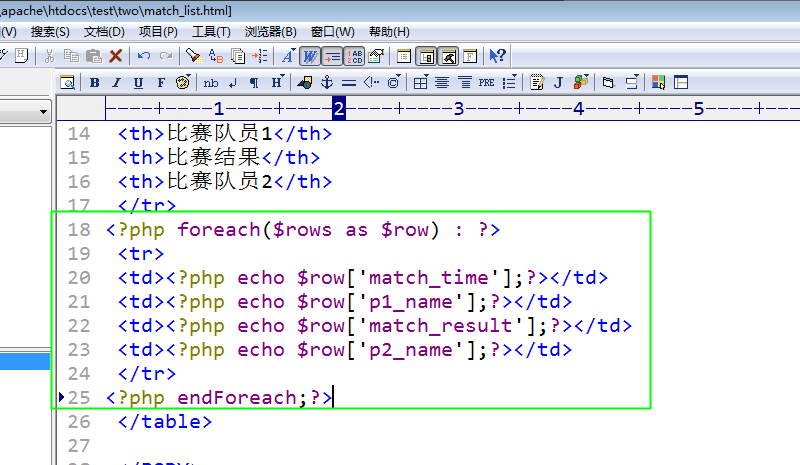
最后，需要将显示和逻辑的结果放在一起！

需要在 php页面，将html代码 载入才可以！

此时的请求，一定要去请求负责逻辑的php代码：



需要在html代码中，遍历数据，将数据展示！



代码分工后，易于维护和管理！从管理项目的角度出发的！

### 模板文件

注意当前的html页面，出现：显示格式部分由html代码充当，而数据部分需要通过php来实现。数据，特指数据的展示。（数据的获得实在负责逻辑的php代码中完成）

这样的html文件，就被称之为模（mu）板文件（template）

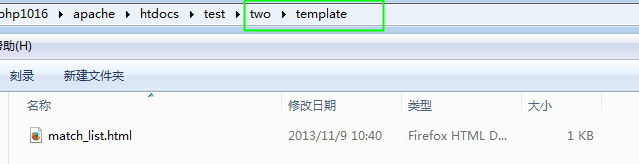
### 限制用户访问php逻辑文件

办法多得是，典型的两种：

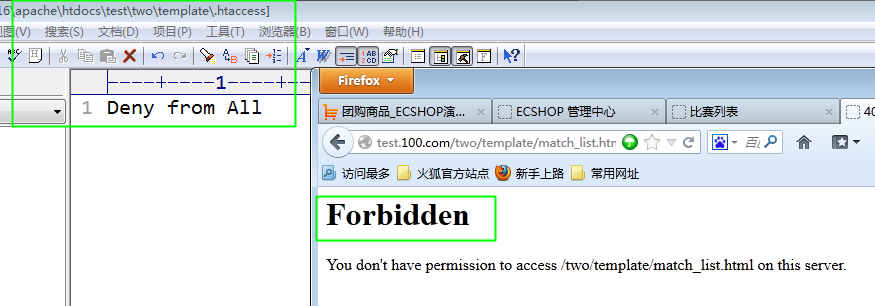
1，.htaccess

deny from all

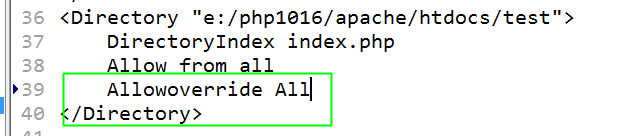
利用Apache的对访问的控制，将某个目录设置成禁止访问。将所有的模板都放入到目录内！



控制该目录的访问：



注意，htaccess生效的前提是，Apache对目录开启了allowoverride :



2，将不允许用户访问的文件，包含模板文件，都放置文档根目录之外！

浏览器请求只能看到文档根目录下的文件。

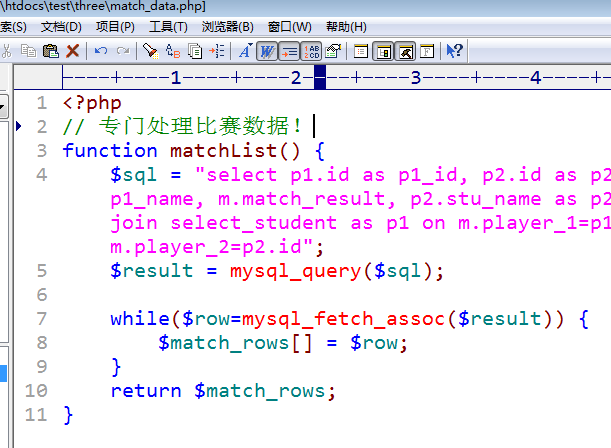
## three，处理数据（处理业务逻辑的代码），从php代码中分离

场景，在完成一个其他页面，在其他页面使用到了相同的功能！

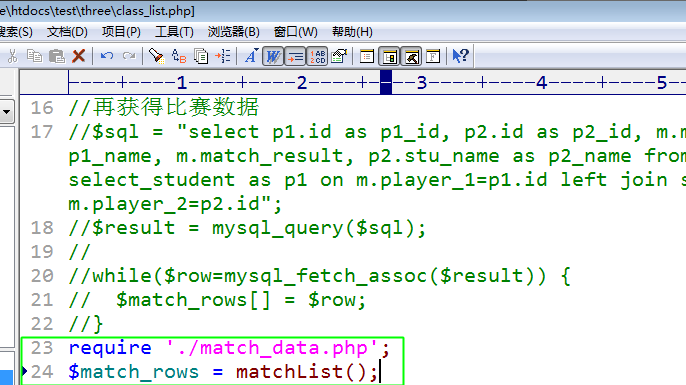
出现的数据操作逻辑的代码的重复！

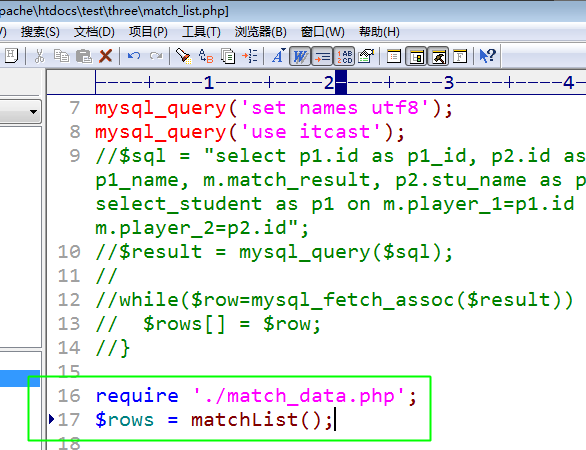
解决：将重复提取！

新增专门处理数据的php代码

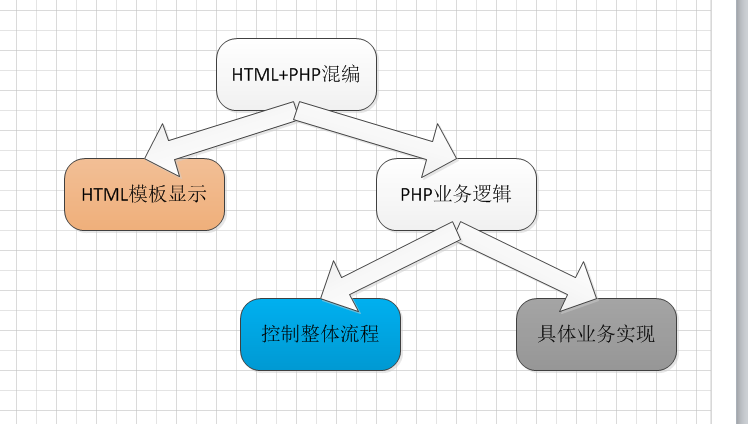


在需要比赛列表时，调用该函数即可！





one，two，three的进化流程：



# mvc项目分层，设计思想

mvc，项目的分层思想，指的是完成一个业务逻辑，需要三大部分，分别是：

1. 具体的业务逻辑实现的部分，数据操作，称之为 M，Model，模型！
2. 具体显示样式的实现部分，html+css+js，称之为 V，View，视图！
3. 负责整体流程控制的部分，负责调用M和V。成为 C，Controller，控制器！

参考four的实现

项目如果是按照上面的分工方式实现的，项目就是按照 MVC的设计思想设计的！

（只是一个项目分层思想，不决定语法）

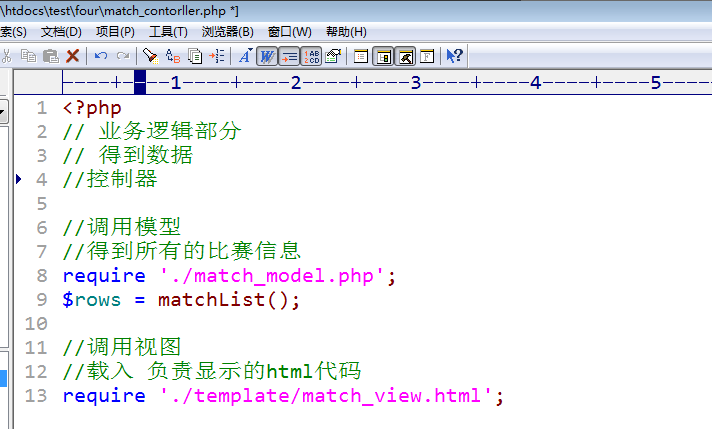


典型，mvc的实现：

用户浏览器去请求控制器

控制调用模型实现业务逻辑数据处理

控制器调用视图，利用模板将数据展示！



mvc==thinkphp，zend-framework

mvc思想，

mvc框架，基于mvc设计思想所实现的框架就称之为mvc框架！

# 模型的处理

## 模型类

业务逻辑的处理，最终都是通过操作数据库完成！

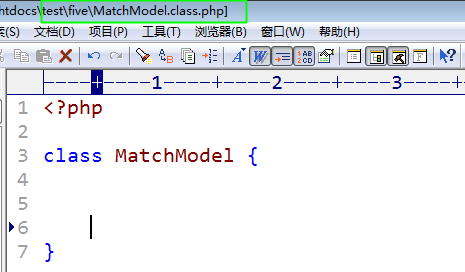
mvc分层中，典型的模型是：

每个模型，是一个对象！一个模型对象，负责完成数据库中一个表的操作！

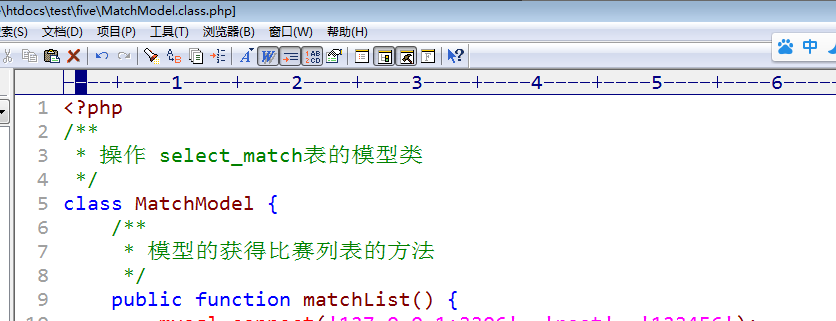


此时，针对于某个表的操作，就应该被封装到相应的模型类中！

例如，操作，match表，会有如下的模型：

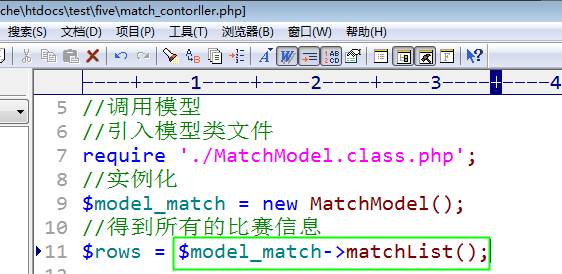


完成比赛列表，在获得数据时，应该通过实例化该模型类的对象，调用其获得数据的方法！



使用该模型类：

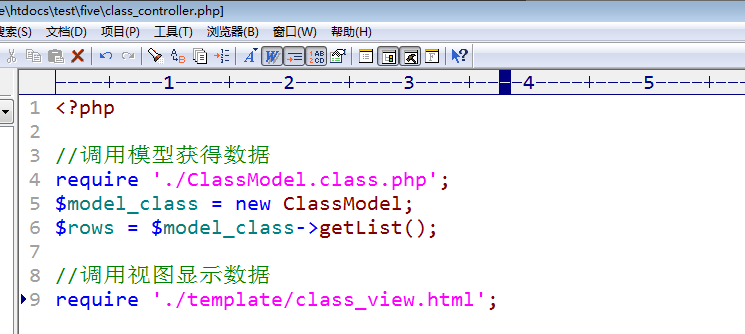
实例化该类对象，调用该方法：



反三：班级列表

**一，控制器**

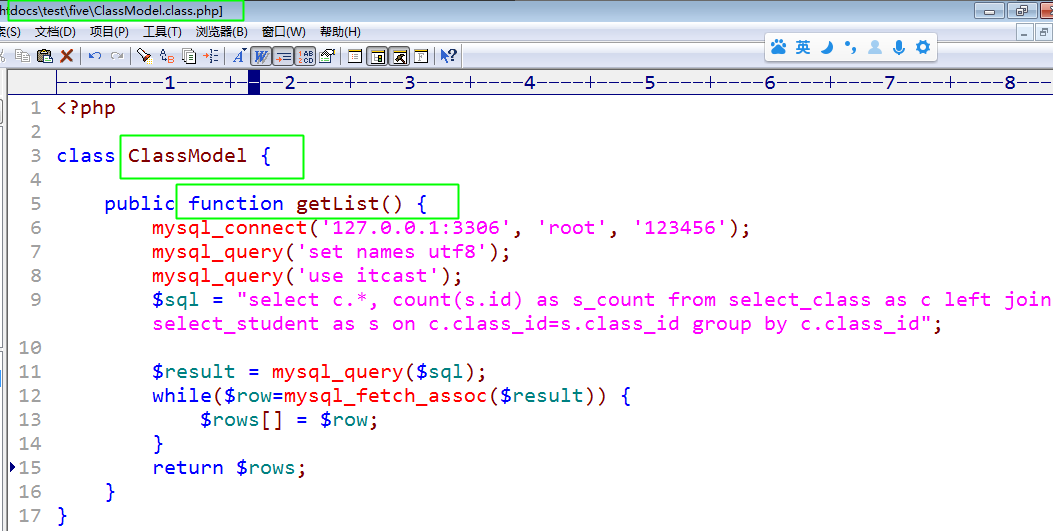
增加一个控制器文件，并完成向 模型和视图 发出命令



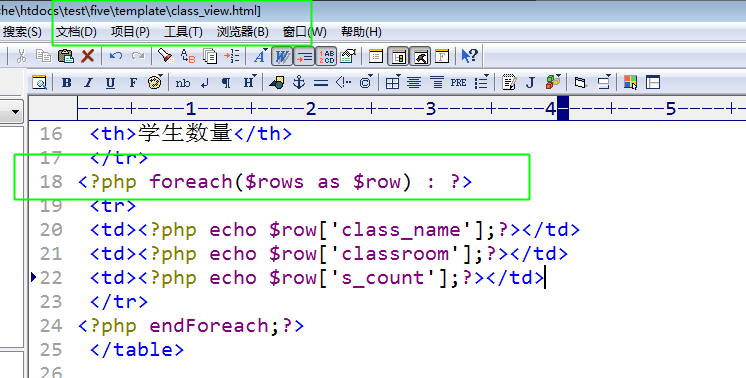
1. **模型**

建立一个ClassModel模型类文件

并增加一个getList方法，返回所有的班级数据



1. **增加一个视图模板**

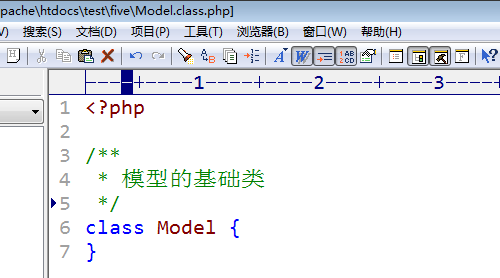


## 基础模型类

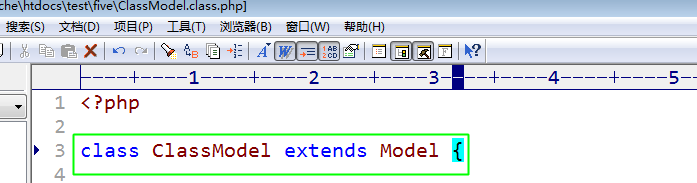


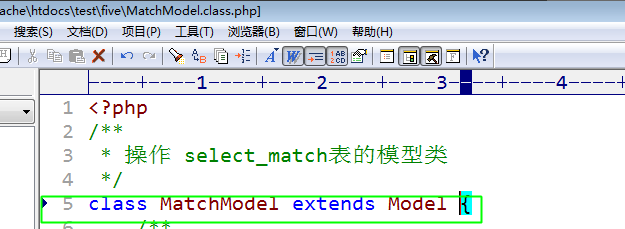
模型的公共操作的提取

增加一个模型基础类：

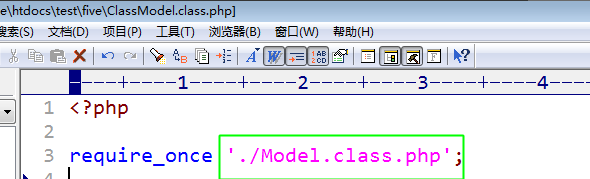


所有的模型都应该继承自该模型基础类：





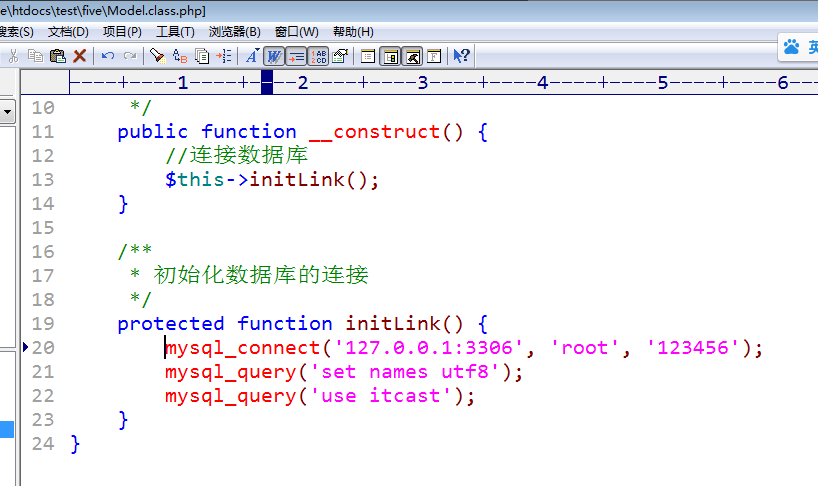
别忘了载入Model类才可以：



因该在实例化模型对象时，就完成数据库的连接！

在基础模型类中，增加构造方法，被子类所继承。需要在构造方法中，完成数据的连接

Model.class.php



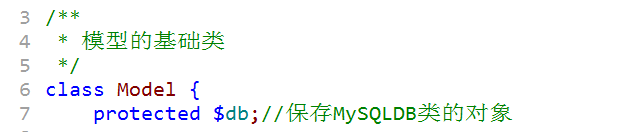
此时，一旦实例化就完成了连接工作

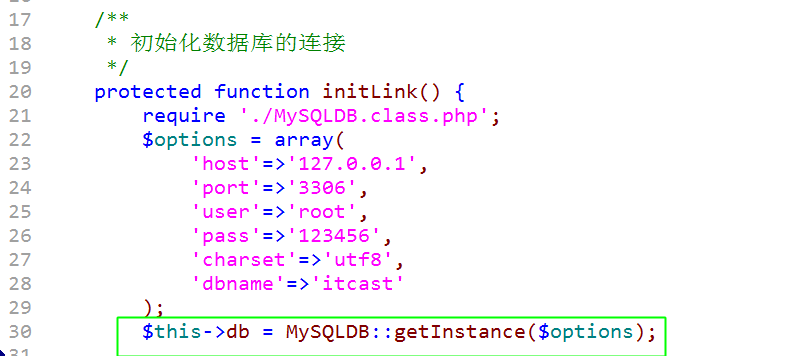
## 模型类利用MySQLDB类对象操作数据



此时，在基础模型类中，初始化数据库连接，其实就是实例化MysQLDB类的对象并，保存到当前的对象的$db属性上！

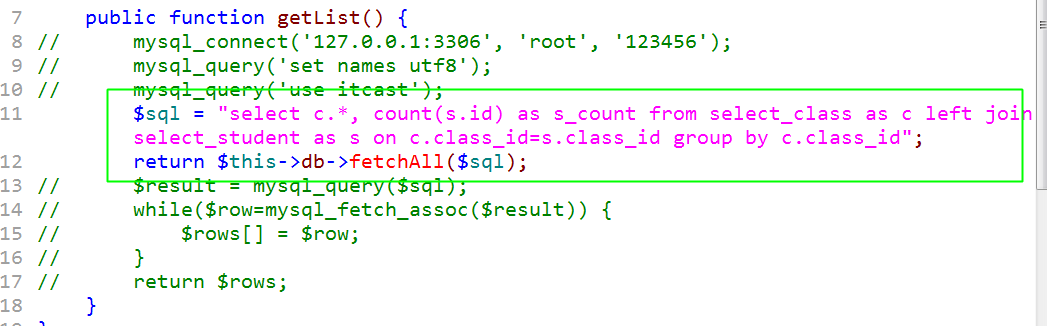
Model.class.php





具体的模型中，如果需要执行sql，就应该使用模型对象$db属性中保存的MySQLDB类对象来完成：

ClassModel.class.php



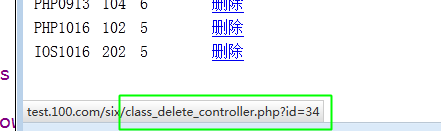
# 控制器的处理

场景：

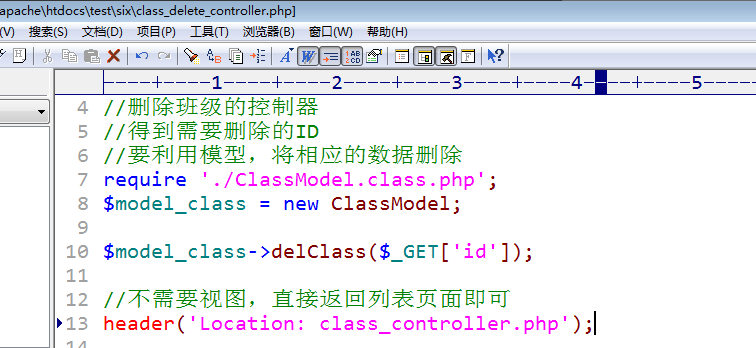
增加一个班级的添加功能（删除，修改），如何完成

删除，控制器，模型，视图

**控制器：**

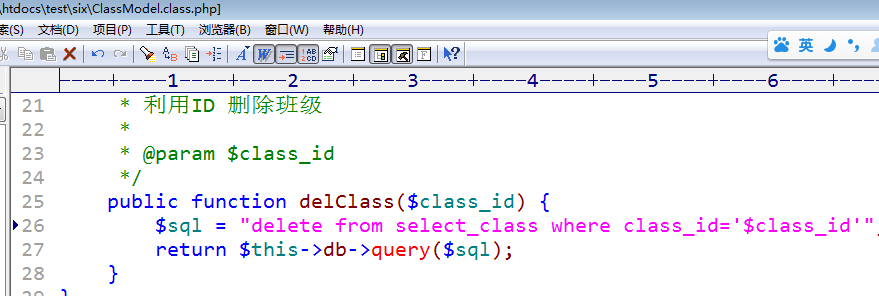


class\_delete\_controller.php



**模型**

在 ClassModel中则增加方法即可



**视图**

不需要！

## 增加模块，控制器代码的集合

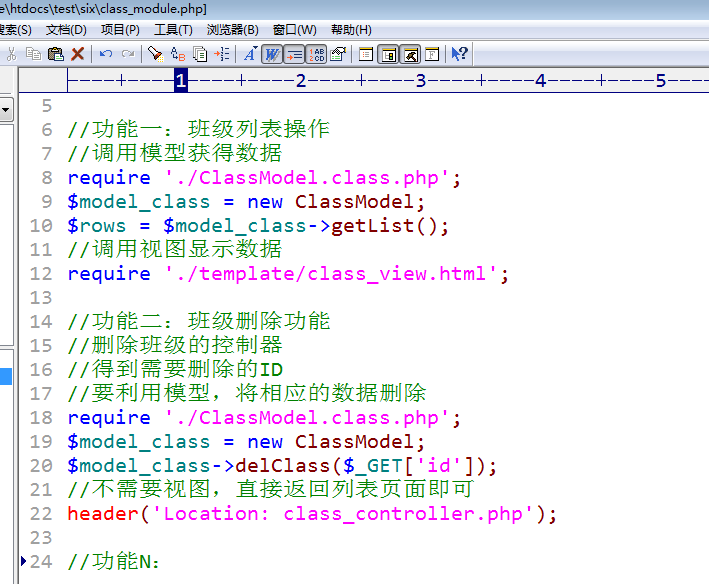
不能是每个操作都增加一个控制器文件，那么如何管理这些操作呢？

使用模块的概念！

模块，module，类似的操作的集合，例如，班级管理模块！

增加班级管理模块文件：

一个模块文件，就是多个功能的集合！



此时，完成班级操作，都要请求，class\_module.php这个文件！

问题是，如何判断，当前的请求应该执行哪个具体的工作？

应该在请求 该模块文件时，增加一个参数，表示当前用户浏览器想要执行的操作！

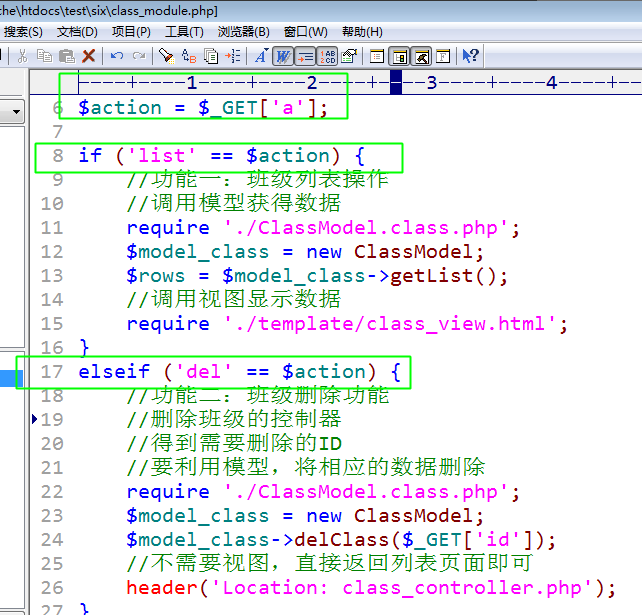
a（action），动作的！一个操作称之为一个动作！

class\_module.php?a=list 列表

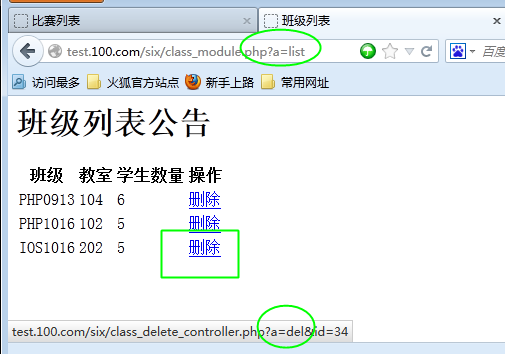
class\_module.php?a=del 删除

典型的应该利用流程分支控制：

class\_module.php

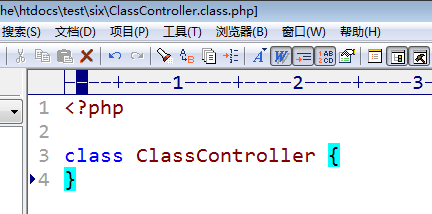


此时，在浏览器请求时：需要：



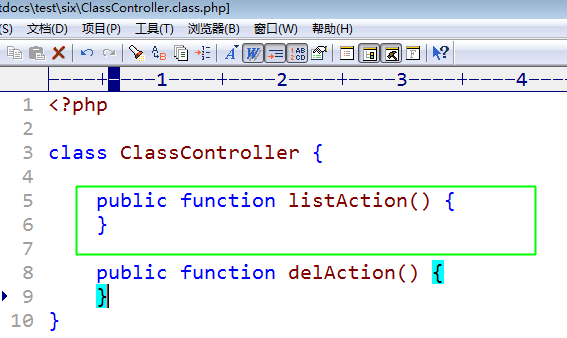
## 控制器类

将刚刚所完成一个控制器模块文件，升级成一个控制器类



模块内的每个功能，变成控制器模块类的一个方法：

ClassController.class.php



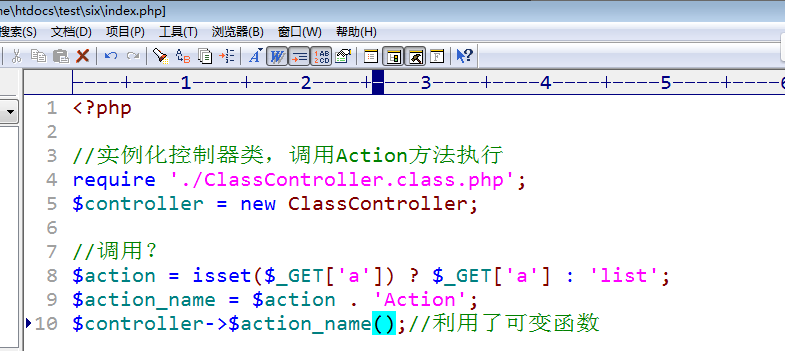


此时，需要对控制器类进行实例化，还需要确定调用哪个方法！

需要找一个额外的文件，进行classController类的实例化！

新增文件：

index.php



此时，我需要看到列表：



## 入口文件，index.php

由于所有的控制器都是类！

所有的请求都需要从index.php开始，因此被称之为项目的入口文件！

主要负责，接收到请求，并且根据请求参数，将该请求分发到某个控制器的某个具体方法动作上。 因此入口文件也被称之为：请求分发器！

由于功能上整体流程控制一部，也是控制器。因此也被称之为： 前端控制器

如果有多个控制器类，需要多个实例化控制器类的操作！

如何在一个文件中，完成动态的控制类实例化的工作？

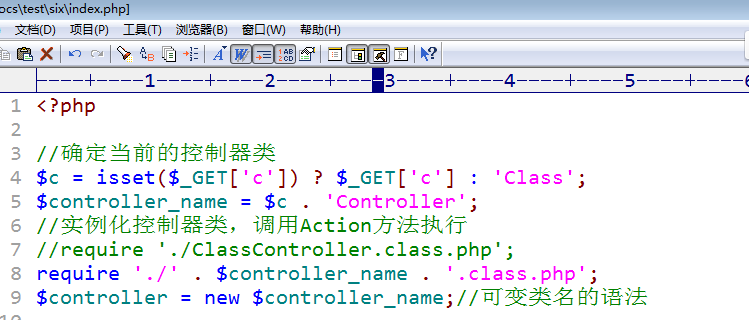
类似于action动作的区别，增加请求参数，表示当前所请求的控制器类！

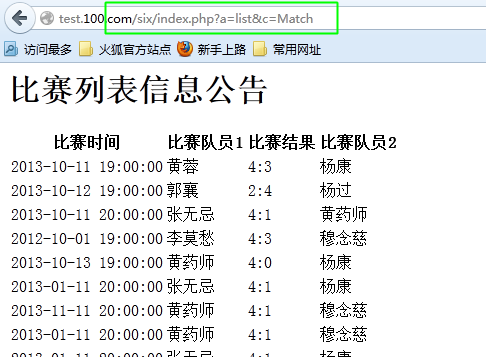
c,controller,表示当前所请求的控制类

例如  
班级列表： index.php?c=Class&a=list

学生删除：index.php?c=Student&a=del

因此，需要在实例化控制器类时，需要先判断当前的c参数，再来确定：





注意：

一旦完成了上面规划：

要求，用户的所有请求，都要经过index.php入口文件，分发给控制器的动作再往下执行！

index.php?c=xxx&a=yyy



# 预习

框架的概念。将共用的代码提取出来！

# 作业

利用six版本，实现 比赛，学生，班级信息展示！