# 控制器层典型实现

## 控制器类

依据功能的相关性，将一系列相关的功能，使用一个控制器类来处理，而该控制器的每个方法，就对因某个功能。

注意：控制器是按照功能划分的。（而不是像模型一样，按表来划分）

比赛相关功能控制器类：



## 前端控制器（请求分发器，入口文件）

以上的listAction()操作应该如何被调用呢？

实例化，并掉用方法即可！

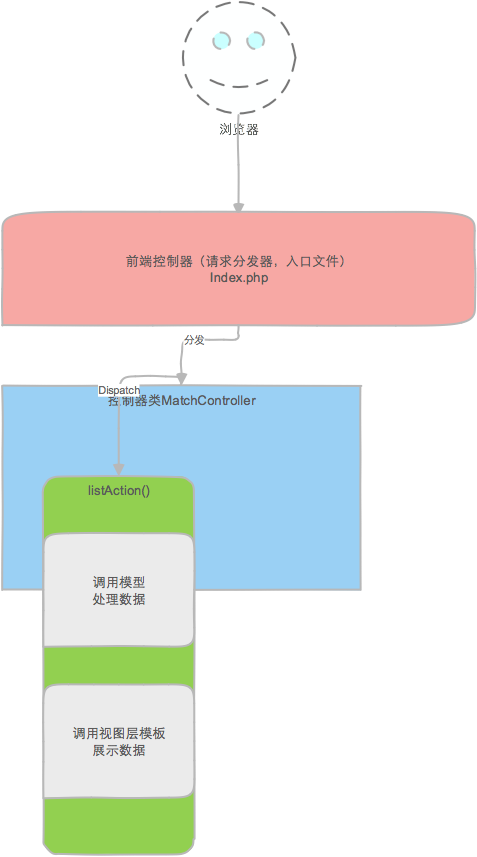
在哪里实例化或调用呢？

增加一个可以实例化并调用控制器方法的文件。





逻辑流程：



### 动作action分发参数：a

如何做到一个前端控制器，可以调用一个控制器类不同方法动作呢？

在请求前端控制器index.php时，向其传递a参数，表示当前所需要执行的动作名，例如：

功能：比赛列表：

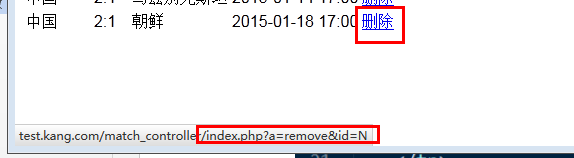
Index.php?a=list

功能：比赛删除：

Index.php?a=remove

Tip：链接地址的形成，应该在HTML代码中就确定好了，再存在一个默认动作即可！

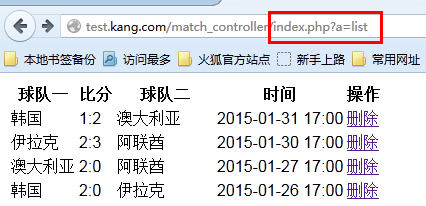


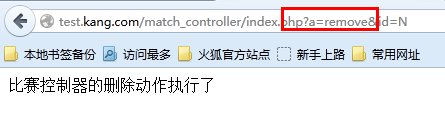


Index.php判断a参数，执行相应的动作即可：



测试：





### 控制器controller分发参数：c

如果需要执行其他控制器的某个动作应该处理处理前端控制器？

在请求前端控制器index.php时，向其传递c参数，表示当前所需要执行的控制器类名，例如：

比赛的列表动作：

Index.php?c=Match&a=list

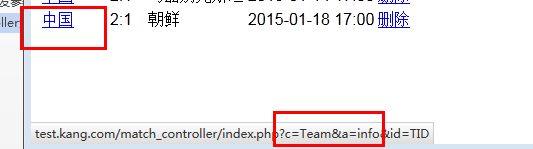
球队的信息动作

Index.php?c=Team&a=info&id=TID

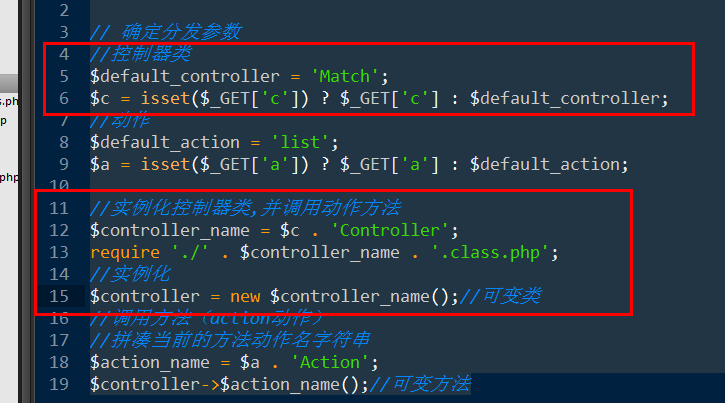
c,a在HTML的链接地址中，自动形成好的







Index.php对c分发参数进行处理：



测试：



### 使用常量存储分发参数

思考：

在一次请求周期中，所请求的控制器名（当前控制器），和所请求的动作名（当前动作），是否会发生更改？

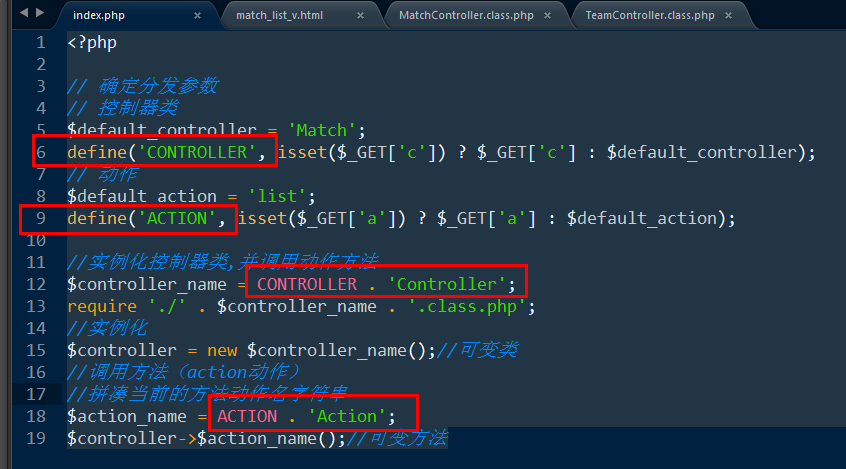
不会发生更改！

如何在语法的层面上，保证在一次请求周期内，当前控制器与当前动作不会发生改变？

存储在变量中，不能保证。

应该使用常量，进行存储当前控制器及其当前动作，保证

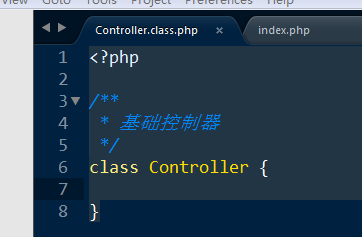
Index.php中实现：



## 基础控制器

增加为 所有的控制器提供基础代码控制器类：

Controller



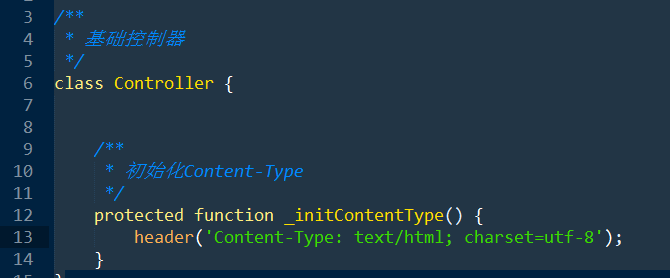
其他控制器，继承自基础控制器：





### 初始化Content-Type的基础操作

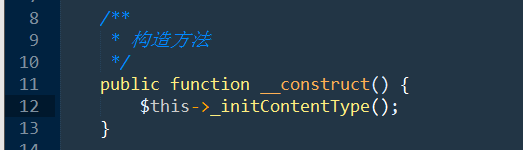
Controller.class.php



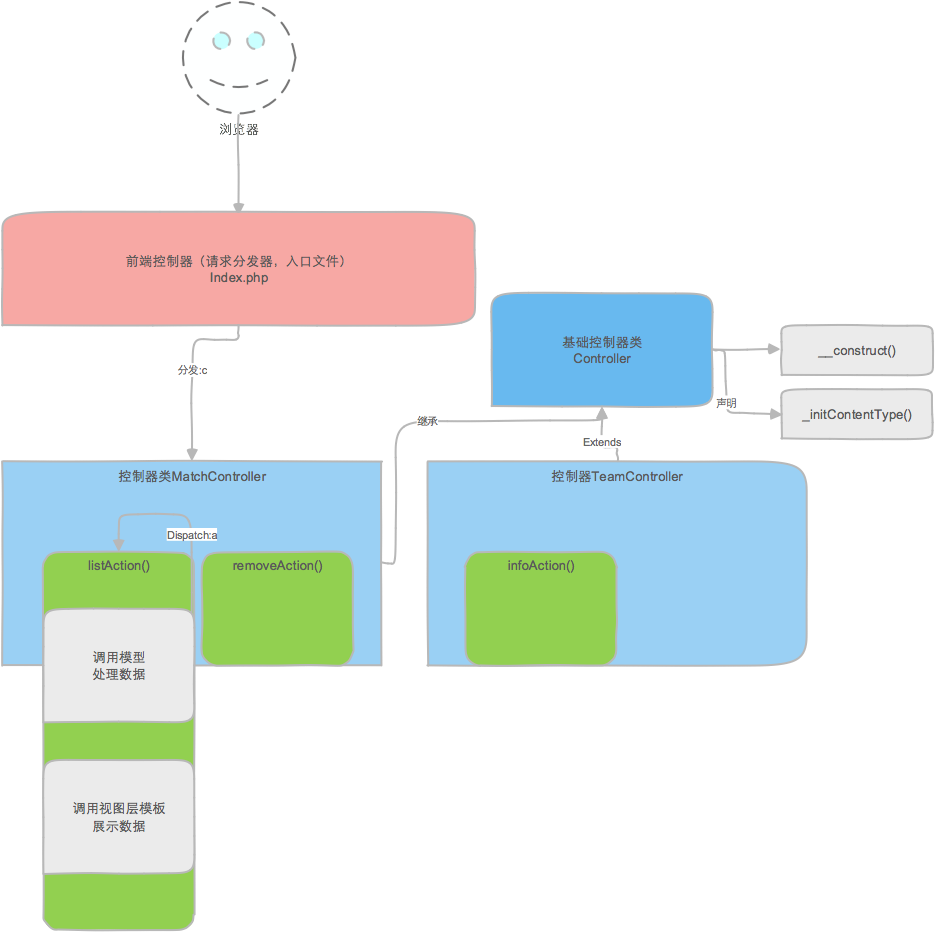
调用：

在实例化控制器对象时调用，在构造方法中：

Controller.class.php



## 逻辑图例



注意：

浏览器的请求地址（URL），都是固定的形式：

Index.php?c=Controller&a=action&

称之为单入口模式

项目对外提供的任何功能，都是由某个控制器类的某个方法来实现的。

# 目录布局

存在项目的根目录 shop34

## 框架代码与应用程序代码划分

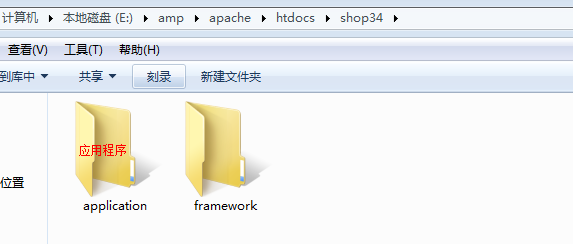
框架代码：

在各个应用程序间可以通用的代码。

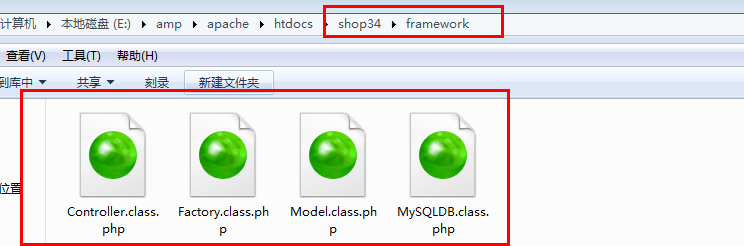
应用程序代码：

当前项目的业务逻辑实现的代码。

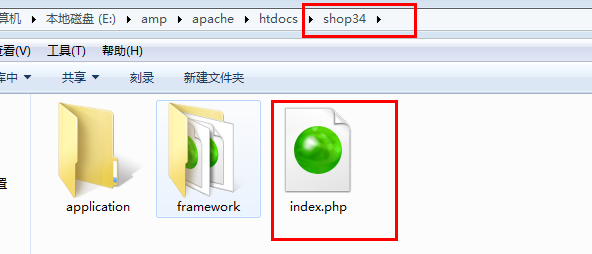
分别创建两个子目录存储，框架与应用程序代码：



示例：框架内的代码：



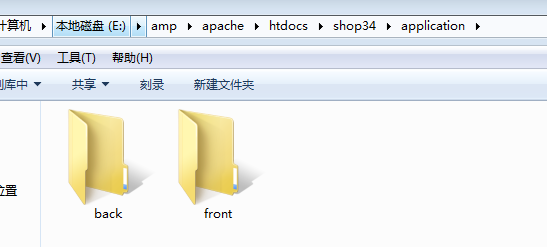
## 入口文件在站点根目录



## 平台platform的划分

功能的聚合！

（也称为：模块module，分组group）

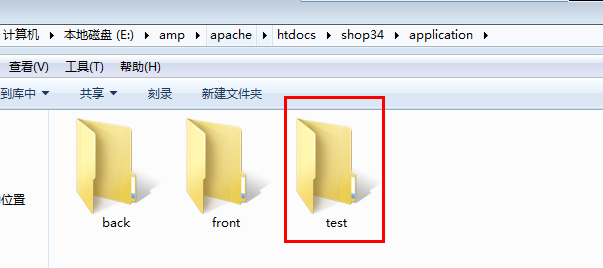


辨析：

前台，后台。

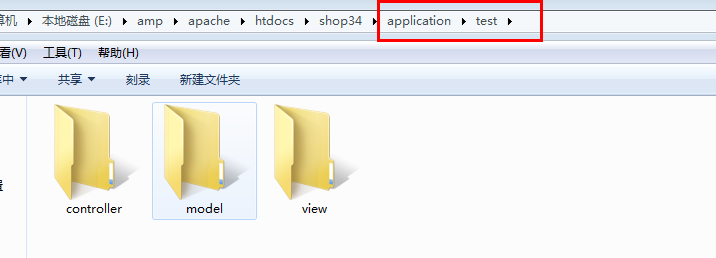
美工，程序。

需要将实例代码，作为测试test平台进行处理的话，应该为test平台建立平台子目录：



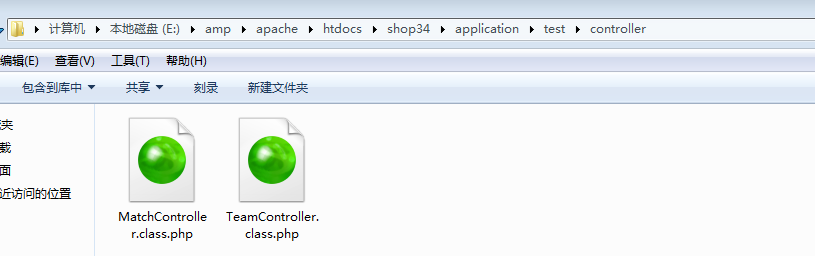
## MVC层的划分

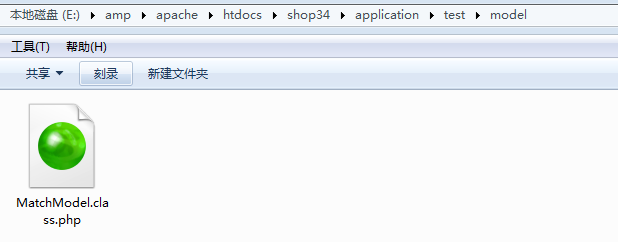
功能由MVC三部分实现的，因此将某个平台下，分成：

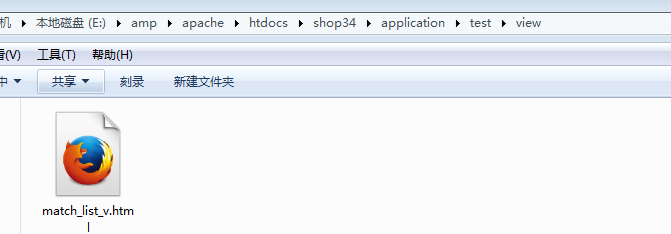


对当前平台下的 控制器，模型，视图 文件进行分别管理！

例如：







## 整体目录布局

Shop34/

Index.php入口文件

Application/应用程序目录

Front/前台

Back/后台

Test/测试平台

Model/模型

View/视图

Controller/控制器

Framework/框架代码目录

MySQLDB.class.php数据库操作类DAO类

Factory.class.php工厂类

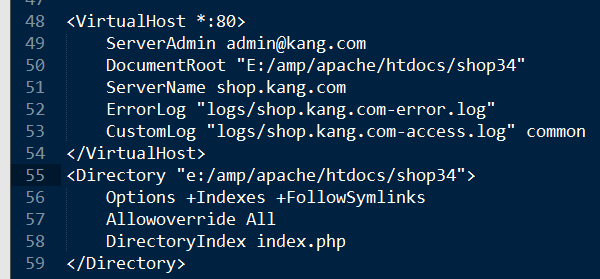
Model.class.php基础模型类

Controller.class.php基础控制器类

## 运行起来

配置一个可以访问到shop34的虚拟主机：

Apache/conf/extra/httpd-vhost.conf



Apache restart

Hosts：



### 修改需要载入的文件路径

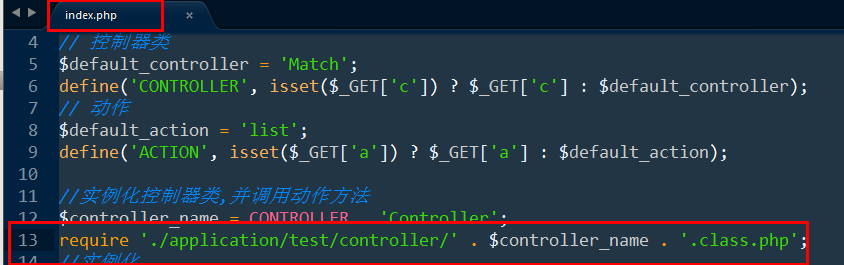
如果写相对路径，注意当前目录。

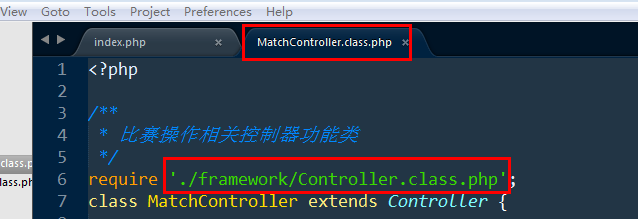
当前目录不是代码所在文件的目录（代码不是在文件中运行，而是在内存中）。

当前位置由浏览器所请求的脚本来确定（当前PHP处理脚本时，第一个处理的文件位置，被视为当前工作目录）。

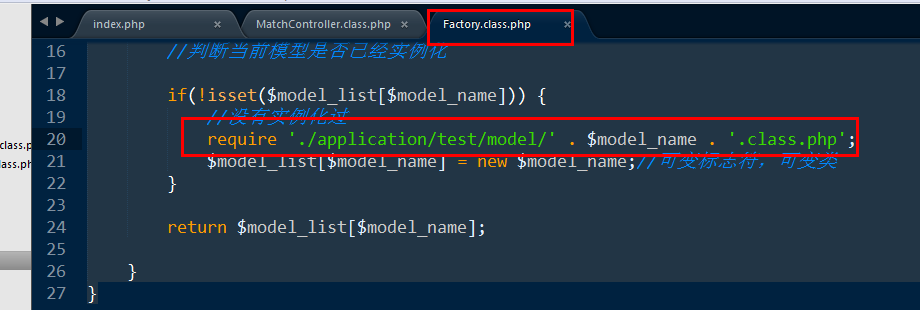
可见，单入口程序，所有的相对地址，都是相对于入口文件地址而言的。

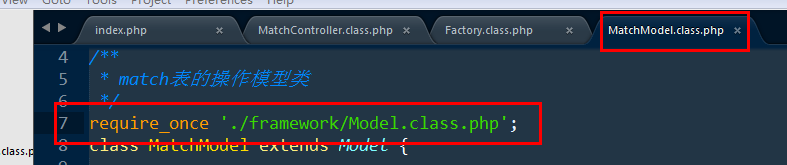
Tip：函数 getCWD()可以用来获得当前工作目录。Current Working Directory

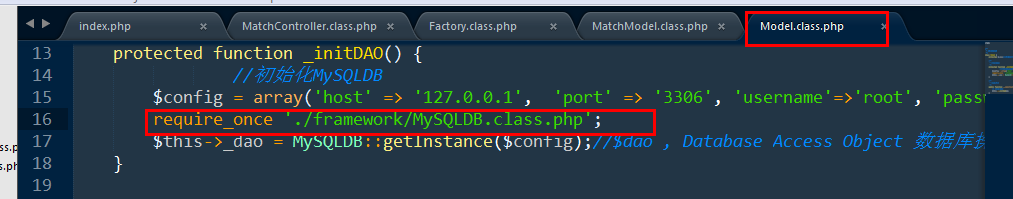














## 分发参数p，确定当前平台

URL：

测试平台比赛列表功能：

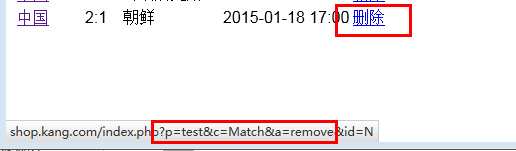
Index.php?p=test&c=Match&a=list

前台用户注册

Index.php?p=front&c=User&a=register

生成正确的URL：





入口文件index.php处理：





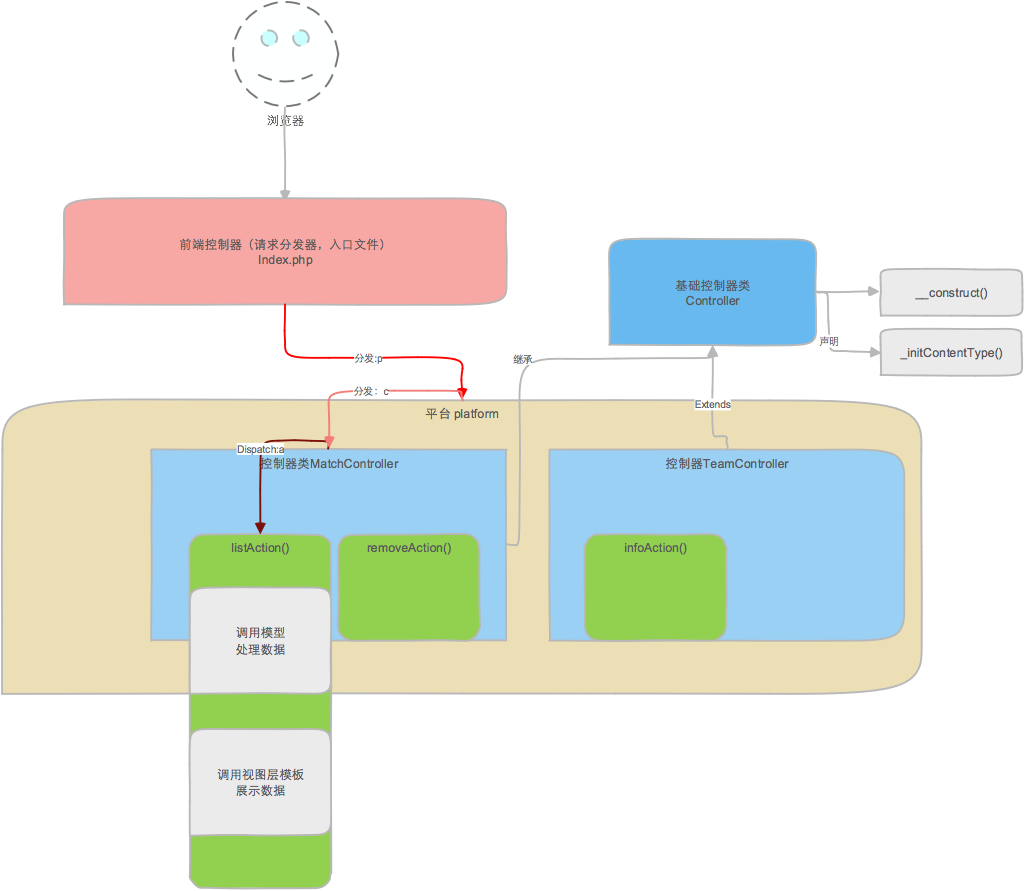
Tip：

分发参数共3个：

P：平台（目录）

C：控制器（类）

A：动作（方法）



# 自动加载

总结：

控制器类（可增加）

模型类（可增加）

框架核心类（确定的）

对于所有的类，分成两方面考虑：

一：可以确定的类。（当自动加载方法确定时，类文件已经确定了）

二：可以被增加的类。（当自动加载方法确定时，没有办法确定）

对已经确定的类，采用最简洁的方式进行加载。

对于需要增加的类，需要通过类名 的规律，完成其位置的判断，进而完成自动加载。

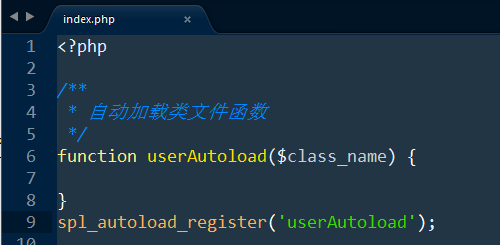
规律是：

XXXController结尾的，控制器类，在当前平台下controller目录将其载入

XXXModel结尾的，模型类，当前平台下的model目录将其载入。

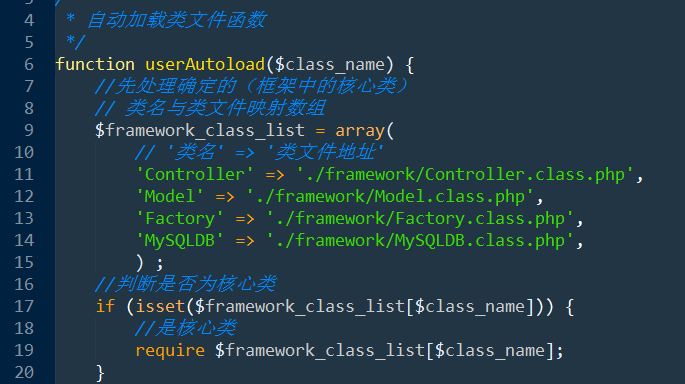
编码实现：

在入口中增加自动加载方法，并注册：



实现 userAutoload：

核心类加载：



可增加类（控制器，模型）：



测试：

将项目所有对类文件的加载都删掉！

# 目录地址常量

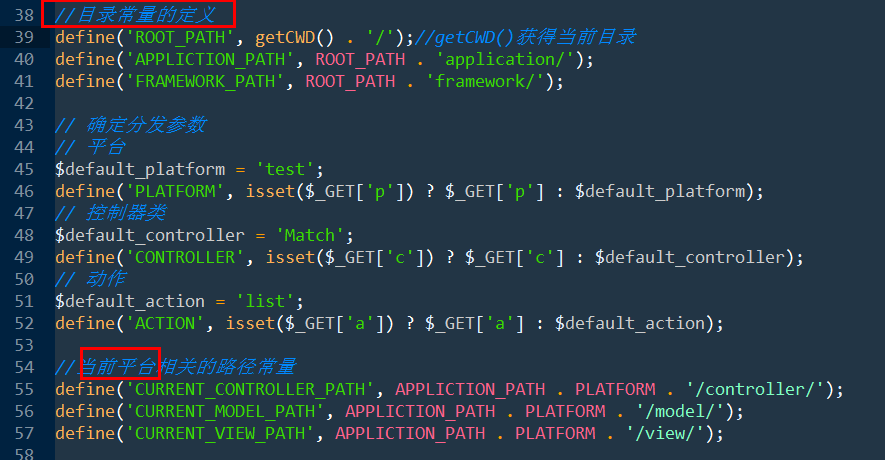
项目中，使用目录常量的形式，管理项目中所需要的地址，通过拼凑，连接操作形成某个位置。

使用目录常量

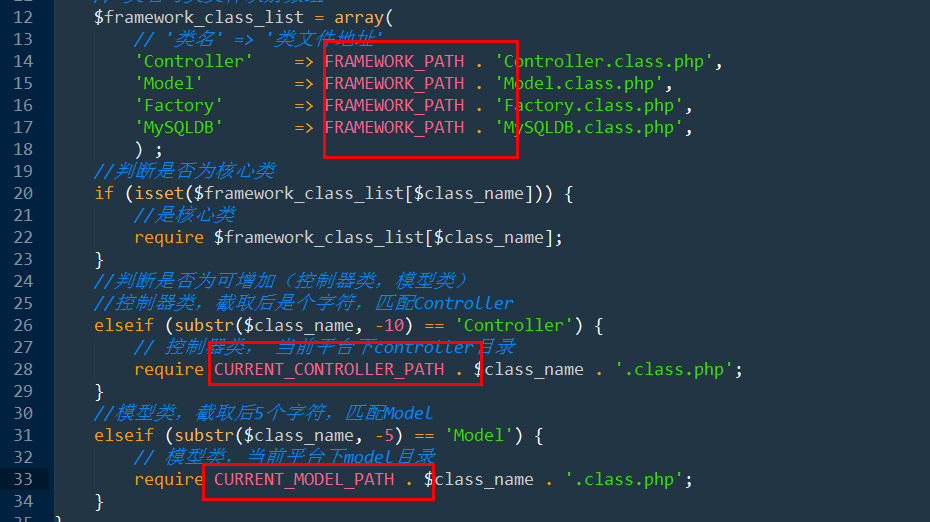
作为项目的初始化阶段，开始初始化目录常量：

先确定根目录，再拼凑确定子目录：

Index.php

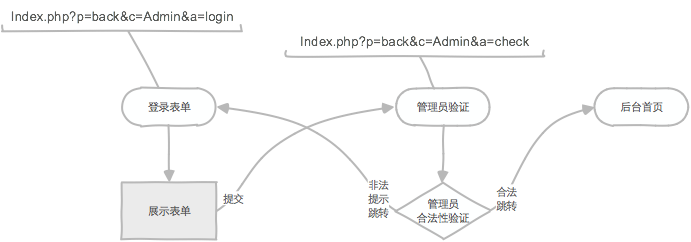


所有的涉及到文件地址的都有常量来完成：





# 登录功能分析

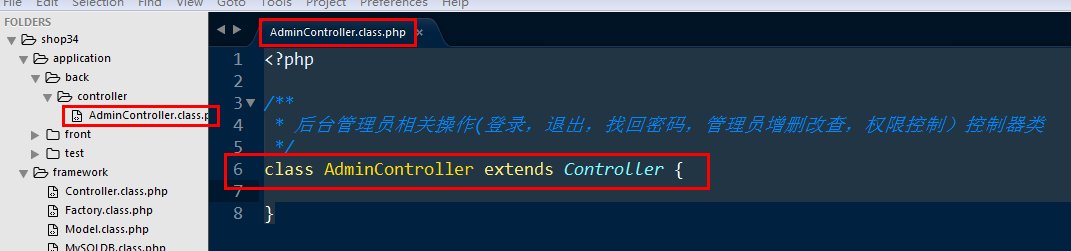


# 功能：登录表单

## C控制器-动作

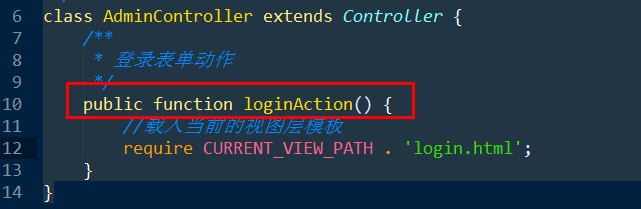
back后台admin控制器loginAction动作完成该功能。

Application/back/controller/AdminController.class.php



增加动作：

完成动作：



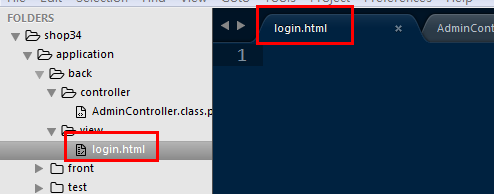
## M模型

不需要！

## V视图层-模板

增加登录表单模板，展示！

Application/back/view/login.html

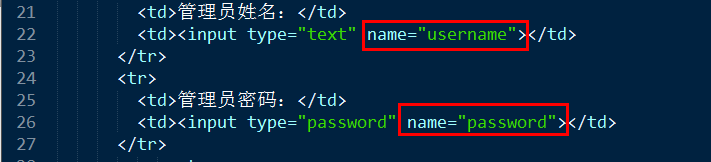


按照ecshop的后台样式展示结构即可！

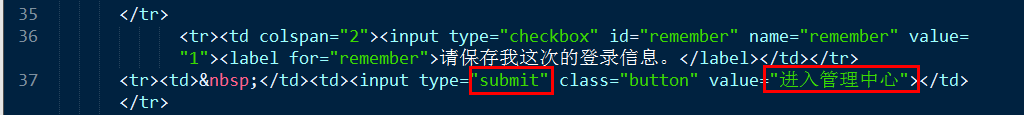
表单action地址：



必要的元素：

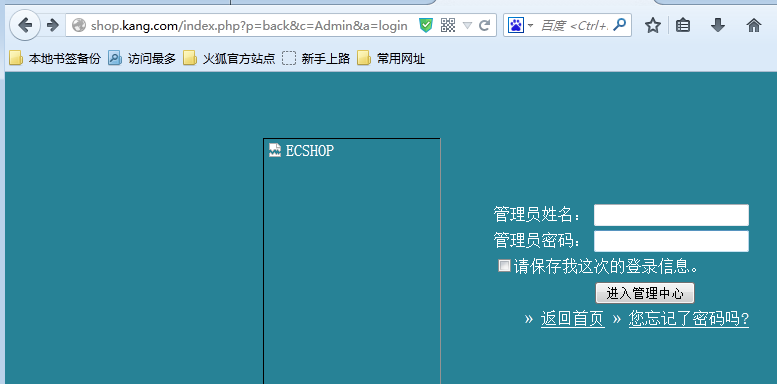


提交：



测试请求该动作：

Index.php?p=back&c=Admin&a=login



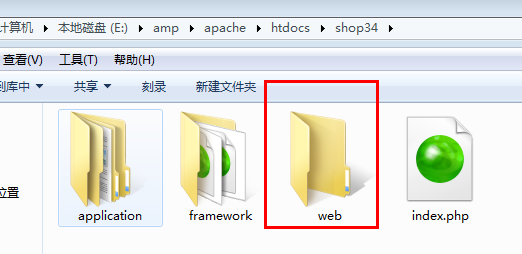
# 处理页面上的图片，CSS，JS

项目中所用的CSS，JS，图片需要在浏览器端请求到，在浏览器解释执行的。

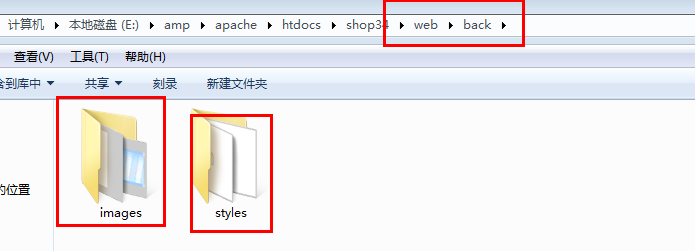
与项目中核心类，模型类，控制器类，模板文件不同，这些不能在浏览器端直接请求到。

应该将css，图片统一管理在一个可以被访问到的地址！

新建一个子目录，专门用于存放哪些可以被浏览器直接请求的资源（JS，CSS，Image）

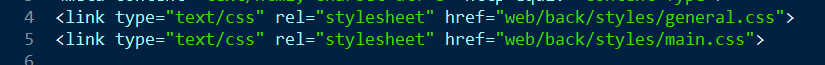


并依据平台进行管理：



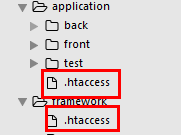
模板中载入css或者图片的路径需要更改：

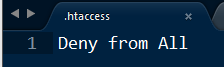
Login.html





Tip:防止浏览器直接请求到framework及其appliction





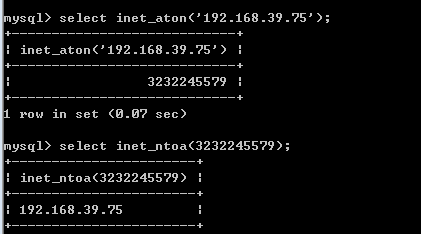
# 设计管理员表

67.255.255.255

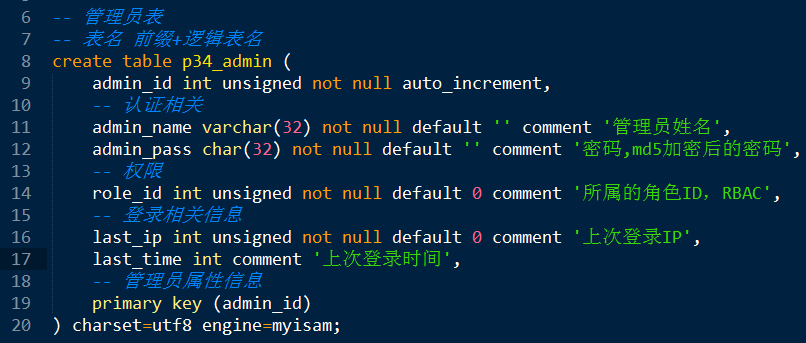
一个字节：8个位：

00000000——————11111111

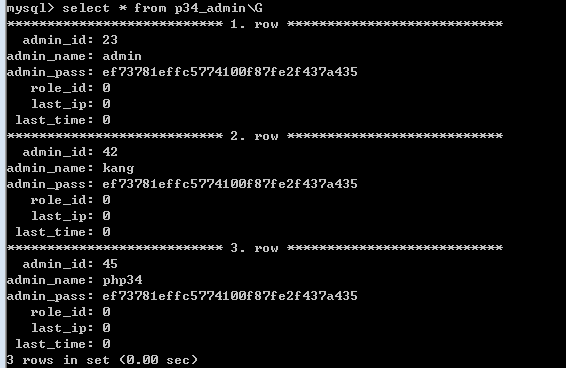
一个IPv4地址与一个无符号整型，可以进行相互转化：



表语句：



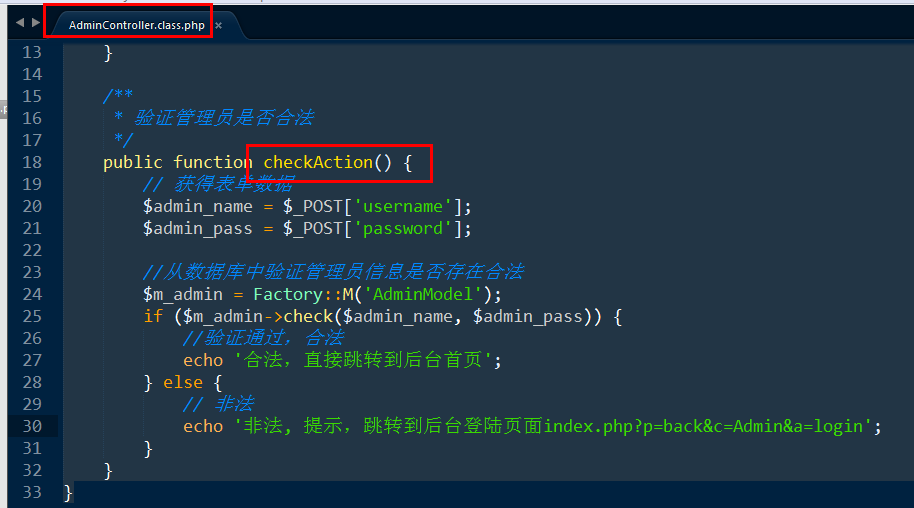
插入几条测试数据：



# 功能：管理员验证

## 控制器-动作

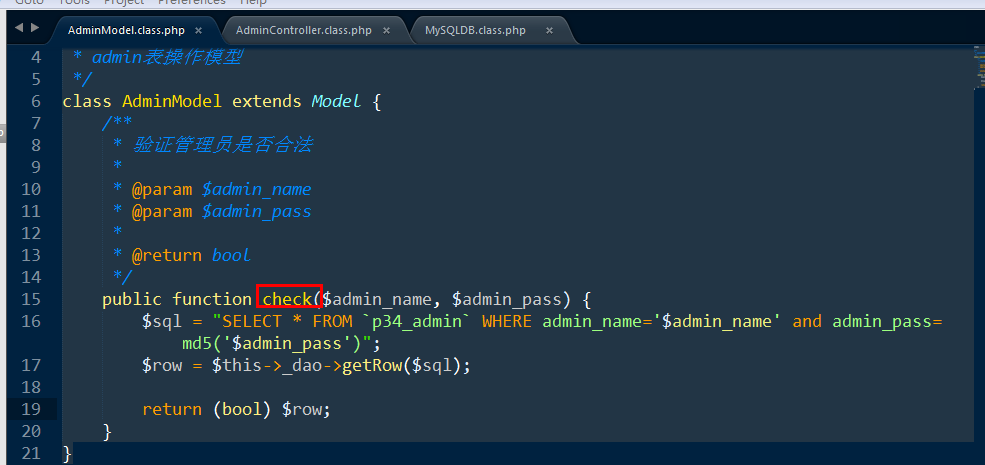
Back/AdminController->checkAction();



## 模型

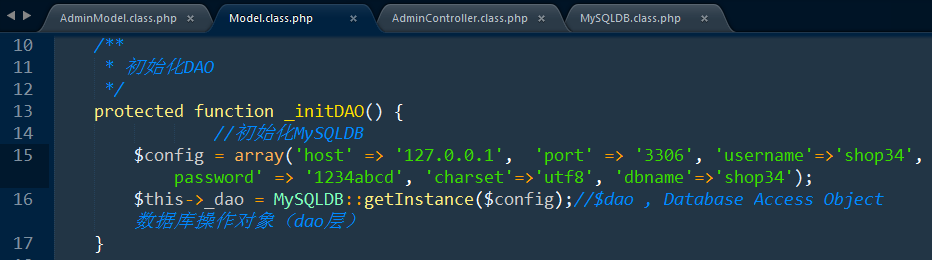
为操作admin表，建立模型AdminModel

Application/back/Model/AdminModel.class.php



测试：

别忘了修改数据服务器信息，为项目所用！（增加配置文件）



## 视图-模板

不需要