# 案例（项目）流程

## 需求

要做什么

我们：

带有登录功能的商品管理模块。

（参考成型的ecshop产品）

## 设计

怎么去做

架构

我们：

Lamp

## 编码实现

做

二次开发：

在一个成熟（常规业务逻辑实现）的产品上，完成自己项目的特定业务逻辑。

论坛（discuz，phpwind）

电商（ecshop，ecmall，shopex）

内容管理CMS（dede，wordpress）

社交（ThinkSNS，UCenter Home）

基于框架开发：

框架：规范了结构，基础功能 但 没有常规业务逻辑实现一种产品。

（ZendFramework，ThinkPHP，YII，symfony，CI 等等）

我们：

选择，基于框架开发。

基于 自定义的框架完成 业务逻辑。

## 测试

解决程序漏洞

## 发布

运营

## 维护升级

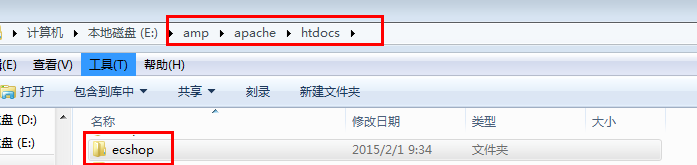
# 安装ecshop来参考

## 下载解压





## 上传到web服务器



## 通过web浏览器请求该项目

[可以设置虚拟主机]

请求后，就会自动进入安装页面，按照提示步骤安装即可!

必要环境：



如果mysql不支持，开启mysql扩展：php.ini



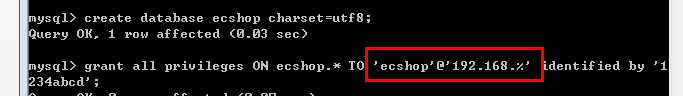
如果GD不支持，开启GD扩展：php.ini

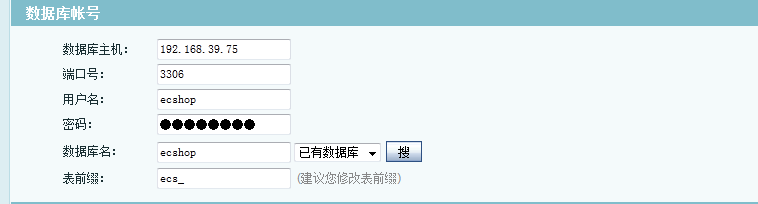


重启apache服务器！

配置数据库：

指定数据库，并设置好相应的用户权限。

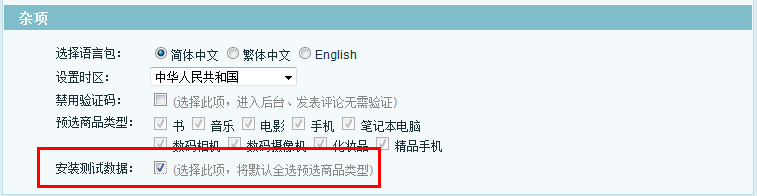




后台超级管理员帐号：



杂项，选中测试数据



测试访问：

前台：ecshop/index.php

后台：ecshop/admin/index.php

如何管理项目中大量的PHP代码(PHP文件)？

# 基本实现

功能，比赛列表，球队信息

数据表初始化：

比赛列表：

球队一 比分 球队二 比赛时间

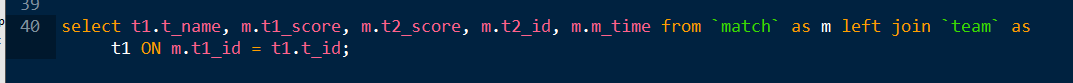
韩国 1：2 澳大利亚 15-01-31 17:00

从SQL思考，如何完成上面的查询：

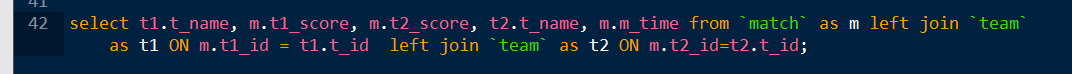
获得所有的比赛信息：

select \* from `match` ;

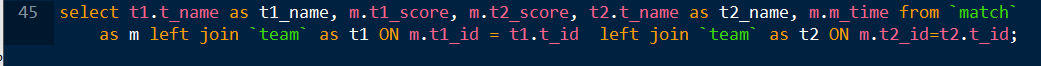
获得球队一名字：



获得球队二的名字：一个表在一次查询中可以出现多次，需要使用不同的别名进行标识！



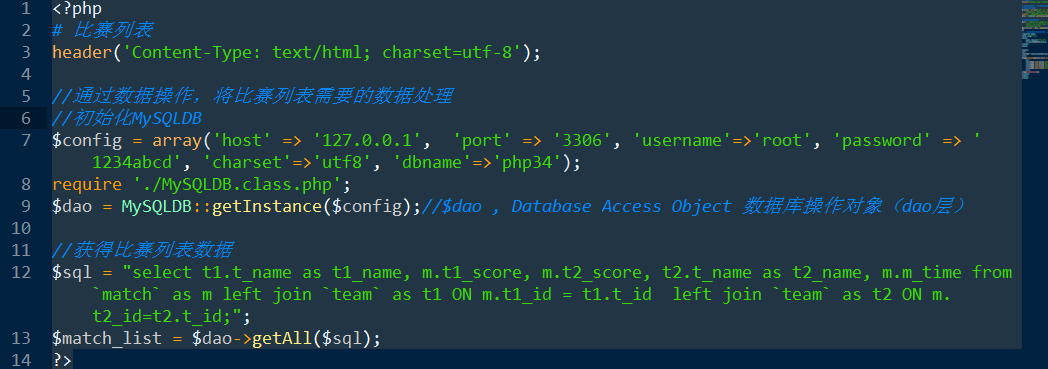
由于列名相同，需要起列别名：



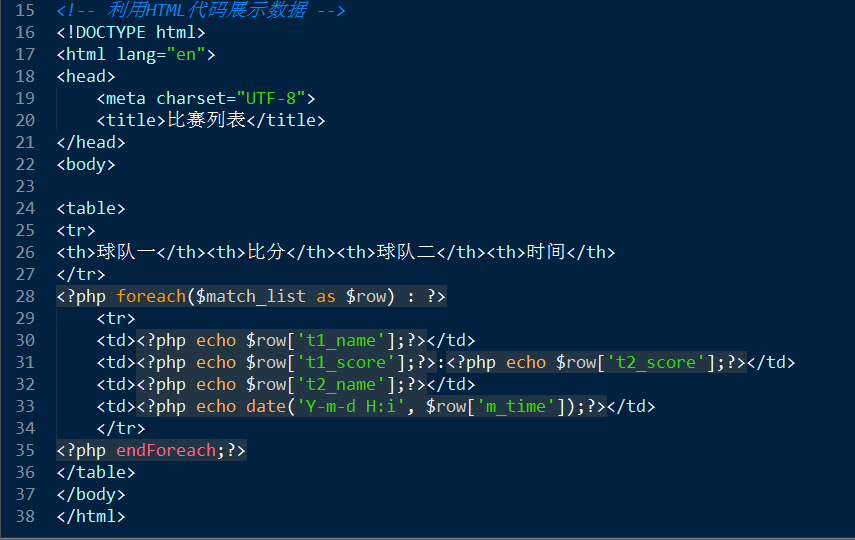


## PHP代码完成比赛列表

处理数据



展示：

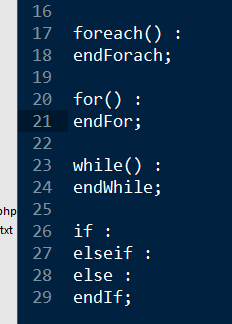


注意：

PHP逻辑最好与HTML展示分离出来

在HTML中嵌套PHP代码时，仅仅将需要PHP处理的内容，在PHP标签中输出，其他的都在PHP标签外输出。便于后期更新

模板中使用流程控制结构，建议使用 模板语法 来替代 大括号语法，包括代码段！

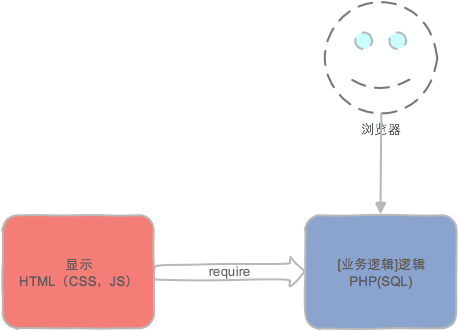


# 显示逻辑相分离

将功能强制分成两个部分，显示html文件，和逻辑PHP文件：

要求浏览器请求负责功能的PHP逻辑文件：

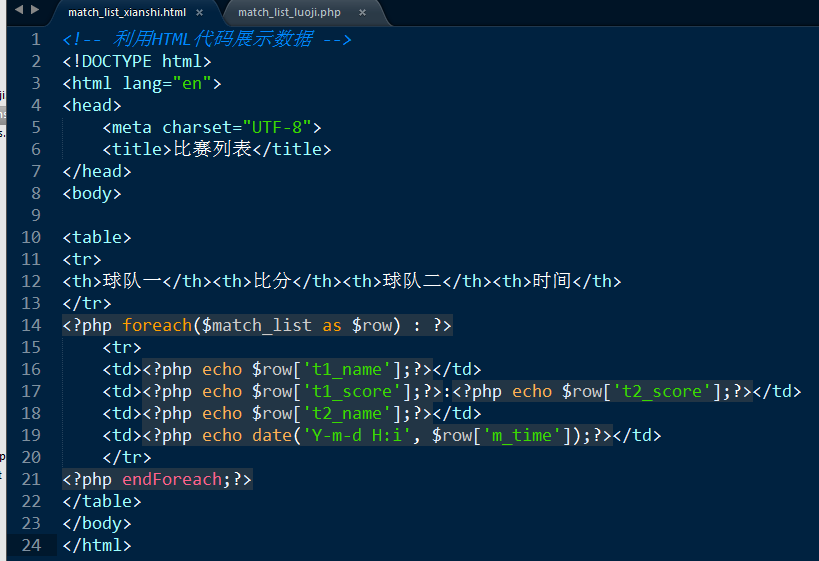
该PHP逻辑文件，对需要的显示内容进行载入。

s

逻辑PHP：



显示HTML：



浏览器请求：



好处：

为了更好的管理大量的代码：更新，重用 变得容易。

## 模板文件 – template

HTML，负责显示部分功能文件，称之为模板文件。

指的是：结构由HTML相关代码确定，数据部分由动态脚本（PHP）来填充，这样一种混编文件称之为模板。

“混编”：模板中PHP代码，仅仅负责输出数据，而一定不要负责处理数据。

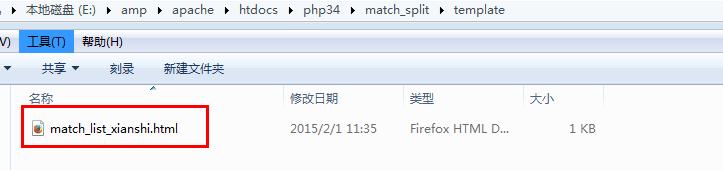
注意：浏览器应该，（不能）去请求负责展示的模板文件：

可见，需要将不能被浏览器所访问（模板）的文件隐藏起来，方案如下：

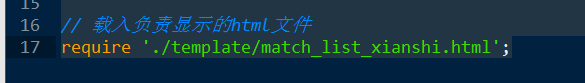
A计划：

通过web服务器对请求的控制，不允许浏览器直接访问

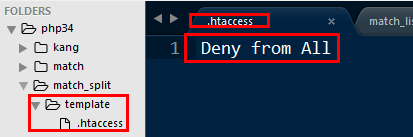
通常将不要被访问的统一管理到某个位置：



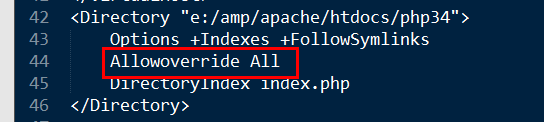
PHP代码载入正确的模板地址：



控制目录权限：通过apache的分布式配置文件完成



Tip:以上的配置，依赖于apache的主配置文件，允许目录内使用分布式配置文件.htaccess



B计划：

直接将不要被浏览器访问的文件资源，放在文档根目录之外！

因为浏览器可以请求的最上级目录，就是文档根目录了！

# MVC

以ecshop的前台为例：

功能一：

首页

购物车数据，商品分类数据，其他的首页需要的数据

功能二：

拍卖活动

购物车数据，商品分类数据，拍卖相关数据

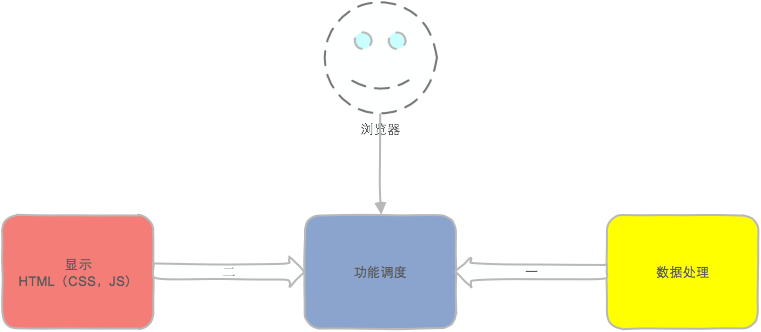
功能三：

团购商品

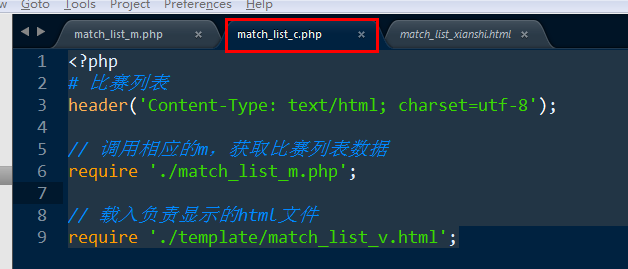
购物车数据，商品分类数据，团购相关数据

可见，应该数据的处理分离出来，在需要的功能中调用即可！

此时，采用三层分离（MVC）的方式来实现比赛列表：



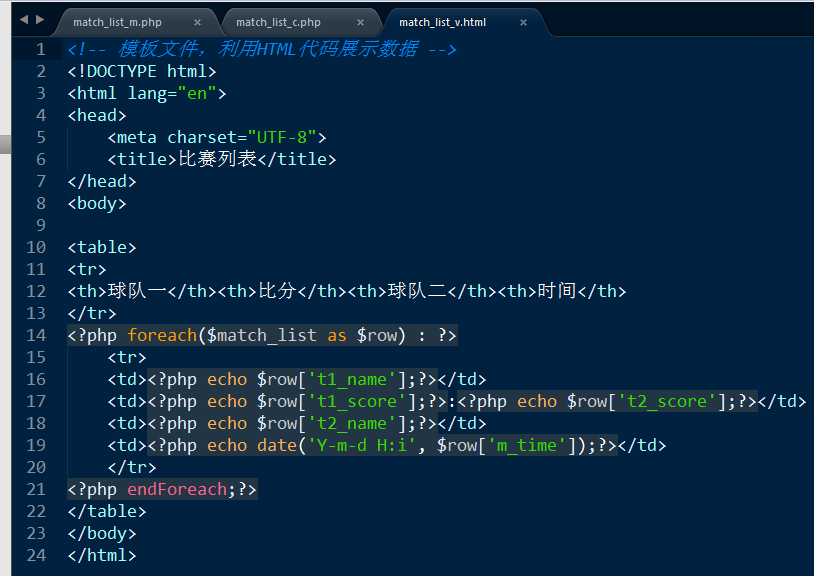
C：功能调度：



M，数据处理



V，结果展示



注意：浏览器需要请求C来获得需要的内容！

以上的划分方式，称之为MVC，项目架构思想！

其中：

M：Model，模型，项目中数据处理（业务逻辑处理）单元。

V：view，视图，项目中结果展示的单元（模板文件）。

C：Controller，控制器，项目中负责某个功能整体流程调度单元。

强调：MVC项目架构思想（项目中代码间的组织关系），不决定语法形式（OOP，面向过程），仅仅决定项目代码分工方式！

特点如下：

浏览器仅仅C控制器交互。

模型M和视图V都被控制器C所调用。

模型M和视图V之间，不存在任何的交互。

辨析：

MVC pk MVC框架？

MVC：架构思想

MVC框架：基于MVC思想架构的一个框架（基础代码的集合）

# 模型层典型实现

典型的MVC框架中，模型层代码组织结构是什么样的！

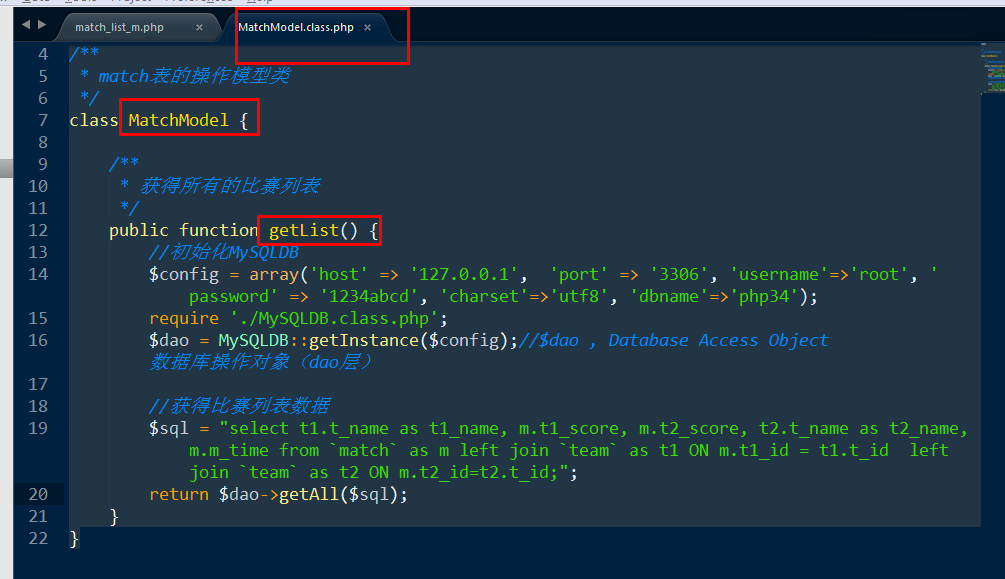
## 模型类

每张表，对应一个操作模型，当前表中的所有操作，都是用该模型完成！

[模型类]每张表的操作模型，由某个模型类实例化而来的对象【语法】。

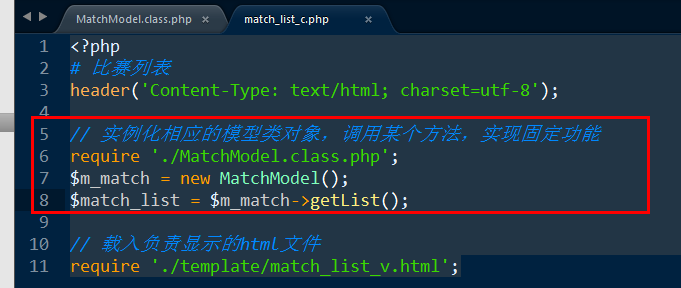
每个表操作，对应模型对象的一个方法。

模型类的示例代码：



同时修改控制器中，使用模型的方法：

C：



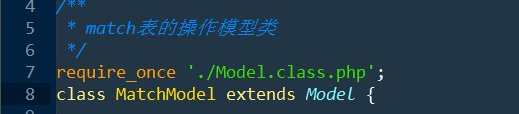
Tip：模型，在项目中，通常指的是模型类的对象，而不是模型类本身。

## 基础模型类

显而易见，在模型中，可能会出现重用的代码，例如（得到DAO对象过程），而且是每个模型对象的每个方法中都需要！



被其他的具体模型类所继承：

MatchModel 

基础模型类中增加，初始化数据库操作对象的功能：

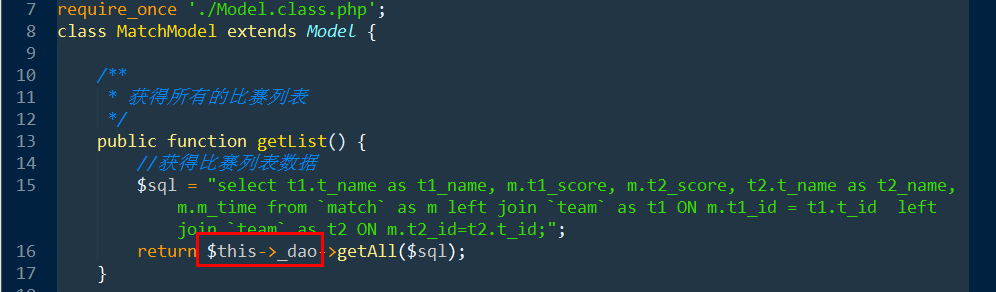
Model.class.php



将实例化好的MYSQLDB类的对象，存储到模型对象的属性上，从而保证，模型类的每个方法中都可以使用该属性。

模型方法中的使用：

MatchModel->getList();

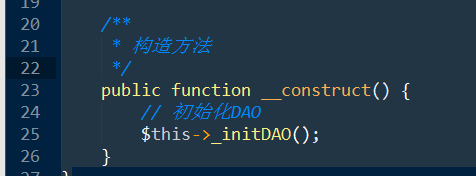


何时调用该初始化DAO的方法？

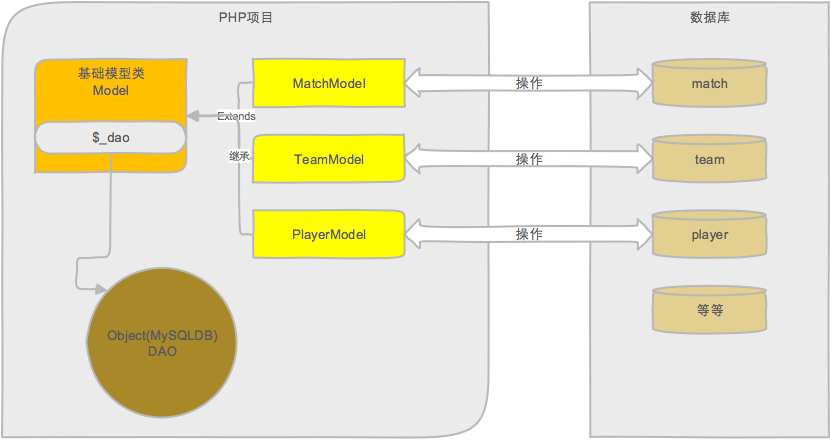
在实例化模型类对象时，就需要操作数据库，就需要执行初始化DAO的方法。

可见在构造方法中被调用即可：

Model.class.php



此时的模型结构关系：



## 模型的单例

如果在一个功能（控制器）中，如果使用某个表的多次操作，应该使用该表的一个模型就可以完成全部任务。

如何保证模型的单例？

典型的，可以通过一个单例工厂来实现（为什么不三私一公？是多个（所有的模型类）都需要单例效果）

工厂类：

直接new，不能实现需要的业务逻辑，需要辅助一段代码逻辑代码，才能确定如何去实例化对象，此时需要工厂类。

模型对象的单例效果：

不能在需要模型时直接就实例化，因为不能实现单例效果，需要一段逻辑代码，来判断当前模型类是否已经实例化过，如果实例化过，则直接返回实例化过的对象，否则实例化新的。

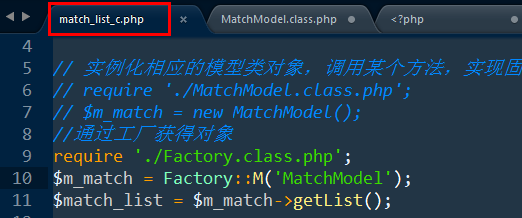
代码实现：

增加工厂类：

Factory.class.php



控制器中，为了得到单例对象，则需要通过工厂类的M方法完成：



## 模型的流程

