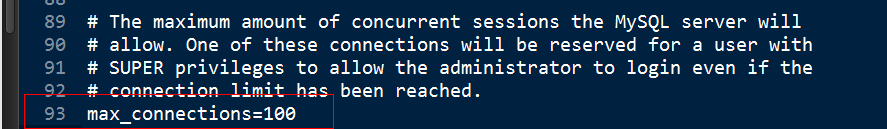
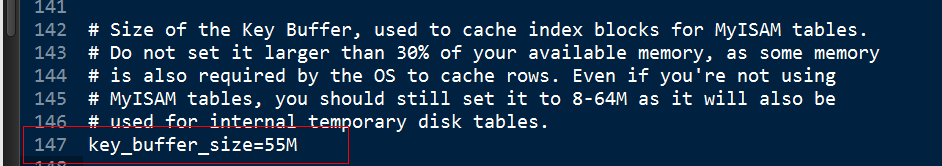
# 服务器配置

支持的最大连接数：



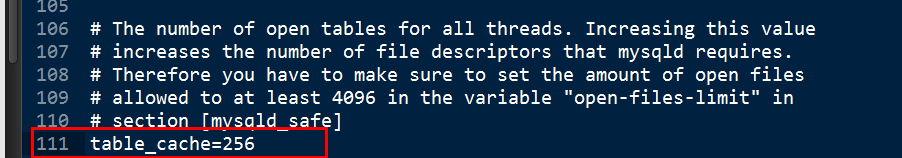
MyISAM

键缓存大小：



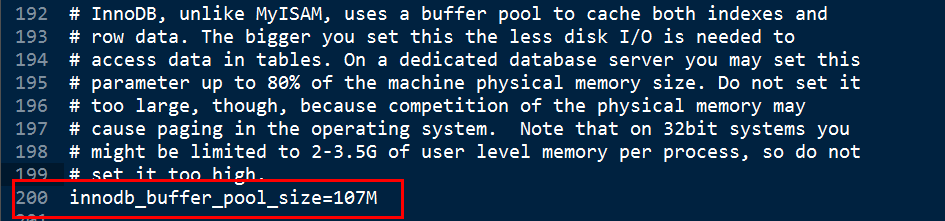
表缓存：

缓存的是 表文件的句柄



Innodb：

Innodb缓存，都是使用该缓存池例如，索引，事务日志缓存，等。



# 静态化

## 介绍

静态网站：

由HTML页面组成网站。

动态网站：

页面是数据，业务逻辑通过服务器端动态脚本生成。

效率上，浏览器请求静态页面，效率要高很多！

因此，才出现静态化的，将动态网站变化成静态的，便于高速访问。

原理，将原本动态形成的页面，存储为静态的HTML代码页，使浏览器直接请求该代码页。

类似 静态页面缓存。



测试，一普通的动态脚本，与一个静态HTML代码，相差多少：

动态：

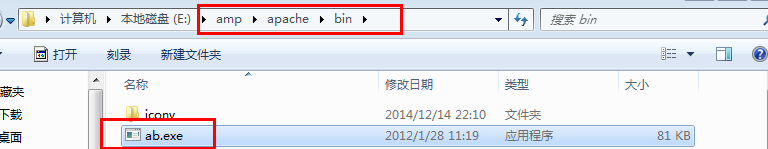


静态：

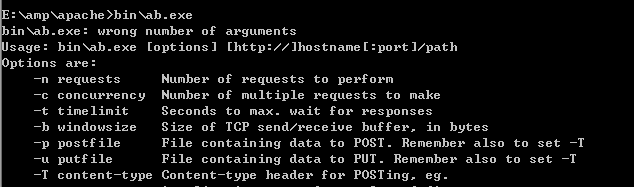


测试，还可以使用典型压力测试工具：

PHP开发常用的，就是Apache自带的 ab（apache benchmark,基准测试，压力测试。）程序。

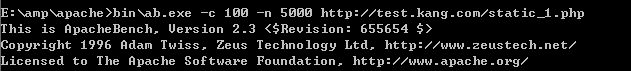


基于CMD的程序：



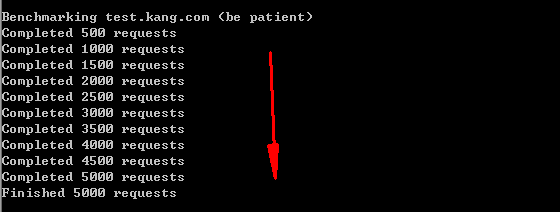
典型的测试，测试在某种并发的情况下完成某些请求所消耗的时间及其其他信息：

Ab.exe –c 100 –n 5000 URL

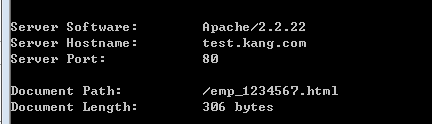


测试结果如下 ：

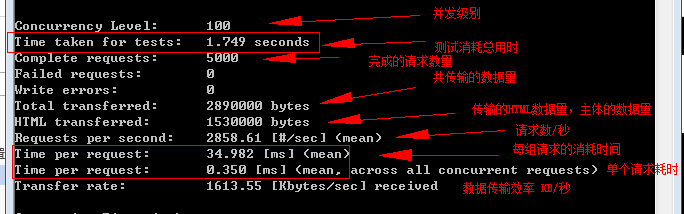
完成进度



响应概要信息：

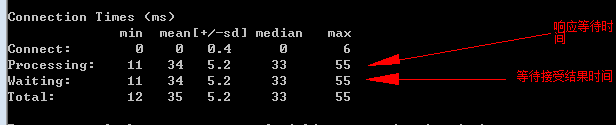


基准测试结果：

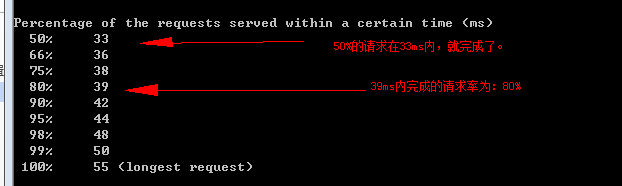


整体统计信息：

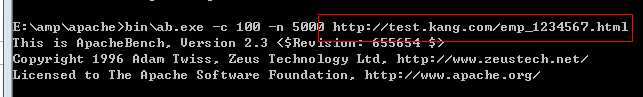
平均到每个请求的时间消耗

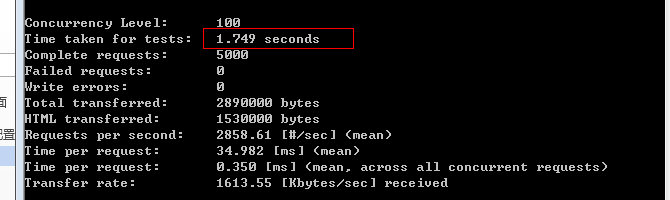


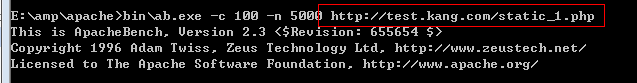
响应时间的分区统计：

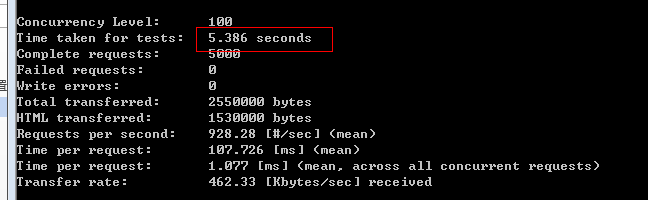


测试 静态的和动态的执行差异：









## 实现技术 – OB

需要的核心技术是OB，output\_buffering,输出缓存。

作用是：存储PHP所生成的响应主体内容。



操作缓冲，需要开启，处理被缓存的内容，关闭。

（输出缓存功能，PHP核心是支持的，不需要再开启扩展）

### 开启

Ob\_start();

### 获取内容

Ob\_get\_contents();

Ob\_get\_clean();

Ob\_get\_flush();

### 清空

Ob\_clean()

Ob\_get\_clean();

Ob\_end\_clean();

### 刷新

将缓冲区内容向下传输！

Ob\_flush()

Ob\_get\_flush();

Ob\_end\_flush();

Tip: 在脚本周期结束时，刷新操作会自动执行。

### 关闭

Ob\_end\_flush();

Ob\_end\_clean();

## 静态文件的制作

需要同时记录该数据所对应的静态文件地址。

通常，需要在 需要静态化的数据记录上，增加一个字段，存储该记录的静态地址：



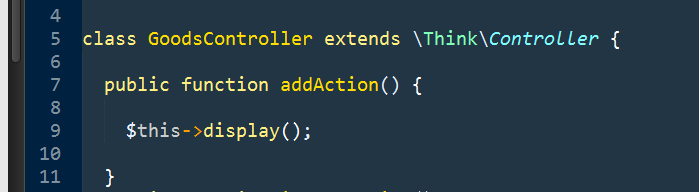
演示在项目生成静态化的：TP

实现 商品详细页，静态化效果。

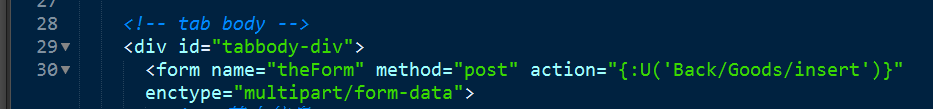
### 后台

添加商品时，生成该商品的静态页，存储其URL地址

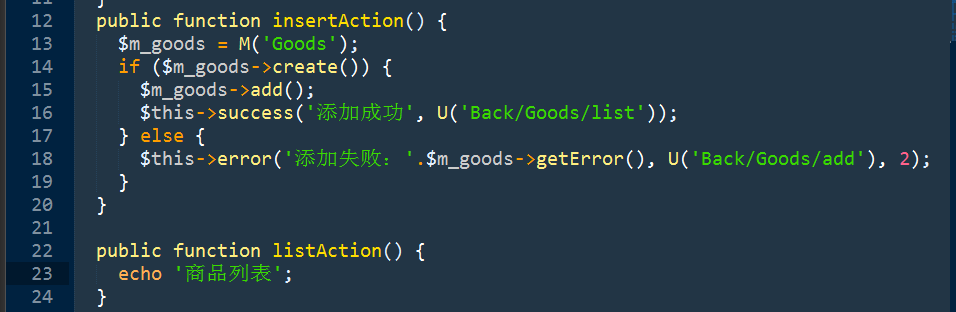
添加表单功能：



需要的模板：

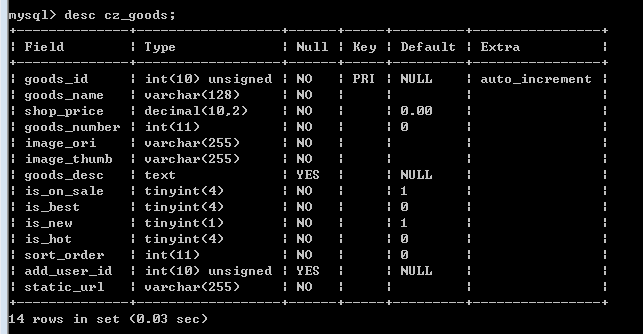


处理添加数据：



建表 cz\_goods:

 like 拷贝已有表的结构来创建新表。

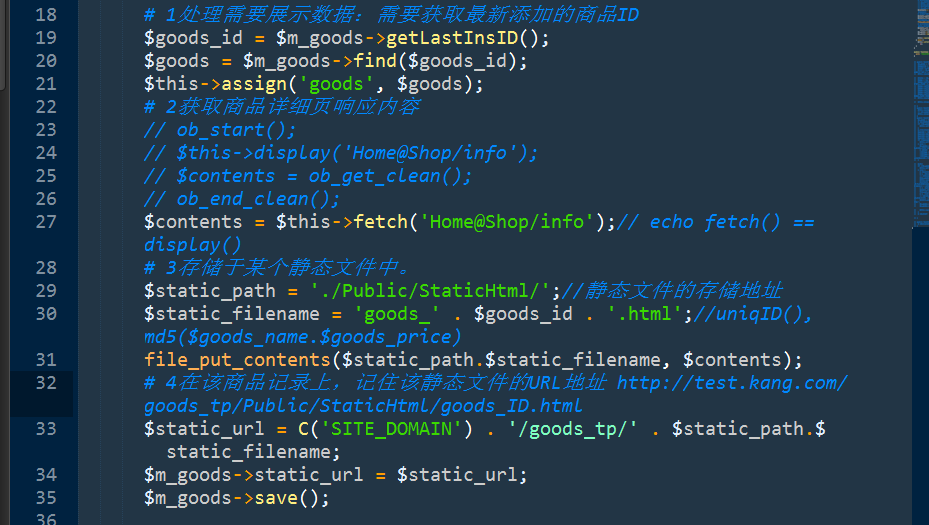


静态化相关操作

添加商品时，生成该商品的静态页，存储其URL地址

Static\_url 就是我们添加的存储静态URL的字段。

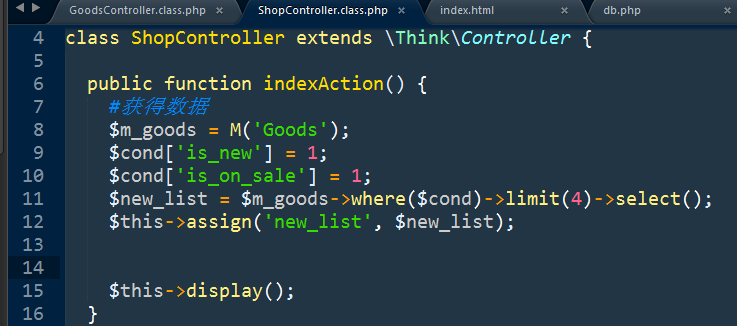
商品添加成功：



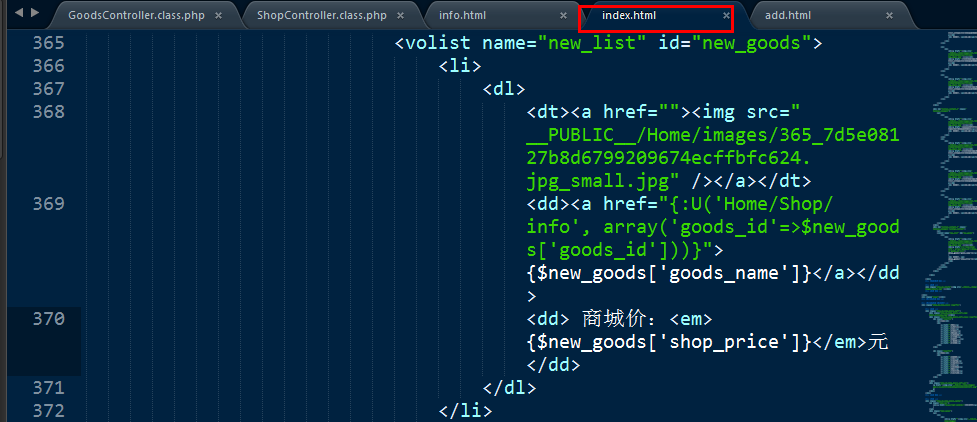
### 前台

当需要获取商品详细页时，展示静态页。

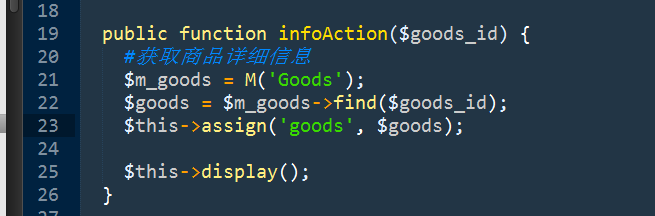
首页动作：



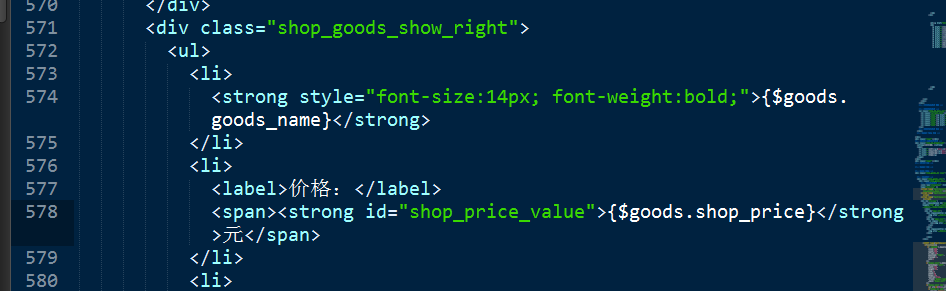
展示模板：



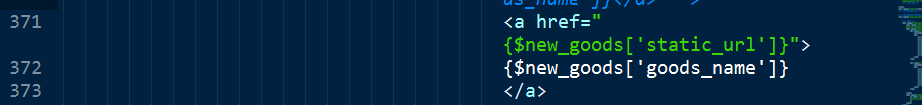
商品详情：



处理模板：



前台在请求时，使用这个静态连接即可：



### 扩展问题

1. 静态页面中动态（即时）数据问题。

Ajax。

1. 静态化内容，一旦改动，都需要重新生成静态页面。
2. 整站静态化，内容聚合页和内容详细页。其中详细页适合做静态化。
3. 管理大量静态文件。简单的分子目录存储。甚至分硬盘，分文件服务器存储。

# 伪静态 – URL优化

不是优化效率的方案。

指的是URL优化的一种方式

取决于是否存在 查询参数（queryString）

动态的URL： script.php?param=value&key=value

静态的URL（pathinfo）：script.php/param/value/key/value.html

静态化url的目的，是为了SEO（搜索引擎优化），还能为了简化URL，美化URL。

典型的URL优化，伪静态的方式如下：

1利用web服务器提供的重写URL功能（例如Apache mod\_rewrite）

2利用PHP自身对URL进行分析处理。（pathinfo，就是PHP处理的）（推荐）

## URL路由-PHP自身处理URL优化

以TP框架为例：

URL是TP（几乎所有的主流框架）提供的，一种自定义分析URL，并解析到指定指定的方法。

例如：

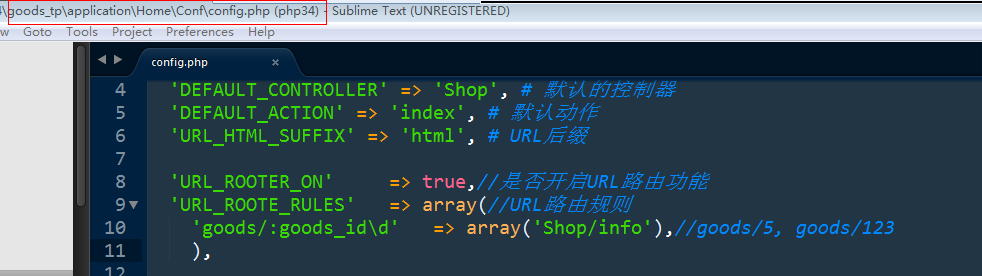
URL：<http://test.kang.com/goods_tp/index.php?m=Home&c=Shop&a=info&goods_id=5>

Pathinfo: <http://test.kang.com/goods_tp/index.php/Home/Shop/info/goods_id/5>

自定义： <http://test.kang.com/good_to/index.php/goods/5>

如果需要实现自定义的URL，来请求动作，则需要 URL路由功能：

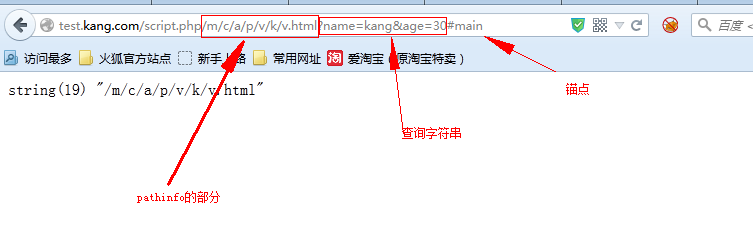
在 某个模块的配置文件中，配置如下的路由信息：



## PATHINFO-PHP自身处理

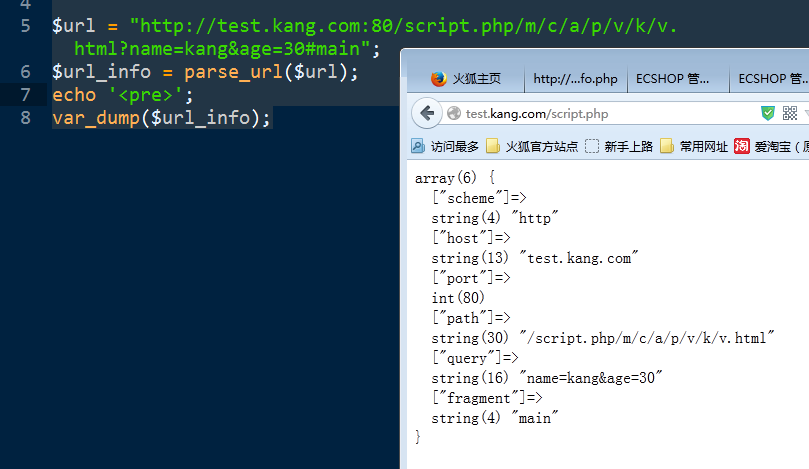
URL路由，和PATHINFO的实现：

获取用户请求的特定字符串，进行分析。



PHP提供了常见的URL格式字符串解析功能：

Parse\_url();

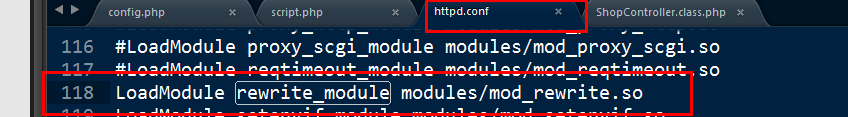


## 隐藏入口文件脚本

使用 web服务器（Apache httpd）完成：

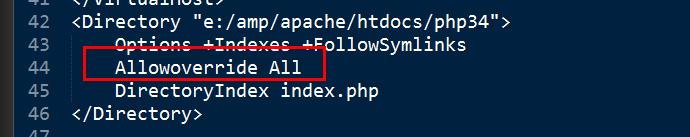
Apache的URL重写（mod\_rewrite）功能可以完成将一个URL，映射到另一个URL上。

### 开启Apache的rewrite模块

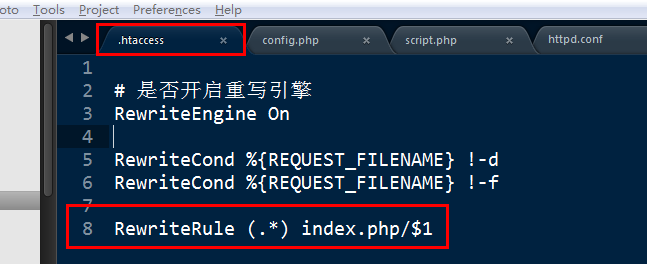


Apache httpd restart

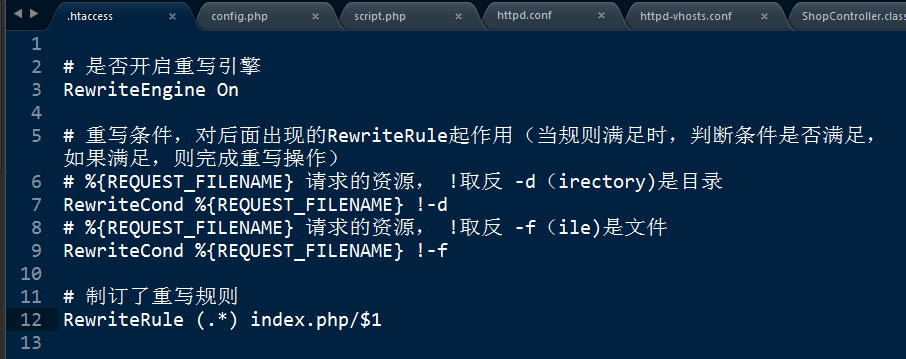
### [可选] 开启项目目录的.htaccess权限



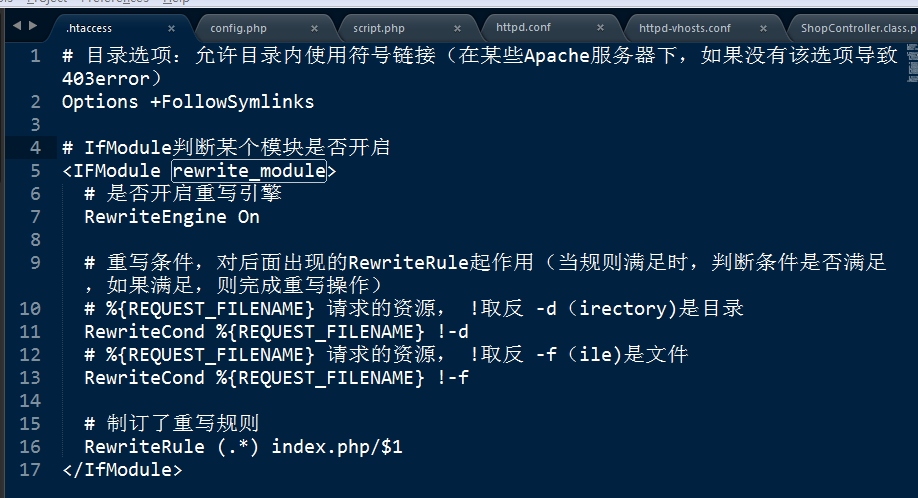
### .htaccess配置Apache重写规则



解释如下：



其他的相关指令：



重新不仅仅使用在URL隐藏index.php功能：

例如：防止倒链：

