淘淘商城第二天

# 第一天内容：

1. 电商行业的了解。
2. 淘淘商城项目介绍
3. 后台系统的搭建。
   1. 使用maven管理
   2. 聚合工程
   3. 创建maven模块，根据层级
4. svn使用
   1. 工程上传
   2. 下载工程（需要转换成maven工程）
5. Ssm框架的整合

# 课程计划

1. 商品列表查询工程
   1. Easyui
   2. 分页处理，分页插件
2. 商品的添加
   1. 商品类目选择-easyui异步tree控件的使用
   2. 图片上传（图片服务器开头，nginx）
   3. 富文本编辑器使用
   4. 添加的实现

# 展示首页

创建一个controller，做一个页面跳转。展示index.jsp的内容。

## 分析

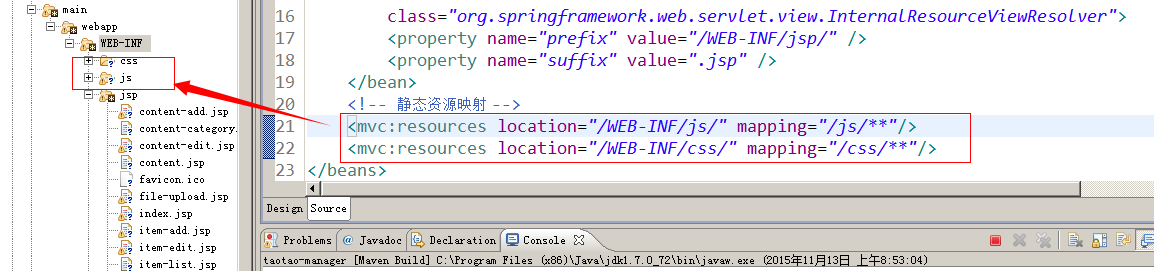
请求的url：/

返回的结果：返回一个string类型，是一个逻辑视图。

## 代码实现

|  |
| --- |
| @Controller  **public** **class** PageController {  @RequestMapping("/")  **public** String showIndex() {  **return** "index";  }  } |

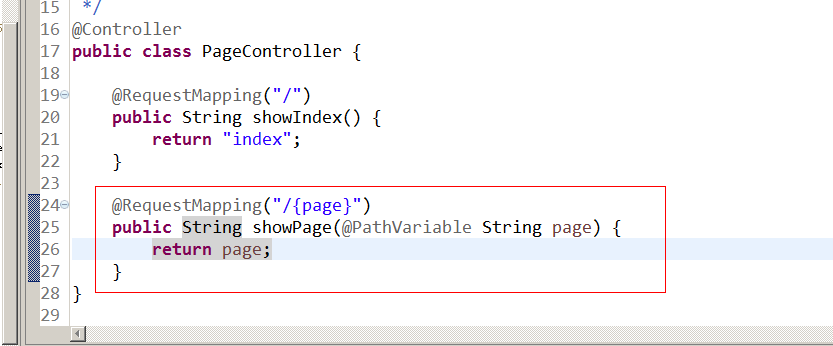
## 静态资源映射



# 商品列表展示

## 分析

打开商品列表页面，应该创建一个Handler接收请求，跳转到对应的页面。请求和页面的名称一致。可以统一处理。



查询的表：tb\_item

单表查询sql：SELECT \* from tb\_item **LIMIT 0,10**

需要实现分页，使用逆向工程，可以使用mybatis的分页插件。

## 分页插件

### 原理

SqlSessionFactory

拦截器，需要实现mybatis的拦截器接口。

对sql语句进行编辑。

把修改后sql语句设置回去。

数据库

MappedStatement

Sql语句的封装

Executor

执行器对象

SqlSession

提供很多用户方法。

### 使用方法

插件叫做PageHelper如果你也在用Mybatis，建议尝试该分页插件，这个一定是最方便使用的分页插件。

该插件目前支持Oracle,Mysql,MariaDB,SQLite,Hsqldb,PostgreSQL六种数据库分页。

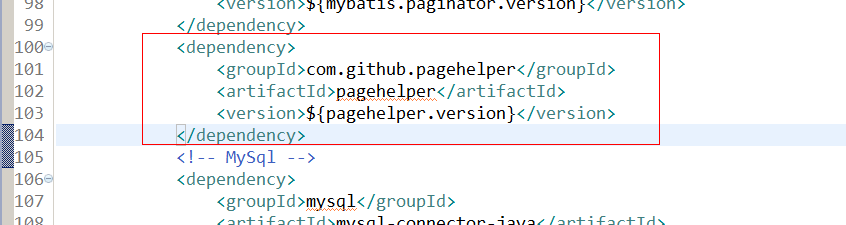
使用方法：

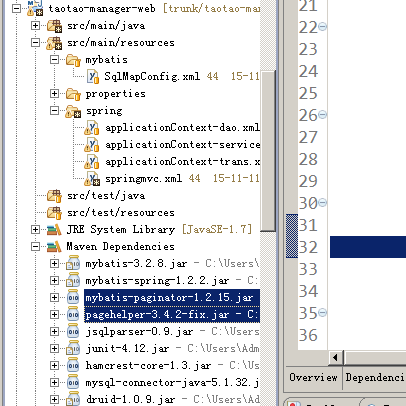
第一步：需要在SqlMapConfig.xml，配置一个plugin。

第二步：在sql语句执行之前，添加一个PageHelper.startPage(page,rows);

第三步：取分页结果。创建一个PageInfo对象需要参数，查询结果返回的list。从PageInfo对象中取分页结果。

#### 添加jar包到工程中





#### 修改SqlMapConfig.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd">  <configuration>  <!-- 配置分页插件 -->  <plugins>  <plugin interceptor=*"com.github.pagehelper.PageHelper"*>  <!-- 指定使用的数据库是什么 -->  <property name=*"dialect"* value=*"mysql"*/>  </plugin>  </plugins>  </configuration> |

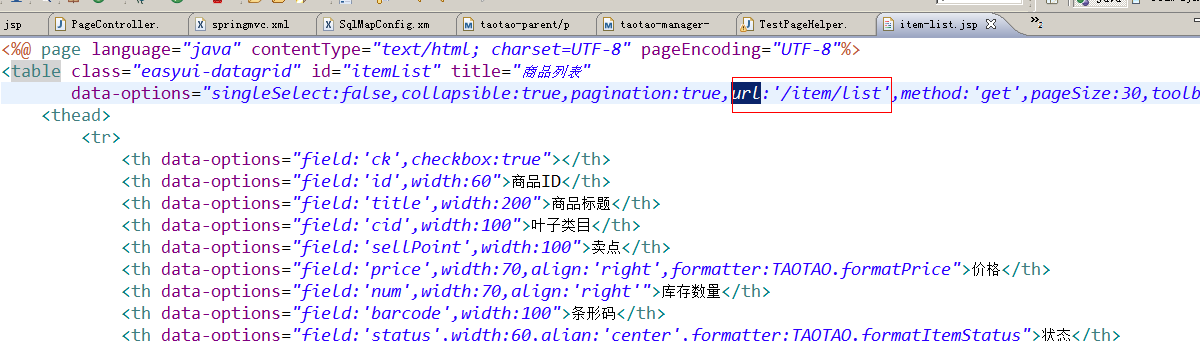
#### 代码测试

|  |
| --- |
| **public** **class** TestPageHelper {    @Test  **public** **void** testPageHelper() **throws** Exception {  //1、获得mapper代理对象  ApplicationContext applicationContext = **new** ClassPathXmlApplicationContext("classpath:spring/applicationContext-\*.xml");  TbItemMapper itemMapper = applicationContext.getBean(TbItemMapper.**class**);  //2、设置分页  PageHelper.*startPage*(1, 30);  //3、执行查询  TbItemExample example = **new** TbItemExample();  List<TbItem> list = itemMapper.selectByExample(example);  //4、取分页后结果  PageInfo<TbItem> pageInfo = **new** PageInfo<>(list);  **long** total = pageInfo.getTotal();  System.***out***.println("total:" + total);  **int** pages = pageInfo.getPages();  System.***out***.println("pages:" + pages);  **int** pageSize = pageInfo.getPageSize();  System.***out***.println("pageSize:" + pageSize);    }  } |

## 展示商品列表

### 页面分析

请求的url：



Ajax请求。

请求的参数：<http://localhost:8080/item/list?page=1&rows=30>

响应的数据：json数据。

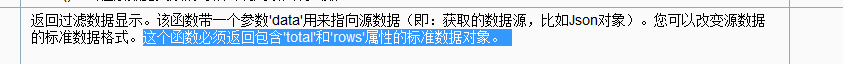
包含total、rows两个属性：

Total：查询结果的总记录数

Rows：集合，包含显示的所有数据。其中集合中每个元素的key应该和dategrid的field对应。

Easyui中datagrid控件要求的数据格式为：

{total:”2”,rows:[{“id”:”1”,”name”,”张三”},{“id”:”2”,”name”,”李四”}]}



### Dao层

可以使用逆向工程生成的代码

### Service层

参数：page（显示的页码）、rows（每页显示的记录数）

返回值：创建一个pojo表示返回值。应该包含total、rows两个属性。

应该放到taotao-common工程中，和其他系统共用。

|  |
| --- |
| **public** **class** EasyUIDataGridResult {  **private** **long** total;  **private** List<?> rows;  **public** **long** getTotal() {  **return** total;  }  **public** **void** setTotal(**long** total) {  **this**.total = total;  }  **public** List<?> getRows() {  **return** rows;  }  **public** **void** setRows(List<?> rows) {  **this**.rows = rows;  }    } |

|  |
| --- |
| @Override  **public** EasyUIDataGridResult getItemList(**int** page, **int** rows) {  //分页处理  PageHelper.*startPage*(page, rows);  //执行查询  TbItemExample example = **new** TbItemExample();  List<TbItem> list = itemMapper.selectByExample(example);  //取分页信息  PageInfo<TbItem> pageInfo = **new** PageInfo<>(list);  //返回处理结果  EasyUIDataGridResult result = **new** EasyUIDataGridResult();  result.setTotal(pageInfo.getTotal());  result.setRows(list);    **return** result;  } |

### Controller

接收两个参数：page、rows

调用Service查询商品列表。

返回：EasyUIDataGridResult（json数据），需要使用@ResponseBody

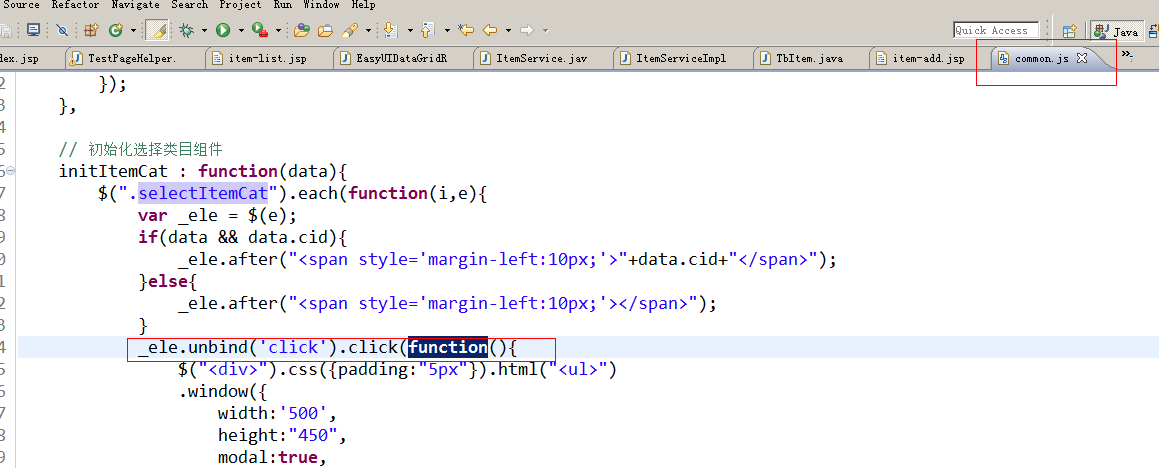
|  |
| --- |
| @RequestMapping("/item/list")  @ResponseBody  **public** EasyUIDataGridResult getItemList(Integer page, Integer rows) {  EasyUIDataGridResult result = itemService.getItemList(page, rows);  **return** result;  } |

# 添加商品-类目选择

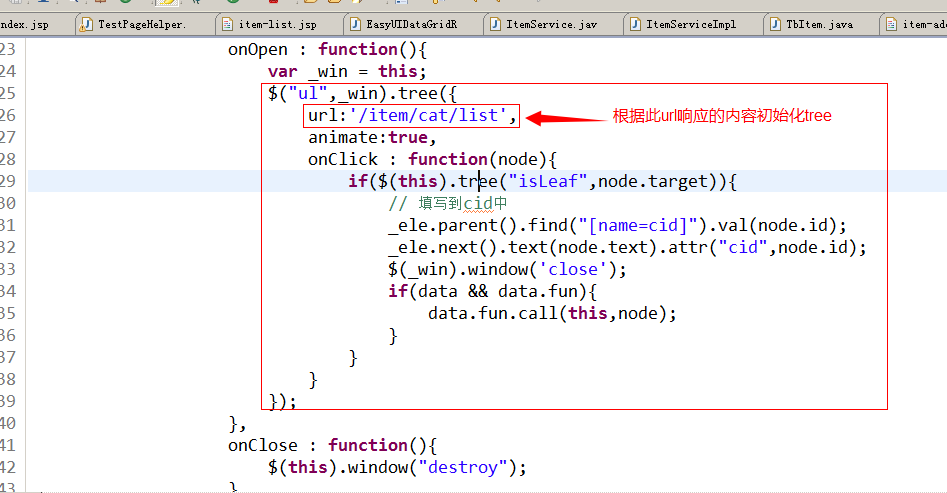
## 分析

商品类目使用的表：tb\_item\_cat

初始化类目选择：



Easyui的异步tree控件：



请求的url：/item/cat/list

请求的参数：id（当前节点的id）

响应的结果：json数据。

[{

"id": 1,

"text": "Node 1",

"state": "closed"

}

{

"id": 2,

"text": "Node 2",

"state": "closed"

}

]

如果当前节点为父节点，state应为“closed”、如果是叶子节点“open”

## Dao层

查询tb\_item\_cat表，根据id查询商品分类列表。可以使用逆向工程。

## Service层

### 分析

接收参数parentId，根据parentId查询分类列表。返回pojo列表。

Pojo应该包含三个属性：

id、text、state

应该放到taotao-common工程中。

|  |
| --- |
| **public** **class** EasyUITreeNode {  **private** **long** id;  **private** String text;  **private** String state;  **public** **long** getId() {  **return** id;  }  **public** **void** setId(**long** id) {  **this**.id = id;  }  **public** String getText() {  **return** text;  }  **public** **void** setText(String text) {  **this**.text = text;  }  **public** String getState() {  **return** state;  }  **public** **void** setState(String state) {  **this**.state = state;  }      } |

### 代码实现

|  |
| --- |
| @Service  **public** **class** ItemCatServiceImpl **implements** ItemCatService {  @Autowired  **private** TbItemCatMapper itemCatMapper;    @Override  **public** List<EasyUITreeNode> getItemCatList(**long** parentId) {  // 根据parentId查询分类列表  TbItemCatExample example = **new** TbItemCatExample();  //设置查询条件  Criteria criteria = example.createCriteria();  criteria.andParentIdEqualTo(parentId);  //执行查询  List<TbItemCat> list = itemCatMapper.selectByExample(example);  //转换成EasyUITreeNode列表  List<EasyUITreeNode> resultList = **new** ArrayList<>();  **for** (TbItemCat tbItemCat : list) {  //创建一个节点对象  EasyUITreeNode node = **new** EasyUITreeNode();  node.setId(tbItemCat.getId());  node.setText(tbItemCat.getName());  node.setState(tbItemCat.getIsParent()?"closed":"open");  //添加到列表中  resultList.add(node);  }  **return** resultList;  }  } |

## Controller

接收参数，parentId。调用Service查询分类类别，返回列表（json数据），需要使用@ResponseBody。

请求的url：/item/cat/list

|  |
| --- |
| @Controller  @RequestMapping("/item/cat")  **public** **class** ItemCatController {  @Autowired  **private** ItemCatService itemCatService;    @RequestMapping("/list")  @ResponseBody  **public** List<EasyUITreeNode> getItemCatList(@RequestParam(value="id", defaultValue="0")Long parentId) {  List<EasyUITreeNode> list = itemCatService.getItemCatList(parentId);  **return** list;  }    } |

# 图片上传

## 图片上传分析

### 传统方式

并发量小、用户少：

访问图片

http://ip:port/project/images/xxx.jpg

上传图片

上传到images目录

Webapp

|-index.html

|-WEB-INF

|-images

Tomcat

并发量高、用户多：

Tomcat1

请求图片

http://ip:port/project/images/a.jpg

上传图片

Webapp

|-index.html

|-WEB-INF

|-images

|-?

Tomcat2

Webapp

|-index.html

|-WEB-INF

|-images

|-a.jpg

### 互联网项目

解决方案，新建一个图片服务器

Tomcat1

上传图片

Webapp

|-index.html

|-WEB-INF

~~|-images~~

~~|-?~~

Tomcat2

Webapp

|-index.html

|-WEB-INF

~~|-images~~

~~|-a.jpg~~

上传图片’

请求图片

http://imageserver-ip:port/project/images/a.jpg

上传到图片服务器

图片服务器，独立ip

Images

|-a.jpg

专门保存图片，不管是哪个服务器接收到图片，都把图片上传到图片服务器。

图片服务器上需要安装一个http服务器，可以使用tomcat、apache、nginx

## Nginx

### 什么是nginx

是一个使用c语言开发的高性能的http服务器及反向代理服务器。

Nginx是一款高性能的http 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器。由俄罗斯的程序设计师Igor Sysoev所开发，官方测试nginx能够支支撑5万并发链接，并且cpu、内存等资源消耗却非常低，运行非常稳定。

### Nginx的应用场景

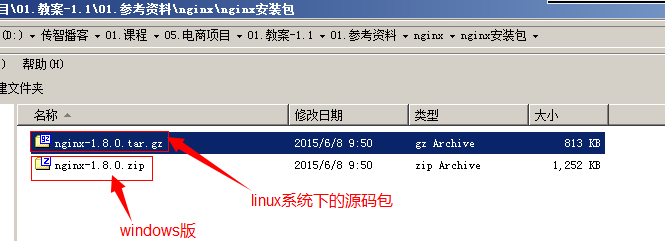
1. http服务器。Nginx是一个http服务可以独立提供http服务。可以做网页静态服务器。
2. 虚拟主机。可以实现在一台服务器虚拟出多个网站。例如个人网站使用的虚拟主机。
3. 反向代理，负载均衡。当网站的访问量达到一定程度后，单台服务器不能满足用户的请求时，需要用多台服务器集群可以使用nginx做反向代理。并且多台服务器可以平均分担负载，不会因为某台服务器负载高宕机而某台服务器闲置的情况。

### Nginx的安装

Nginx一般推荐安装到linux系统，而且要安装c语言的编译环境gcc。

#### 下载：

进入http://nginx.org/en/download.html 下载nginx1.8.0版本（当前最新稳定版本）。



#### 先安装nginx依赖的包：

nginx是C语言开发，建议在linux上运行，本教程使用Centos6.5作为安装环境。

* gcc

安装nginx需要先将官网下载的源码进行编译，编译依赖gcc环境，如果没有gcc环境，需要安装gcc：yum install gcc-c++

* PCRE

PCRE(Perl Compatible Regular Expressions)是一个Perl库，包括 perl 兼容的正则表达式库。nginx的http模块使用pcre来解析正则表达式，所以需要在linux上安装pcre库。

**yum install -y pcre pcre-devel**

注：pcre-devel是使用pcre开发的一个二次开发库。nginx也需要此库。

* zlib

zlib库提供了很多种压缩和解压缩的方式，nginx使用zlib对http包的内容进行gzip，所以需要在linux上安装zlib库。

**yum install -y zlib zlib-devel**

* openssl

OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及SSL协议，并提供丰富的应用程序供测试或其它目的使用。

nginx不仅支持http协议，还支持https（即在ssl协议上传输http），所以需要在linux安装openssl库。

**yum install -y openssl openssl-devel**

#### 安装步骤

第一步：把nginx的源码上传到linux系统

第二步：把压缩包解压缩。

第三步：进行configure。

./configure \

--prefix=/usr/local/nginx \

--pid-path=/var/run/nginx/nginx.pid \

--lock-path=/var/lock/nginx.lock \

--error-log-path=/var/log/nginx/error.log \

--http-log-path=/var/log/nginx/access.log \

--with-http\_gzip\_static\_module \

--http-client-body-temp-path=/var/temp/nginx/client \

--http-proxy-temp-path=/var/temp/nginx/proxy \

--http-fastcgi-temp-path=/var/temp/nginx/fastcgi \

--http-uwsgi-temp-path=/var/temp/nginx/uwsgi \

--http-scgi-temp-path=/var/temp/nginx/scgi

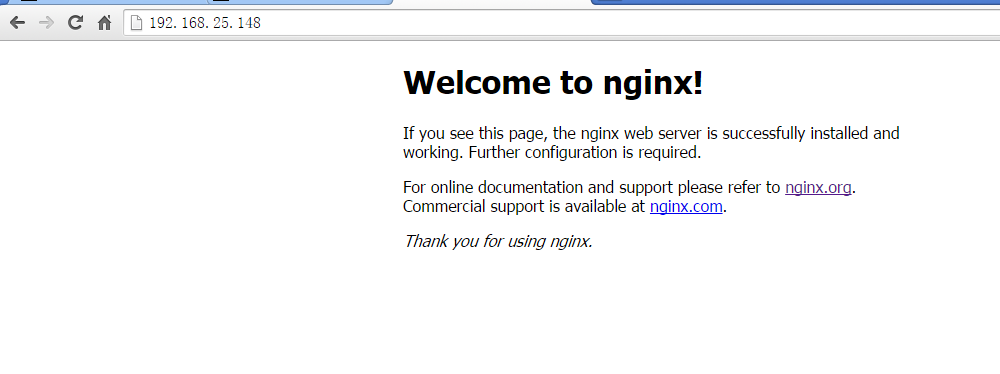
**注意：上边将临时文件目录指定为/var/temp/nginx，需要在/var下创建temp及nginx目录**

第四步：make

第五步：make install

### Nginx的启动、停止

1、启动：进入nginx的sbin目录，./nginx就可以启动。



如果访问不到，首先查看防火墙是否关闭。

2、关闭nginx：

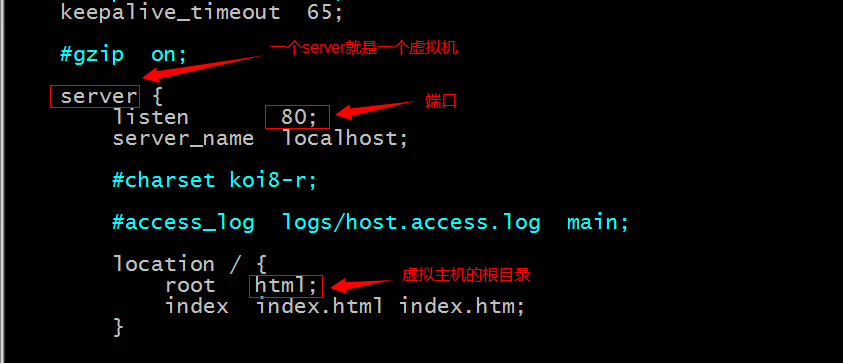
可以使用kill命令，但是不推荐使用。

推荐使用：./nginx -s stop

1. 刷新配置：./nginx -s reload

### Nginx的配置

在/usr/local/nginx/conf目录下nginx.conf文件是nginx的配置文件。



### 使用nginx配置虚拟机

#### 通过端口区分虚拟机

在nginx.conf文件中添加一个Service节点，修改端口号就可以

|  |
| --- |
| server {  listen 81;  server\_name localhost;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html81;  index index.html index.htm;  }  } |

#### 通过域名区分虚拟机

##### 域名介绍

DNS服务器

通过ip地址访问服务器

返回ip地址

Pc机

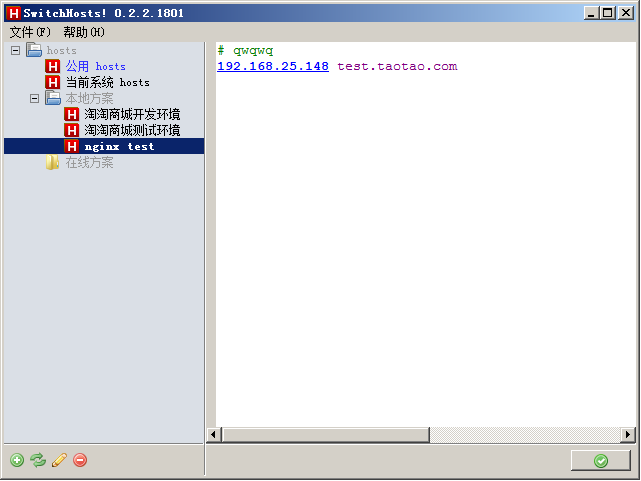
Web服务器

根据域名换ip地址

可以通过修改host文件指定域名的ip地址。

Host文件的位置：C:\Windows\System32\drivers\etc

可以使用工具：



##### 配置基于域名的虚拟主机

需要修改nginx.conf配置文件。

|  |
| --- |
| server {  listen 80;  server\_name test3.taotao.com;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html-test3;  index index.html index.htm;  }  } |

修改配置后需要重新加载配置文件。