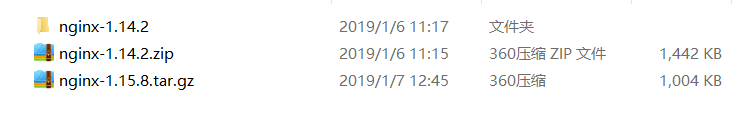
**Nginx手册**

# 1 nginx安装



## windows安装

1.14.2.zip是windows版本，直接解压即可。

## linux安装

1.15.8.tar.gz是linux版本，直接解压即可，

（注意：不要将nginx解压到带有中文目录的目录中，否则启动不起来）

### 安装Nginx

获取Nginx安装包

wget http://nginx.org/download/nginx-1.15.8.tar.gz

解压nginx：tar -zxvf nginx-1.15.8.tar.gz

进入nginx目录：cd nginx-1.15.8安装依赖的库

yum -y install libpcre3 libpcre3-dev openssl libssl-dev libperl-dev

编译安装：

make

make install

查看安装路径：whereis nginx



启动、停止：

cd /usr/local/nginx/sbin/

./nginx #启动

./nginx -s stop #此方式相当于先查出nginx进程id再使用kill命令强制杀掉进程

./nginx -s quit #此方式停止步骤是待nginx进程处理任务完毕进行停止

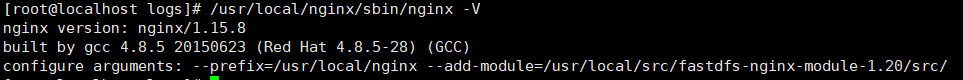
./nginx -s reload

验证启动状态：wget <http://本机IP地址>

|  |
| --- |
| 如果没有反应：  修改nginx.conf 配置文件： server {  listen 80;  server\_name 192.168.153.136;( 为当前机器ip ) |

查看此时的nginx版本：发现fastdfs模块已经安装好了

/usr/local/nginx/sbin/nginx -V

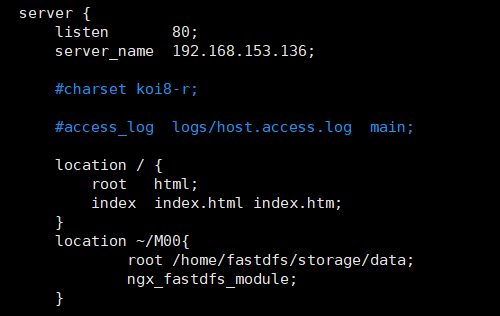


### 配置Nginx

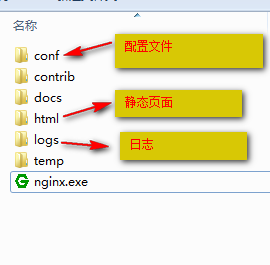
配置nginx，80端口server增加location如图：

cd /usr/local/nginx/conf/

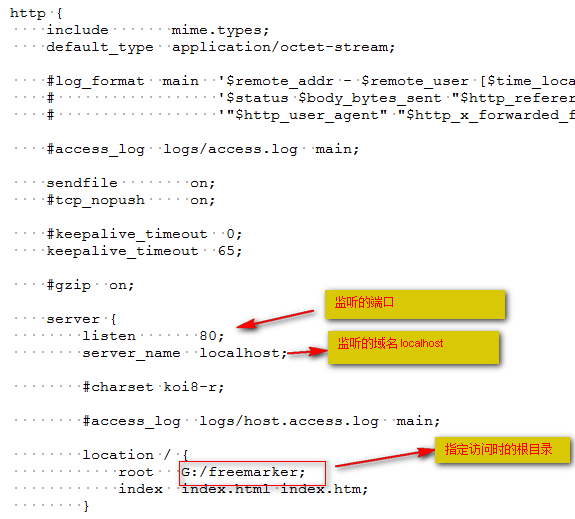
vi nginx.conf



# 2 目录结构



# 3 nginx.conf配置文件



# 4 多站点配置nginx.conf

#user nobody;

worker\_processes 1;

#error\_log logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#pid logs/nginx.pid;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

#log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

# '$status $body\_bytes\_sent "$http\_referer" '

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

#access\_log logs/access.log main;

sendfile on;

#tcp\_nopush on;

#keepalive\_timeout 0;

keepalive\_timeout 65;

#gzip on;

#-----站点一

upstream manager {

server 127.0.0.1:9000;

}

server {

listen 80; #监听端口

server\_name taotao.manager.com; #域名访问

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

proxy\_pass http://manager; #站点ip

root html; #根目录

index index.html index.htm; #根目下的页面

}

}

#-----站点二

upstream item {

server 127.0.0.1:9400;

}

server {

listen 80; #监听端口

server\_name item.taotao.com; #域名访问

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

proxy\_pass http://item; #站点ip

root html; #根目录

index index.html index.htm; #根目下的页面

}

}

#-----站点三

upstream search {

server 127.0.0.1:9300;

}

server {

listen 80; #监听端口

server\_name search.taotao.com; #域名访问

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

proxy\_pass http://search; #站点ip

root html; #根目录

index index.html index.htm; #根目下的页面

}

}

#-----站点四

upstream protal {

server 127.0.0.1:9100;

}

server {

listen 80; #监听端口

server\_name protal.taotao.com; #域名访问

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

proxy\_pass http://protal; #站点ip

root html; #根目录

index index.html index.htm; #根目下的页面

}

}

#-----站点五（静态网页站点）

server {

listen 80; # 监听端口

server\_name localhost; # 站点域名

location / {

root G:/freemarker; # 站点根目录

index index.html index.htm index.php; # 默认导航页

}

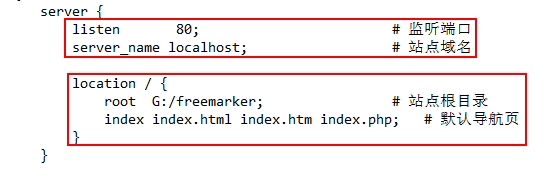
}

}

也可直接在proxy\_pass处写ip，配置：



站点二配置：直接在http{ … }中在添加一个server{ … } 修改相应的内容



或者使用这种方式：

#配置一个ip地址

upstream tomcat1 {

server 127.0.0.1:8082;

}

server {

listen 80;

server\_name www.taotao.com;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

proxy\_pass <http://tomcat1>; #直接引用配置的ip名称

index index.html index.htm;

}

# 5 Nginx负载均衡配置

如果一个服务由多台服务器提供，需要把负载分配到不同的服务器处理，需要负载均衡。

只需要：在upstream中添加多个server ip; 即可

upstream tomcat2 {

server 192.168.25.148:8081;

server 192.168.25.148:8082;

}

默认的负载均衡的策略就是**轮询的方式**。

可以根据服务器的实际情况调整服务器权重。权重越高分配的请求越多，权重越低，请求越少。默认是都是1

|  |
| --- |
| upstream tomcat2 {  server 192.168.25.148:8081;  server 192.168.25.148:8082 weight=2;  } |

其他的负载均衡的策略：

1.通过IP地址的hash值 做映射。

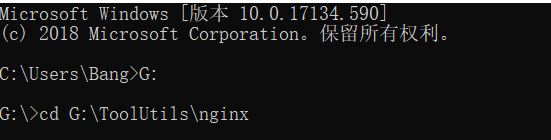
2.通过URL的方式计算出Hash值 。

3.随机策略。

4.最少并发量。

# 6 操作命令

先进入到安装目录中，在执行cmd命令



启动命令：start nginx.exe





关闭命令： nginx.exe -s stop



刷新配置文件：nginx. exe -s reload



# 7 更多：

有时候你想在一台服务器上为不同的域名运行不同的站点。比如www.siteA.com作为博客，www.siteB.com作为论坛。你可以把两个域名的IP都解析到你的服务器上，但是没法在Nginx的根目录里同时运行两个不同的网站。这时候，你就需要使用虚拟目录了。假设你把博客放在”/home/user/www/blog”下，论坛放在”/home/user/www/forum”下。下面我们就开始配置了：

* 在Nginx配置目录下，创建一个”vhost”目录。本例假设Nginx是默认安装，配置目录在”/etc/nginx”

$ sudo mkdir /etc/nginx/vhost

* 创建siteA的配置文件

$ sudo vi /etc/nginx/vhost/vhost\_siteA.conf

* 输入以下配置信息

[复制代码](javascript:void(0);)

server {

listen 80; # 监听端口

server\_name www.siteA.com siteA.com; # 站点域名

root /home/user/www/blog; # 站点根目录

index index.html index.htm index.php; # 默认导航页

location / {

# WordPress固定链接URL重写

if (!-e $request\_filename) {

rewrite (.\*) /index.php;

}

}

# PHP配置

location ~ \.php$ {

fastcgi\_pass unix:/var/run/php5-fpm.sock;

fastcgi\_index index.php;

include fastcgi\_params;

}

}

[复制代码](javascript:void(0);)

* 同siteA一样创建siteB的配置文件，两者仅有的不同是”server\_name”和”root”目录

$ sudo vi /etc/nginx/vhost/vhost\_siteB.conf

[复制代码](javascript:void(0);)

server {

...

server\_name www.siteB.com siteB.com; # 站点域名

root /home/user/www/forum; # 站点根目录

...

}

[复制代码](javascript:void(0);)

* 打开nginx.conf文件

sudo vi /etc/nginx/nginx.conf

* 将虚拟目录的配置文件加入到”http {}”部分的末尾

http {

...

include /etc/nginx/vhost/\*.conf;

}

* 重启Nginx服务

$ sudo service nginx restart

* 现在访问www.siteA.com和www.siteB.com，你将发现浏览器会打开不同的站点

## 禁止访问小技巧

假如你的Nginx根目录设在”/home/user/www”，你想阻止别人通过”http://IP地址/blog”或”http://IP地址/forum”来访问你的站点，最简单的方法就是禁止IP地址访问。方法如下：

1. 打开Nginx网站默认配置文件，记得先备份

$ sudo cp /etc/nginx/sites-available/default /etc/nginx/sites-available/default\_bak

$ sudo vi /etc/nginx/sites-available/default

1. 将所有内容删除，只留以下配置

server {

listen 80 default\_server;

server\_name \_;

return 404;

}

1. 重启Nginx后，别人将无法通过IP地址访问网站了

**如果你不想禁止IP地址访问整个目录，只是要防止别人通过IP访问你的博客和论坛。那就需要禁止”/blog”和”/forum”的目录访问。**

1. 打开Nginx网站默认配置文件，同上面一样，记得先备份
2. 在”server { }”部分加上以下配置

[复制代码](javascript:void(0);)

location ^~ /blog/ {

deny all;

}

location ^~ /forum/ {

deny all;

}

[复制代码](javascript:void(0);)

1. 重启Nginx即可

Nginx详情介绍

# 域名访问系统

## 目前访问系统存在的问题

目前访问后台系统：

http://localhost:8081/

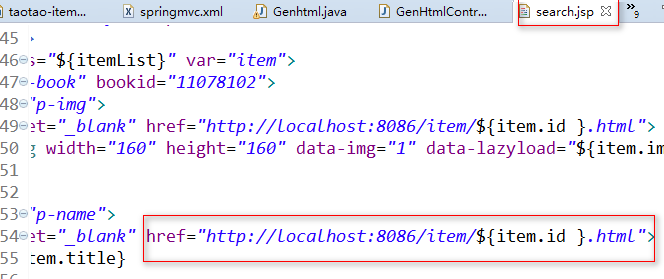
目前访问首页系统：

http://localhost:8082/

目前访问搜索结果页面：

http://localhost:8085

在搜索结果页面search.jsp中，代码中写死了其他的系统的全路径。



问题：

1.localhost只能访问本地，不能访问其他的服务器；

系统应当要部署在测试环境和生产环境。

可以使用ip地址？

2.开发环境的ip地址和测试环境的ip地址是不一样的。每次环境变化的时候，都需要修改访问的ip地址

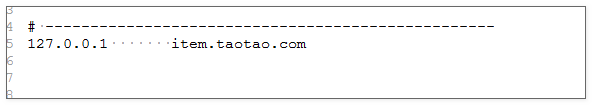
3.页面加载资源或者请求其他系统的URL时，使用了全路径，一旦环境发生改变，资源无法加载，请求无法访问。

4.ip地址没有意义，不容易记忆，用户不会通过ip地址进行访问，一般通过域名访问。

## 使用域名访问

例如：访问本地开发环境下的商品详情页面：

修改本机hosts文件：



访问地址：http://item.taotao.com:8086/item/536563.html,

存在的问题：

访问时域名后还需要加上端口。

可以使用nginx进行配置，达到只使用域名访问的目的。

# 什么是nginx

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用c语言所开发，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。开源。

# 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的使用和安装

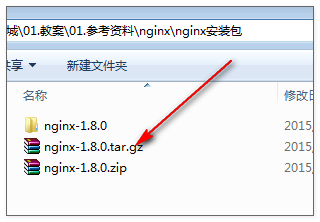
## nginx的安装

下载nginx：

官方网站：

<http://nginx.org/>

使用的版本是1.8.0版本。



Nginx提供的是源码。

## 安装要求的环境

1. 需要安装gcc的环境。**yum -y install gcc-c++**
2. 第三方的开发包。

* 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的源码包上传到linux系统

第二步：解压缩

[root@localhost ~]# tar -zxf nginx-1.8.0.tar.gz

（高版本，可直接跳过三 四步,直接make make install ）

第三步：

**目录指定为/var/temp/nginx，需要在/var下创建temp及nginx目录:**

**mkdir /var/temp/nginx/client -p**

第四步：使用configure命令创建一makeFile文件。

./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 \

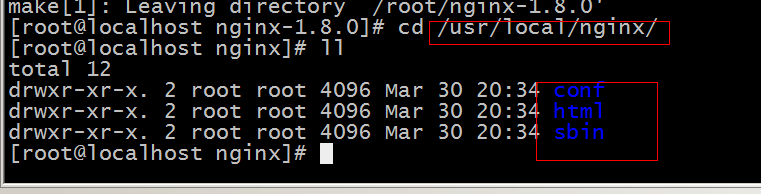
--http-uwsgi-temp-path=/var/temp/nginx/uwsgi \

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

[root@localhost sbin]# mkdir /var/temp/nginx/client -p

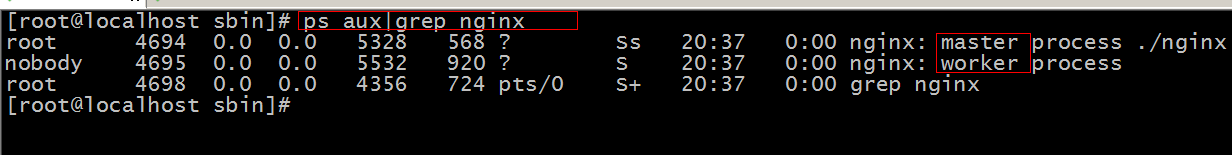
第四步：编译,使用：make

第五步：make install



## 启动nginx

进入sbin目录

[root@localhost sbin]# ./nginx   


## 测试访问nginx

浏览器输入：192.168.25.148



访问的默认端口是80端口。注意：看是否关闭防火墙。

## nginx的3个命令

* 启动命令

./nginx

* 关闭命令

./nginx -s stop 和

./nginx -s quit

* 刷新配置文件

./nginx -s reload

修改了nginx.conf文件之后，可以不重启Nginx，能马上生效

# nginx配置虚拟主机

虚拟的主机就是在一台服务器启动多个网站。

如何区分不同的网站：

1. 端口不同
2. 域名不同

## 通过端口区分不同的虚拟的主机

Nginx的配置文件配置项解释如下：

/usr/local/nginx/conf/nginx.conf

|  |
| --- |
| #user nobody;  worker\_processes 1;  #error\_log logs/error.log;  #error\_log logs/error.log notice;  #error\_log logs/error.log info;  #pid logs/nginx.pid;  events {  worker\_connections 1024;  }  http {  include mime.types;  default\_type application/octet-stream;  #log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  # '$status $body\_bytes\_sent "$http\_referer" '  # '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  #access\_log logs/access.log main;  sendfile on;  #tcp\_nopush on;  #keepalive\_timeout 0;  keepalive\_timeout 65;  #gzip on;  一个server节点就是一个虚拟主机  server {  listen 80;  server\_name localhost;  #charset koi8-r;  Html是nginx安装目录下的html目录  #access\_log logs/host.access.log main;  location / {  root html;  index index.html index.htm;  }  }  } |

可以配置多个server，配置了多个虚拟的主机。可以使用vim命令来修改，但是比较麻烦。

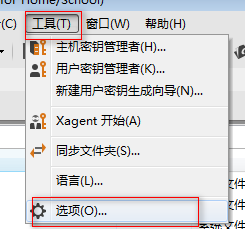
可以使用工具远程修改。

### 使用xftp配置远程编辑器

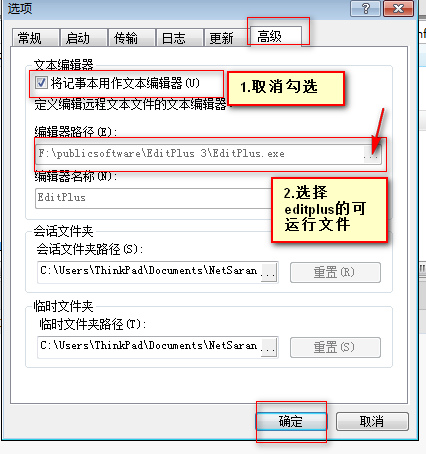
直接使用VIM命令不太方便，一般需要修改远程工程中的文件，除了使用vim命令外，还可以配置文本编辑器，直接作用于远程文件。

第一步：使用xshell连接到虚拟机，先新建远程文件输出

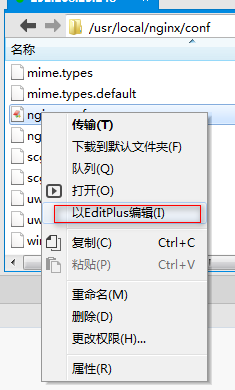
第二步：选择选项



第三步：选择高级》取消勾选 》 点击... （选择文本编辑的可运行文件）



第四步：使用



### 添加配置虚拟主机

修改nginx的配置文件，如下：

/usr/local/nginx/conf/nginx.conf

|  |
| --- |
| #user nobody;  worker\_processes 1;  #error\_log logs/error.log;  #error\_log logs/error.log notice;  #error\_log logs/error.log info;  #pid logs/nginx.pid;  events {  worker\_connections 1024;  }  http {  include mime.types;  default\_type application/octet-stream;  #log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  # '$status $body\_bytes\_sent "$http\_referer" '  # '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  #access\_log logs/access.log main;  sendfile on;  #tcp\_nopush on;  #keepalive\_timeout 0;  keepalive\_timeout 65;  #gzip on;  server {  listen 80;  server\_name localhost;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html;  index index.html index.htm;  }  }  server {  listen 81;  server\_name localhost;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html-81;  index index.html index.htm;  }  }  } |

修改完成需要每次配置文件修改之后都需要重新加载配置文件

[root@localhost nginx]# sbin/nginx -s reload

## 通过域名区分不同的虚拟的主机

### 什么是域名

域名就是网站。

[www.baidu.com](http://www.baidu.com)

[www.taobao.com](http://www.taobao.com)

[www.jd.com](http://www.jd.com)

一级域名：又叫顶级域名

Baidu.com

Taobao.com

Jd.com

二级域名：

[www.baidu.com](http://www.baidu.com)

Image.baidu.com

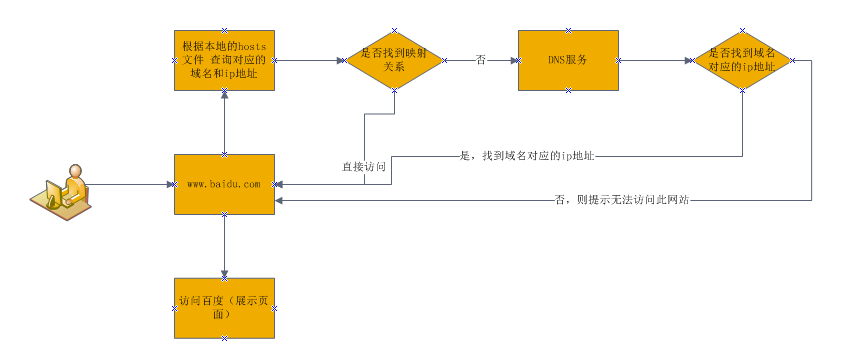
Item.baidu.com

三级域名：

1. Image.baidu.com

Aaa.image.baidu.com

域名访问网站，其本质还是通过Tcp/ip协议访问,如图：以下为通过域名访问的流程图：



DNS服务器：把域名解析为ip地址。保存的就是域名和ip的映射关系,可以简单的理解为一个MAP<KEY,VALUE>

一个域名对应一个ip地址，一个ip地址可以被多个域名绑定。

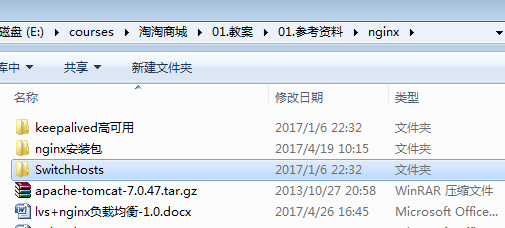
本地测试可以修改host文件。

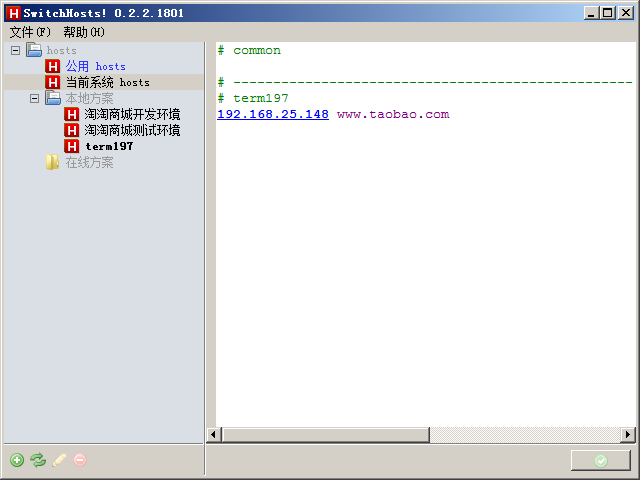
修改window的hosts文件：（C:\Windows\System32\drivers\etc）

可以配置域名和ip的映射关系，如果hosts文件中配置了域名和ip的对应关系，不需要走DNS服务器。

手动修改hosts文件比较麻烦，一般企业中使用hosts切换的工具：switchhosts 来方便切换hosts文件中的配置。

使用时必须以管理员权限运行。





### 配置域名

修改nginx的配置文件，如下：

/usr/local/nginx/conf/nginx.conf

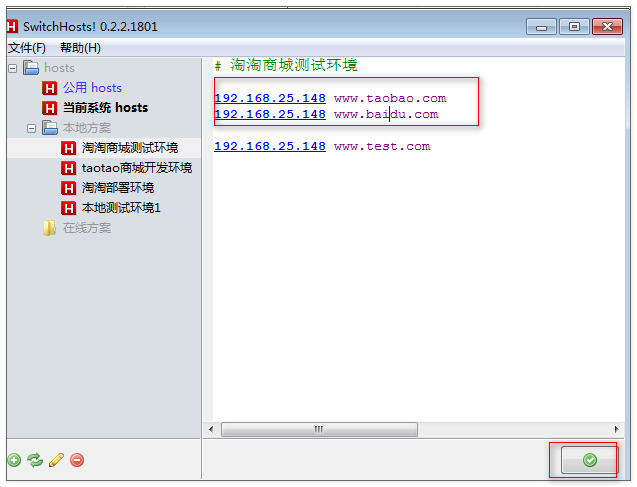
|  |
| --- |
| #user nobody;  worker\_processes 1;  #error\_log logs/error.log;  #error\_log logs/error.log notice;  #error\_log logs/error.log info;  #pid logs/nginx.pid;  events {  worker\_connections 1024;  }  http {  include mime.types;  default\_type application/octet-stream;  #log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  # '$status $body\_bytes\_sent "$http\_referer" '  # '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  #access\_log logs/access.log main;  sendfile on;  #tcp\_nopush on;  #keepalive\_timeout 0;  keepalive\_timeout 65;  #gzip on;  server {  listen 80;  server\_name localhost;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html;  index index.html index.htm;  }  }  server {  listen 81;  server\_name localhost;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html-81;  index index.html index.htm;  }  }  server {  listen 80;  server\_name www.taobao.com;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html-taobao;  index index.html index.htm;  }  }  server {  listen 80;  server\_name www.baidu.com;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html-baidu;  index index.html index.htm;  }  }  } |

添加本地域名映射关系：

域名的配置：在使用switchhosts 添加如下，并应用。

192.168.25.148 www.taobao.com

192.168.25.148 www.baidu.com



# nginx配置反向代理

## 什么是反向代理

正向代理

代理服务器

可以上网

请求转发

Pc3

Pc2

Pc1

局域网

网站

反向代理：

Tomcat(N)

网站入口

公网ip

反向代理服务器

nginx

Tomcat2

Tomcat1

反向代理服务器决定哪台服务器提供服务。

反向代理服务器不提供服务器。也只是请求的转发。

## Nginx实现反向代理

两个域名指向同一台nginx服务器，用户访问不同的域名显示不同的网页内容。

两个域名是www.sina.com.cn和www.sohu.com

nginx服务器使用虚拟机192.168.25.148

反向代理服务器

192.168.25.148

nginx

[www.sina.com.cn](http://www.sina.com.cn)

Tomcat:8080

www.sohu.com

Tomcat:8081

操作步骤：

第一步：安装两个tomcat，分别运行在8080和8081端口。

第二步：启动两个tomcat。

第三步：反向代理服务器的配置，在/usr/local/nginx/conf/nginx.conf中http节点中配置:

|  |
| --- |
| upstream tomcat1 {  server 192.168.25.148:8080;  }  server {  listen 80;  server\_name www.sina.com.cn;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  proxy\_pass http://tomcat1;  index index.html index.htm;  }  }  upstream tomcat2 {  server 192.168.25.148:8081;  }  server {  listen 80;  server\_name www.sohu.com;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  proxy\_pass http://tomcat2;  index index.html index.htm;  }  } |

第四步：nginx重新加载配置文件

./nginx -s reload

第五步：在本地配置域名

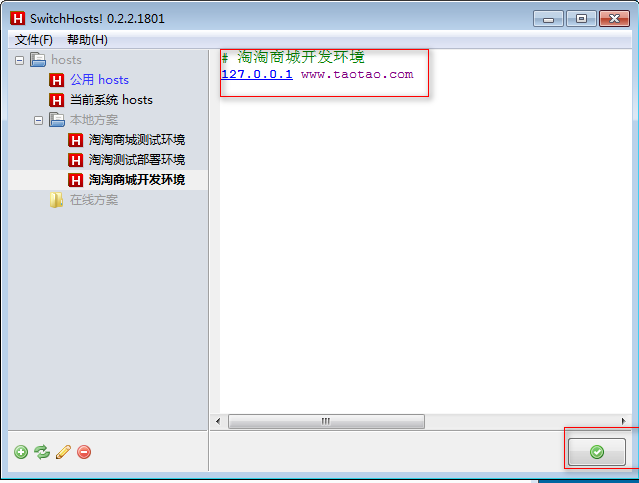
在hosts文件中添加域名和ip的映射关系，可使用switchhosts来设置

192.168.25.148 www.sina.com.cn

192.168.25.148 www.sohu.com

## 使用域名来访问系统改造

1.先配置本地域名和ip地址的映射：



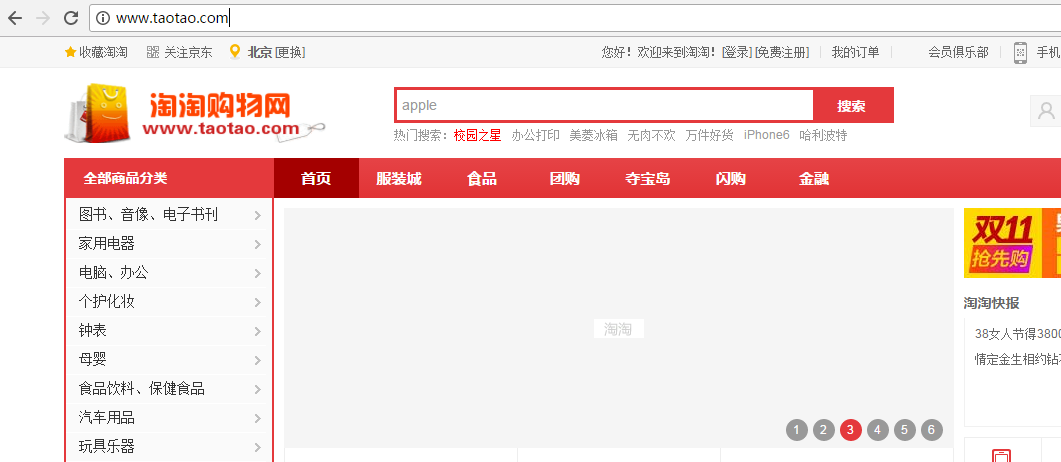
2.可使用本地windows版本的nginx 配置nginx.conf,添加server节点如下：

|  |
| --- |
| upstream tomcat1 {  server 127.0.0.1:8082;  }  server {  listen 80;  server\_name www.taotao.com;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  proxy\_pass http://tomcat1;  index index.html index.htm;  }  #error\_page 404 /404.html;  # redirect server error pages to the static page /50x.html  #  error\_page 500 502 503 504 /50x.html;  location = /50x.html {  root html;  }  # proxy the PHP scripts to Apache listening on 127.0.0.1:80  #  #location ~ \.php$ {  # proxy\_pass http://127.0.0.1;  #}  # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000  #  #location ~ \.php$ {  # root html;  # fastcgi\_pass 127.0.0.1:9000;  # fastcgi\_index index.php;  # fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;  # include fastcgi\_params;  #}  # deny access to .htaccess files, if Apache's document root  # concurs with nginx's one  #  #location ~ /\.ht {  # deny all;  #}  } |

3.windows下启动nginx

双击exe

4.查看测试效果：



# nginx配置负载均衡

如果一个服务由多台服务器提供，需要把负载分配到不同的服务器处理，需要负载均衡。

upstream tomcat2 {

server 192.168.25.148:8081;

server 192.168.25.148:8082;

}

默认的负载均衡的策略就是**轮询的方式**。

可以根据服务器的实际情况调整服务器权重。权重越高分配的请求越多，权重越低，请求越少。默认是都是1

|  |
| --- |
| upstream tomcat2 {  server 192.168.25.148:8081;  server 192.168.25.148:8082 weight=2;  } |

其他的负载均衡的策略：

1.通过IP地址的hash值 做映射。

2.通过URL的方式计算出Hash值 。

3.随机策略。

4.最少并发量。

## IP\_hash 配置

采用nginx中的的ip\_hash机制

Nginx中的ip\_hash技术能够将某个ip 的请求定向到同一台后端web机器中,这样一来这个ip 下的客户端和某个后端 web机器就能建立起稳固的session.

ip\_hash机制能够让某一客户机在相当长的一段时间内只访问固定的后端的某台真实的web服务器,这样会话就会得以保持,在网站页面进行login的时候就不会在后面的web服务器之间跳来跳去了,也不会出现登录一次的网站又提醒重新登录的情况.

Ip\_hash是在upstream配置中定义的:

upstream nginx.example.com{

server 192.168.74.235:80;

server 192.168.74.236:80;

ip\_hash;

}

server{

listen 80;

location /{

proxy\_pass

http: //nginx .example.com;

}

}

Ip\_hash机制缺陷:

(1).nginx不是最前端的服务器

ip\_hash要求nginx一定是最前端的服务器,否则nginx得不到正确ip,就不能根据ip作hash.

Eg: 使用的是squid为最前端.那么nginx取ip时只能得到squid的服务器ip地址,用这个地址来作分流肯定是错乱的

(2).nginx的后端还有其它负载均衡

假如nginx后端还有其它负载均衡,将请求又通过另外的方式分流了,那么某个客户端的请求肯定不能定位到同一台session应用服务器上,这么算起来,nginx后端只能直接指向应用服务器,或者再搭一人squid,然后指向应用服务器. 最好 的办法是用location作一次分流,将需要session的部分请求通过ip\_hash分流,剩下的走其它后端去.

# nginx的高可用（了解）

## 什么是负载均衡高可用

nginx作为负载均衡器，所有请求都到了nginx，可见nginx处于非常重点的位置，如果nginx服务器宕机，后端web服务将无法提供服务，影响严重。

为了屏蔽负载均衡服务器的宕机，需要建立一个备份机。主服务器和备份机上都运行高可用（High Availability）监控程序，通过传送诸如“I am alive”这样的信息来监控对方的运行状况。当备份机不能在一定的时间内收到这样的信息时，它就接管主服务器的服务IP并继续提供负载均衡服务；当备份管理器又从主管理器收到“I am alive”这样的信息时，它就释放服务IP地址，这样的主服务器就开始再次提供负载均衡服务。

## keepalived+nginx实现主备

### 什么是keepalived

keepalived是集群管理中保证集群高可用的一个服务软件，用来防止单点故障。

Keepalived的作用是检测web服务器的状态，如果有一台web服务器死机，或工作出现故障，Keepalived将检测到，并将有故障的web服务器从系统中剔除，当web服务器工作正常后Keepalived自动将web服务器加入到服务器群中，这些工作全部自动完成，不需要人工干涉，需要人工做的只是修复故障的web服务器。

### keepalived工作原理

keepalived是以VRRP协议为实现基础的，VRRP全称Virtual Router Redundancy Protocol，即虚拟路由冗余协议。

虚拟路由冗余协议，可以认为是实现路由器高可用的协议，即将N台提供相同功能的路由器组成一个路由器组，这个组里面有一个master和多个backup，master上面有一个对外提供服务的vip（VIP = Virtual IP Address，虚拟IP地址，该路由器所在局域网内其他机器的默认路由为该vip），master会发组播，当backup收不到VRRP包时就认为master宕掉了，这时就需要根据VRRP的优先级来选举一个backup当master。这样的话就可以保证路由器的高可用了。

keepalived主要有三个模块，分别是core、check和VRRP。core模块为keepalived的核心，负责主进程的启动、维护以及全局配置文件的加载和解析。check负责健康检查，包括常见的各种检查方式。VRRP模块是来实现VRRP协议的。

详细参考：Keepalived权威指南中文.pdf

### keepalived+nginx实现主备过程

#### 初始状态

VIP虚ip：192.168.101.100

nginx负载均衡服务器（主）

192.168.101.3

keepalived

nginx负载均衡服务器（备）

192.168.101.3

keepalived

心跳

tomat服务器群

#### 主机宕机

VIP虚ip：192.168.101.100

nginx负载均衡服务器（主）

192.168.101.3

keepalived

nginx负载均衡服务器（备）

192.168.101.3

keepalived

心跳

tomat服务器群

#### 主机恢复

VIP虚ip：192.168.101.100

nginx负载均衡服务器（主）

192.168.101.3

keepalived

nginx负载均衡服务器（备）

192.168.101.3

keepalived

心跳

tomat服务器群

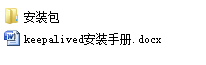
#### 高可用环境

两台nginx，一主一备：192.168.101.3和192.168.101.4

两台tomcat服务器：192.168.101.5、192.168.101.6

#### 安装keepalived

分别在主备nginx上安装keepalived，参考“安装手册”进行安装：



# SSO系统

## 什么是SSO系统

SSO英文全称Single Sign On，单点登录。SSO是在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。它包括可以将这次主要的登录映射到其他应用中用于同一个用户的登录的机制。它是目前比较流行的企业业务整合的解决方案之一。

## 为什么要有单点登录系统

### 传统的登录实现方式



此方式在只有一个web工程时是没有问题。

### 集群环境下

2、刷新页面，要求再次登录

Session

User1

1、访问用户中心，要求用户登

负载均衡服务器

nginx

Session

User1

webapp

Tomcat2

webapp

Tomcat1

登录处理

Redis服务器

Session服务器

登录页面

Session

User1

集群环境下会出现要求用户多次登录的情况。

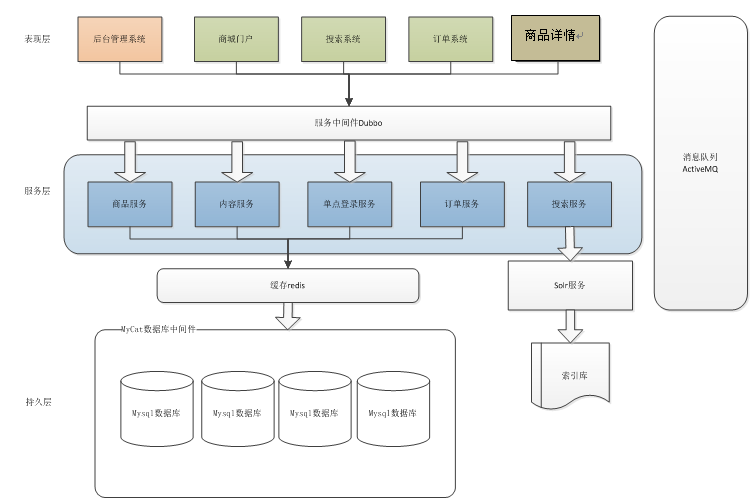
解决方案：

1. 配置tomcat集群。配置tomcatSession复制。节点数不要超过5个。
2. 可以使用Session服务器，保存Session信息，使每个节点是无状态。需要模拟Session。

单点登录系统是使用redis模拟Session，实现Session的统一管理,解决Session的共享问题.

# SSO系统的实现

系统架构图:



需要创建一个sso服务工程，可以参考taotao-manager创建。

## 工程搭建

Taotao-sso（pom聚合工程）

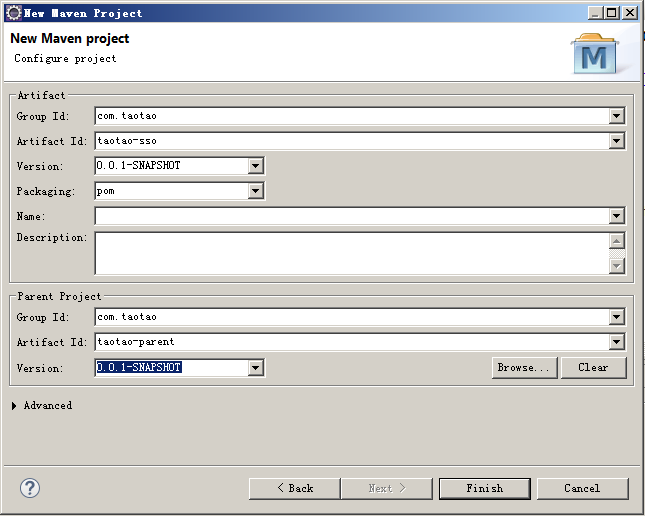
|--taotao-sso-interface（jar）

|--taotao-sso-Service（war）

Taotao-sso-web(war)

可以参考taotao-manager,taotao-manager-web创建

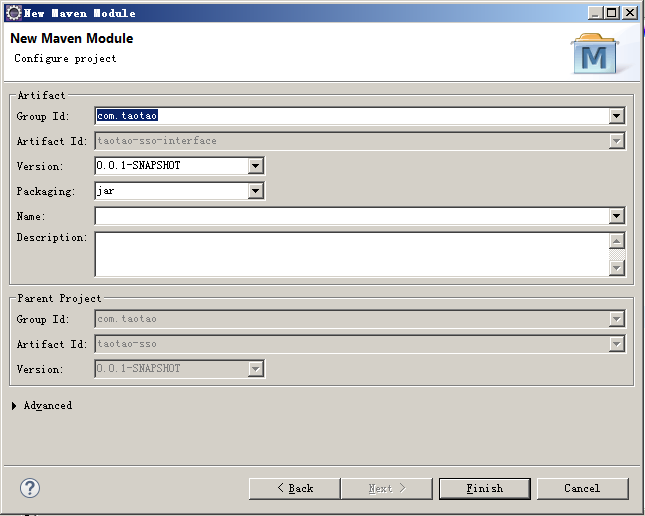
## Taotao-sso



### Pom文件

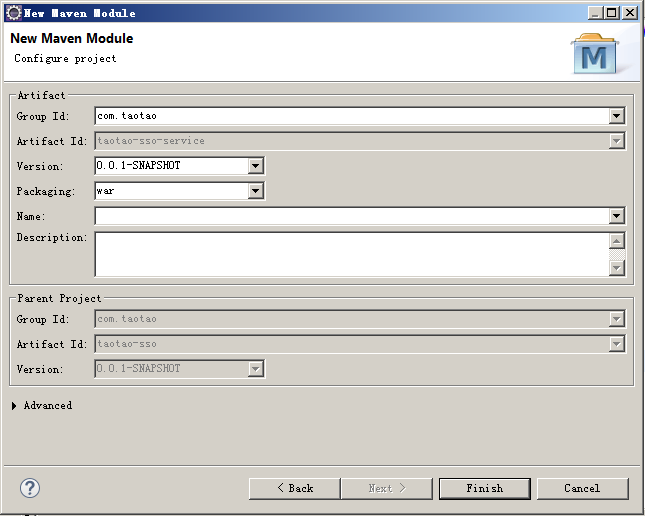
|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>com.taotao</groupId>  <artifactId>taotao-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  </parent>  <groupId>com.taotao</groupId>  <artifactId>taotao-sso</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>pom</packaging>  <dependencies>  <dependency>  <groupId>com.taotao</groupId>  <artifactId>taotao-common</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  </dependencies>  <!-- 配置tomcat插件 -->  <build>  <plugins>  <plugin>  <groupId>org.apache.tomcat.maven</groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <configuration>  <port>8087</port>  <path>/</path>  </configuration>  </plugin>  </plugins>  </build>  </project> |

## taotao-sso-interface





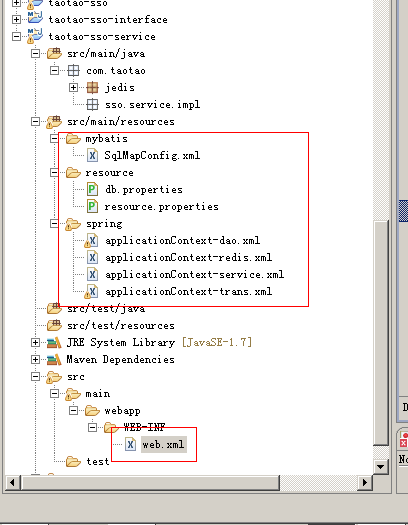
## taotao-sso-service



### Pom文件

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>com.taotao</groupId>  <artifactId>taotao-sso</artifactId>  <version>0.0.1-SNAPSHOT</version>  </parent>  <artifactId>taotao-sso-service</artifactId>  <packaging>war</packaging>  <dependencies>  <dependency>  <groupId>com.taotao</groupId>  <artifactId>taotao-manager-dao</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>com.taotao</groupId>  <artifactId>taotao-sso-interface</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  <!-- spring的依赖 -->  <!-- Spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jms</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  </dependency>  <!-- dubbo相关 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>dubbo</artifactId>  <!-- 排除依赖 -->  <exclusions>  <exclusion>  <groupId>org.springframework</groupId>  <artifactId>spring</artifactId>  </exclusion>  <exclusion>  <groupId>org.jboss.netty</groupId>  <artifactId>netty</artifactId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.apache.zookeeper</groupId>  <artifactId>zookeeper</artifactId>  </dependency>  <dependency>  <groupId>com.github.sgroschupf</groupId>  <artifactId>zkclient</artifactId>  </dependency>  <!-- Redis客户端 -->  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  </dependency>  </dependencies>  </project> |

## 框架整合



## 表现层工程

表现层工程包含登录和注册页面，需要调用sso服务实现。

给app提供服务，restful形式的服务。

Taotao-sso-web（war包）

可以参考taotao-manager-web创建。

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>com.taotao</groupId>  <artifactId>taotao-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  </parent>  <groupId>com.taotao</groupId>  <artifactId>taotao-sso-web</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>war</packaging>  <dependencies>  <dependency>  <groupId>com.taotao</groupId>  <artifactId>taotao-sso-interface</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  <!-- Spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jms</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  </dependency>  <!-- JSP相关 -->  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>servlet-api</artifactId>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jsp-api</artifactId>  <scope>provided</scope>  </dependency>  <!-- dubbo相关 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>dubbo</artifactId>  <!-- 排除依赖 -->  <exclusions>  <exclusion>  <groupId>org.springframework</groupId>  <artifactId>spring</artifactId>  </exclusion>  <exclusion>  <groupId>org.jboss.netty</groupId>  <artifactId>netty</artifactId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.apache.zookeeper</groupId>  <artifactId>zookeeper</artifactId>  </dependency>  <dependency>  <groupId>com.github.sgroschupf</groupId>  <artifactId>zkclient</artifactId>  </dependency>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  </dependency>  </dependencies>  <!-- 配置tomcat插件 -->  <build>  <plugins>  <plugin>  <groupId>org.apache.tomcat.maven</groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <configuration>  <port>8088</port>  <path>/</path>  </configuration>  </plugin>  </plugins>  </build>  </project> |

