需求零

**搭建基于Nginx+Tomcat的网站（实现功能包括：注册、登录、详情、支付等）。Nginx用于实现反向代理，使得动静分离、负载均衡，并统一日志格式。**

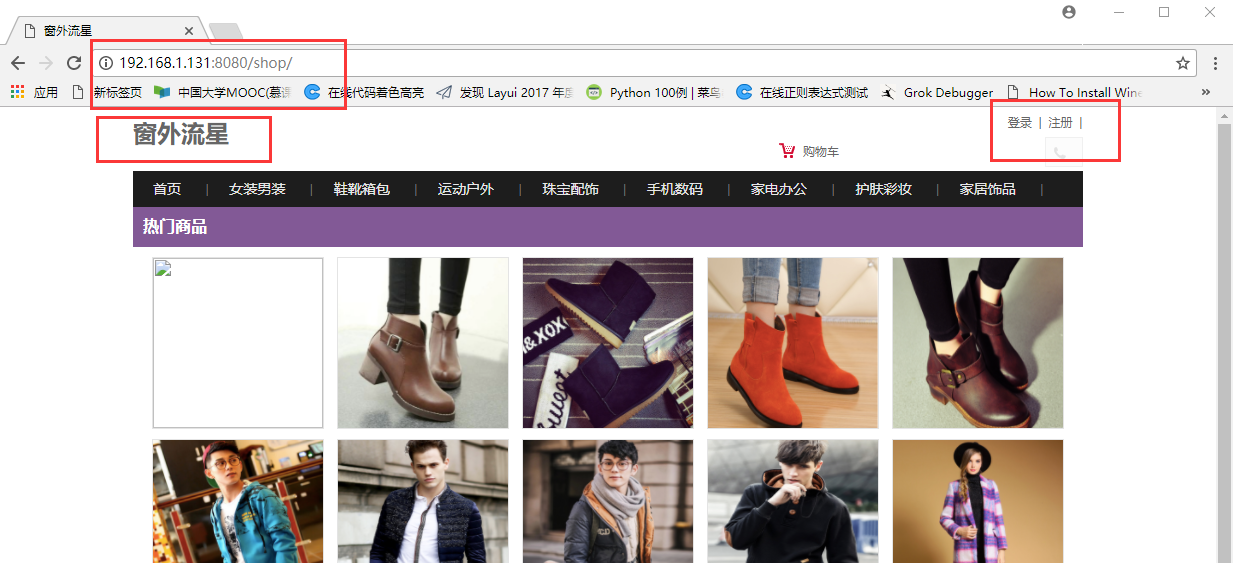
需求0.1：搭建基于Nginx+Tomcat的网站

使用Eclipse运行网站项目，Tomcat只处理JSP动态页面，Nginx来进行反向代理，第一步代理，实现动静网页分离，提高了服务器处理请求的性能，实现负载均衡。

1. 构建网站
2. 通过Eclipse直接打开tomcat，端口不能占用，默认8080端口。Windows+R打开cmd，输入指令netstat –ano查看占用的端口号，再在任务管理器中将相应占用端口的进程关闭。
3. 在进程关闭端口释放的前提下，将下部的对话框中的项目去除，在project选项中点击clean清除缓存，再给tomcat服务add添加项目，然后直接开启tomcat服务，网站搭建完成。
4. 修改网址、端口为自定义：在Eclipse左侧servers的下属文件中找到server.xml，将<Connector connectionTimeout="20000" port="7070" protocol="HTTP/1.1" redirectPort="8443"/>的端口号更改。同时在tomcat的主目录，进入conf文件夹，找到server.xml文件<Host appBase="webapps" autoDeploy="true" name="192.168.1.3" unpackWARs="true">可将域名更改。更改内容的时候不可乱改，像<Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">和<Engine defaultHost="localhost" name="Catalina">等处的localhost不可更改。
5. Eclipse每次修改server.xml文件后，总是报出Could not clean server of obsolete files错误：workspace下的servers项目的server.xml文件和 发布之后的路径下: /workspace/.metadata/.plugins/org.eclipse.wst.server.core/tmp\*/config 中的server.xml文件不一致其实workspace下servers项目中的server.xml是tomcat的config下的一份copy,有时候手动修改了servers项目下的server.xml导致文件不一致。不过有时候因为项目需要,我们需要手动修改server.xml,所以导致发布的时候 两个文件不一致,把workspace/servers/../server.xml 文件内容copy 到 tmp\*/config/server.xml 即可。
6. 局域网其他设备不能访问电脑的Tomcat服务，Windows 10自带的防火墙屏蔽了8080端口，只需要重新设置规则即可：
7. “控制面板→Windows防火墙”，点击左侧菜单中的“高级设置”。
8. 在“高级安全Windows防火墙”设置面板中，鼠标右键点击“入站规则”，从弹出菜单中点击“新建规则”。在“要创建的规则类型”中，点选“端口”，下一步，选择“TCP”，点选“特定本地端口”后填入端口号8080。点击“下一步”，点选“允许连接”，再点“下一步”，根据用户的实际情况和需求勾选域、专用或共用等选项，最后为这个规则命名即可。
9. 在正常clean后添加项目并启动时若只出现红字没有黑字出现，说明服务器启动失败，反复操作后依然如此。可将Tomcat服务器全部删除，重新添加服务器，clean之后再导入项目。
10. 进入网站进行注册用户时，由于网站没有配置邮箱验证，所以需要手动修改Mysql数据库，打开MySQL Workbench，根据代码中jdbc.user = root，jdbc.password = 123123，登陆数据库修改，打开shop数据库，打开shop数据库，右击user表点击select rows limit 1000，即可查看数据库中数据。选中表内的数据state由0改为1，点击apply，完成验证，之后就可以进行用户登陆了。
11. curl –v –L <http://192.168.1.109:2417/shop>用来测试是否能够向网站发送请求。返回302表示重定向，-L参数使得curl支持重定向到指定网站。



1. 疑问：<Host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">将域名name保留为localhost后，端口设置为7070，局域网内设备可访问。
2. 实现注册、登陆、下单等功能

二.搭建Nginx实现反向代理并规定日志格式

1. 搭建Nginx

1）、下载源码包

下载地址：http://nginx.org/download/

这里选择的时1.8.0版本

2）、解压源码包

tar -zvxf nginx-1.8.0.tar.gz

3）、安装依赖包：

yum -y install gcc gcc-c++ make libtool zlib zlib-devel openssl openssl-devel pcre pcre-devel

4）、执行configure：

进入nginx路径，执行：

./configure --prefix=/usr/local/nginx #默认安装路径为/usr/local/nginx，可调

注意：在确定安装路径后，执行编译安装，会在安装路径（此处为/usr/local）下产生一个新的nginx文件夹，与解压文件（不包含sbin）不同，该nginx文件包含conf、sbin等。

5）、编译：

就在configure路径下make

6）、安装：

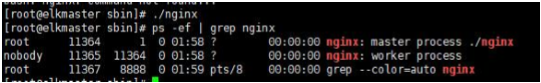
就在configure路径下make install

7）、启动：

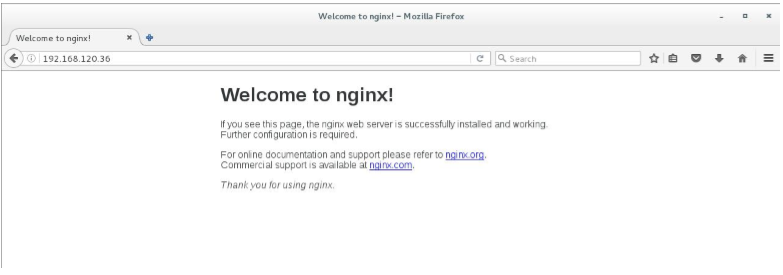
vim /etc/selinux/config将SELINUX=enforcing 修改为 SELINUX=disabled ，重启生效。关闭防火墙。进入nginx/sbin下，执行nginx

8）、检验：

查看nginx进程



打开浏览器，输入ip，即可看到页面



若页面显示403报错，查看logs文件下日志可知出现403 forbidden (13: Permission denied)，主要原因是没有在默认配置路径/usr/local下安装nginx。解决办法：需要修改配置文件中location / 模块下将原来的root html;修改为root /root/Desktop/nginx/html;#/root/Desktop/nginx 为nginx实际安装路径。然后在nginx.conf文件开头添加user root; 。修改目录的读写权限chmod -R 777 /root/Desktop/nginx。最后重启nginx使修改生效。

9）、停止

在sbin目录下调用 nginx -s stop 快速停止nginx

调用 nginx -s quit 完整有序的停止nginx

10）、报错

进程性的 nginx -s stop总是会报错误nginx: [error] open() "/alidata/server/nginx/logs/nginx.pid" failed (2: No such file or directory)，这应该是因为把nginx进程杀死后pid丢失了，nginx.pid此文件夹包含描述系统启动以来系统信息的数据，此文件夹下的文件必须在启动过程初期清除（删除或归零）。下一次再开启nginx -s reload时无法启动，重装nginx可以解决这个问题。也可以通过查看nginx进程，然后kill -9 nginx进程号后在sbin目录下./nginx -c /usr/local/nginx/conf/nginx.conf重启。

2.构建反向代理

1）、概述：

现有一台tomcat服务器部署好一个名为shop的java项目。IP地址为：192.168.34.23（以当前局域网分配IP为准） ，端口号为7070。一台nginx，IP为192.168.120.40 （以当前虚拟机分配IP为准）。

2）、配置：

vim nginx.conf

在http{}中设置upstream模块

upstream shop{

server 192.168.34.23:7070 ;

}

“shop”即为需要访问的名字，可修改。

“server”是tomcat服务器所在IP，要根据实际情况修改。否则反向代理失败。

然后在location中添加：proxy\_pass http://shop ;

proxy\_pass 是Nginx 反向代理最重要的一个配置，该配置接受一个参数， URL ，也就是 Nginx 转发的目的地。

设置服务器端口号

server {

listen 8123；

server\_name localhost;

}

3）、启动：

重新启动nginx服务器

./nginx -s stop

./nginx

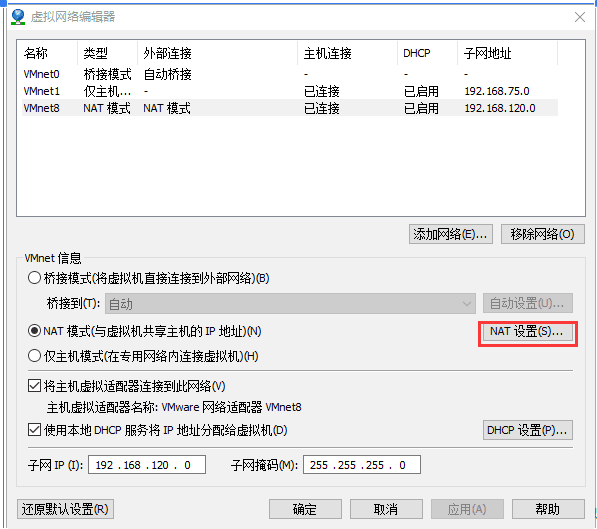
访问192.168.120.40:8123/shop

效果为用户访问以nginx服务器所在虚拟机IP为网址，设定端口8123为目标端口时，反向代理跳转至web网页。

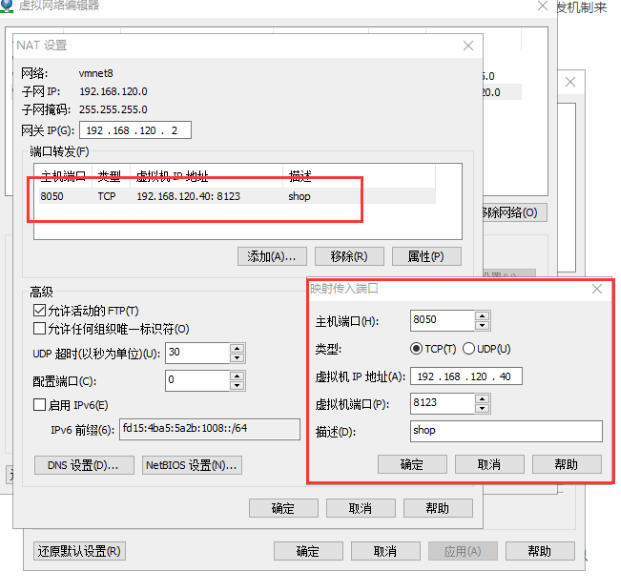
4）、同一局域网访问：

在这种情况下，可以访问虚拟机的nginx地址，但是在同一局域网下的其他电脑不可以。这就是nat模式的一个弊端。别的电脑可以识别本机192.168.34.27 的ip，这是因为在同一局域网内。但是192.168.120.40 是虚拟机的IP，通过虚拟网络适配器再次进行组网，所以192.168.120.40 对于192.168.34.23 这台机子来说是“外网IP”，无法识别，这个时候需要利用VMware里的端口转发机制来进行转发，使得局域网内的其他机器申请可以访问到我虚拟机里的nginx服务器，这种情况下类似于虚拟机IP和本机IP开放在一个局域网下。

1、打开虚拟网络编辑器



2、编辑信息



这里的8123为虚拟机的nginx端口号（上面已设置）--即通过端口号8123也可以访问页面。8050为本物理机的端口号（注意查看是否端口占用）。也就是说，局域网内通过访问192.168.34.27:8050/shop 等同于虚拟机访问192.168.120.40:8123/shop

若虚拟机和主机处于桥接模式，同在一个网段下却无法ping通则将虚拟机设置里网络适配器选项添加复制物理网络连接状态。然后重启网络

systemctl restart network

5）、参考：

https://www.cnblogs.com/yuanjava/archive/2017/05/13/6850764.html

需求0.2：统一日志格式

根据需求二所分析的几种恶意攻击数据的特征来统一规定Nginx日志字段内容，添加cookie字段、请求时间字段等，便于后续的数据特征提取。

一.规定Nginx日志字段并添加特征字段

修改conf下的Nginx.conf，做如下配置

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

upstream shop{

server 192.168.34.23:7070 ;

}

log\_format main '$time\_local $remote\_addr "$my\_cookie" "$request\_completion" $status ' '$request\_time $body\_bytes\_sent ';

#注意此处的my\_cookie前一定要加$才能变成一个变量

sendfile on;

keepalive\_timeout 65;

server {

listen 8123;

server\_name localhost;

set $my\_cookie "";

if ( $http\_cookie ~\* "JSESSIONID=(.+?)(?=;|$)" ){

set $my\_cookie $1;

}

# if ( $http\_cookie ~\* "$Cookie=(.+?)(?=;|$)" )

# {

# set $Cookie $1;

# }

access\_log /root/Desktop/access.log main; #确定日志输出路径

location / {

root html;

index index.html index.htm;

proxy\_pass <http://shop>; # Nginx 转发的目的地

proxy\_cookie\_path /shop /;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

error\_page 500 502 503 504 /50x.html;

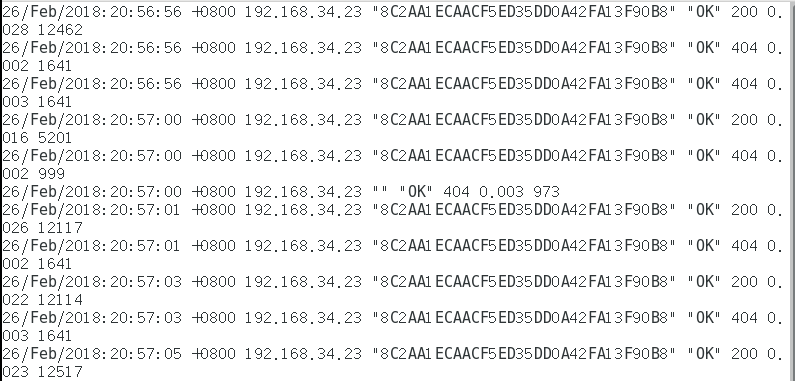
location = /50x.html {

root html;

}

}  
}

得到的Nginx日志如下：



二.注意

1.

如果只是host、端口转换，则cookie不会丢失。 例如：

location /shop {

proxy\_pass http://127.0.0.1:8080/shop;

}

通过浏览器访问http://127.0.0.1/shop时，浏览器的cookie内有jsessionid。 再次访问时，浏览器会发送当前的cookie。

如果路径也变化了，则需要设置cookie的路径转换，nginx.conf的配置如下

location /proxy\_path {

proxy\_pass http://127.0.0.1:8080/shop;

}

通过浏览器访问http://127.0.0.1/proxy\_path时，浏览器的cookie内没有jsessionid。 再次访问时，后台当然无法获取到cookie了。

此时需要加上路径转换：proxy\_cookie\_path /shop /proxy\_path;

则可以将project的cookie输出到proxy\_path上。 正确的配置是：

location / {

proxy\_pass http://127.0.0.1:8080/shop

proxy\_cookie\_path /shop /;

}

2.

参数proxy\_hide\_header是一组不会被传给 client 的 headers，若添加参数

proxy\_hide\_header Set\_Cookie则日志中不会显示cookie。