转自：http://blog.csdn.net/bruce\_6/article/details/38228299

相信很多人都听过nginx，这个小巧的东西慢慢地在吞食apache和IIS的份额。那究竟它有什么作用呢？可能很多人未必了解。

 说到反向代理，可能很多人都听说，但具体什么是反向代理，很多人估计就不清楚了。摘一段百度百科上的描述：

反向代理（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。

 这里讲得很直白。反向代理方式实际上就是一台负责转发的代理服务器，貌似充当了真正服务器的功能，但实际上并不是，代理服务器只是充当了转发的作用，并且从真正的服务器那里取得返回的数据。这样说，其实nginx 完成的就是这样的工作。我们让nginx监听一个端口，譬如80端口，但实际上我们转发给在8080端口的tomcat，由它来处理真正的请求，当请求完成后，tomcat返回，但数据此时没直接返回，而是直接给nginx，由nginx进行返回，这里，我们会以为是nginx进行了处理，但实际上进行处理的是tomcat。

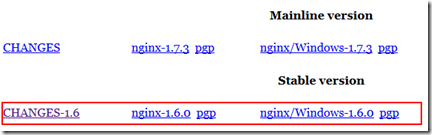
 说到上面的方式，也许很多人又会想起来，这样可以把静态文件交由nginx来进行处理。对，很多用到nginx的地方都是作为静态伺服器，这样可以方便缓存那些静态文件，比如CSS，JS，html，htm等文件。

闲话就不多说了，我们直接来看看nginx怎么使用。

**一、下载**

nginx官网：[http://nginx.org/](http://nginx.org/" \o "http://nginx.org/" \t "_blank)

直接到nginx官方下载地址下载即可：[http://nginx.org/en/download.html](http://nginx.org/en/download.html" \o "http://nginx.org/en/download.html" \t "_blank)，根据各自的操作系统版本下载。本文选择最新的[nginx-1.6.0](http://nginx.org/download/nginx-1.6.0.tar.gz" \t "_blank)windows稳定版本。

[](http://img.blog.csdn.net/20140728131046415)

**二、安装**

安装很简单，直接解压到指定的文件目录即可。解压后的目录如下：

[](http://img.blog.csdn.net/20140728130822062)

双击nginx.exe，发现一闪而过的窗口，代表nginx服务已经启动。

 也可以使用命令进行启动，使用命令行进入nginx根目录下：

1. E:\programs\nginx-1.6.0>start nginx

其他相关命令：

1. nginx -s stop    快速退出
3. nginx -s quit    优雅退出
5. nginx -s reload    更换配置，启动新的工作进程，优雅的关闭以往的工作进程
7. nginx -s reopen     重新打开日志文件

查看任务管理器：

[image](http://img.blog.csdn.net/20140728131049192)

可能不止一个服务存在（我的就有5个服务(┬＿┬)），下面验证是否已经成功启动，直接访问[http://localhost](http://localhost/" \t "_blank)可以看到如下图，代表安装启动成功：

[](http://img.blog.csdn.net/20140728130832453)

**三、配置**

nginx文件目录下有一个conf文件夹，其中有好几个文件，其他先不管，我们打开nginx.conf，可以看到一段：

1. server {
2. listen       80;
3. server\_name  localhost;
5. #charset koi8-r;
7. #access\_log  logs/host.access.log  main;
9. location / {
10. root   html;
11. index  index.html index.htm;
12. }

这段代码在server里面，相当于一个代理服务器，当然可以配置多个。

下面我们仔细来分析一下：

listen：表示当前的代理服务器监听的端口，默认的是监听80端口。注意，如果我们配置了多个server，这个listen要配置不一样，不然就不能确定转到哪里去了。

server\_name：表示监听到之后需要转到哪里去，这时我们直接转到本地，这时是直接到nginx文件夹内。

location：表示匹配的路径，这时配置了/表示所有请求都被匹配到这里

root：里面配置了root这时表示当匹配这个请求的路径时，将会在这个文件夹内寻找相应的文件，这里对我们之后的静态文件伺服很有用。

index：当没有指定主页时，默认会选择这个指定的文件，它可以有多个，并按顺序来加载，如果第一个不存在，则找第二个，依此类推。

下面的error\_page是代表错误的页面，这里我们暂时不用，先不管它。

那我们知道了具体的配置了，怎么让它访问localhost时转到tomcat时。实际上就修改两个地方：

1. server\_name localhost:8080;
3. location / {
5. proxy\_pass http://localhost:8080;
7. }

我们就修改了上面两个地方，我的tomcat在8080端口，可以根据自己的需要修改。这里有一个新元素proxy\_pass，它表示代理路径，相当于转发，而不像之前说的root必须指定一个文件夹。

此时我们修改了文件，是不是就意思着必须先关了nginx再重新启动了，其实不必，nginx可以重新加载文件的。

我们直接运行：

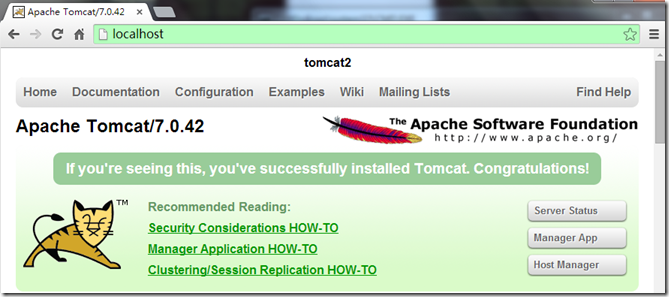
1. nginx -s reload

如果不想直接加载，而只是想检查配置文件有没有问题，可以直接输入:

1. nginx -t

这可以检查配置文件中是否有错。 下面我们所有的修改都假设我们修改完成后运行了nginx -s reload进行重新加载配置文件，请注意。

一切没问题了，然后我们再重新打开http://localhost，我们看到下面的页面：

[](http://img.blog.csdn.net/20140728131059691)

这时，我们发现它并不是刚才的welcome页面了，而是tomcat的管理页面了，不管我们点击什么链接都是没问题的，相当于直接访问[http://localhost:8080](http://localhost:8080/" \t "_blank)一样。

**四、配置集群负载均衡**

有些人会说，这些都只会找一台服务器，但如果我们想在一台服务器挂了的时候，自动去找另外一台，这怎么办？这实际上nginx都考虑到了。

这时，我们之前用的proxy\_pass就有大用途了。

我们把之前的第一个例子，即全部都代理的修改一下：

最后修改如下：

1. upstream local\_tomcat {
3. server localhost:18080;
5. }

8. server{
9. location / {
11. proxy\_pass http://local\_tomcat;
13. }
15. #......其他省略
17. }

 我们在server外添加了一个upstream，而直接在proxy\_pass里面直接用http://+upstream的名称来使用。

我们还是直接来http://localhost，还是和第一个一样的效果，所有链接都没问题，说明我们配置正确。

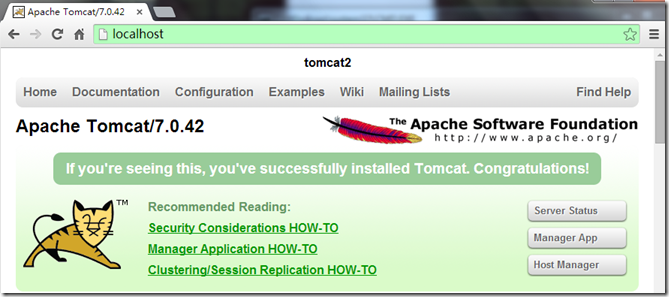
upstream中的server元素必须要注意，不能加http://，但proxy\_pass中必须加。

我们刚才说可以在一个服务器挂了的情况下连到另外一个，那怎么弄呢？

其实很简单，在upstream中的local\_tomcat中配置多一个server。比如我现在弄多一个tomcat，端口在18081，所以我们配置如下：

1. upstream local\_tomcat {
2. server localhost:18080;
3. server localhost:18081;
4. }

此时，我们关闭tomcat，重新来运行[http://localhost](http://localhost/" \t "_blank)看看效果：

[](http://img.blog.csdn.net/20140728131110814)

 我修改了index.jsp页面，增加日志输出信息，便于观察。注意：左上角小猫头上的：tomcat2、tomcat1。说明访问了不同的tomcat。

 但有时我们就不想它挂的时候访问另外一个，而只是希望一个服务器访问的机会比另外一个大，这个可以在server最后加上一个weight=数字来指定，数字越大，表明请求到的机会越大。

1. upstream local\_tomcat {
3. server localhost:18080 weight=1;
5. server localhost:18081 weight=5;
7. }

这时我们给了18081一个更高的权值，让它更有机会访问到，实际上当我们刷新[http://localhost](http://localhost/" \t "_blank)访问的时候发现18081访问机率大很多，18080几乎没机会访问，一般情况下，如果我们必须这样用，不要相关太大，以免一个服务器负载太大。

当然,server还有一些其他的元素，比如down表示暂时不用到该服务器等等。这些可以参考nginx的wiki。也许写了一大堆，有人会有问题，那nginx怎么关闭呢？这倒是个问题，其实直接运行nginx -s stop就可以关闭了。

 至此window下nginx+tomcat负载均衡配置结束，关于tomcat Session的问题通常是采用memcached，或者采用nginx\_upstream\_jvm\_route ，他是一个 Nginx 的扩展模块，用来实现基于 Cookie 的 Session Sticky 的功能。如果tomcat过多不建议session同步，server间相互同步session很耗资源，高并发环境容易引起Session风暴。请根据自己应用情况合理采纳session解决方案。