1.nginx+tomcat怎么设置session共享？

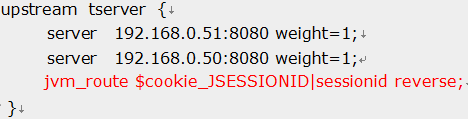
* 1. 下载安装jvm\_route:http ://nginx-upstream-jvm-route.googlecode.com/svn/trunk/nginx-upstream-jvm-route-read-only

cd /usr/src/nginx-1.2.1

patch - p0 < /usr/local/nginx/nginx\_upstream-jvm-route-read-only/jvm\_route.patch

./configure --add-module=/usr/local/nginx/nginx-upstream-jvm-route-read-only

* 1. 修改nginx的配置文件：加入nginx.confjvm\_route $cookie\_JSESSIONID|sessionid reverse;



* 1. 分别打开两台tomcat的server.xml,修改Cluster：

<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster" channelSendOptions="8">

<Manager className="org.apache.catalina.ha.session.DeltaManager"

expireSessionsOnShutdown="false" notifyListenersOnReplication="true"/>

<Channel className="org.apache.catalina.tribes.group.GroupChannel">

<Membership className="org.apache.catalina.tribes.membership.McastService"

address="224.0.0.4"

port="45564"

frequency="500"

dropTime="3000"/>

<Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver"

address="192.100.0.181"

port="4000"

autoBind="100"

selectorTimeout="5000"

maxThreads="6"/>

<!-- timeout="60000"-->

<Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">

<Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender" />

</Sender>

<Interceptor className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>

<Interceptor className="org.apache.catalina.tribes.group.interceptors.MessageDispatch15Interceptor"/>

<Interceptor className="org.apache.catalina.tribes.group.interceptors.ThroughputInterceptor"/>

</Channel>

<Valve className="org.apache.catalina.ha.tcp.ReplicationValve" filter=""/>

<Valve className="org.apache.catalina.ha.session.JvmRouteBinderValve"/>

<Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer"

tempDir="/tmp/war-temp/"

deployDir="/tmp/war-deploy/"

watchDir="/tmp/war-listen/"watchEnabled="false"/>

<ClusterListener className="org.apache.catalina.ha.session.JvmRouteSessionIDBinderListener"/>

<ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener"/>

</Cluster>

* 1. 在tomcat下conf里找到web.xml里面</web-app>前面加入以下这句话  
     <distributable/>
  2. 服务器的启动顺序如下:

tomcat1 --> |tomcat2 --> |nginx

先启动tocmat1,等tomcat1启动完全的时候再启动tomcat2，等两个tocmat全启动之后，再启动nginx

* 1. 设置完毕

2.Nginx的log格式在哪设置？

* 1. 写一个脚本，做个定时任务：
  2. #!/bin/sh

NGX\_INSTALL\_PATH=/home /nginx;

NGX\_LOG\_PATH=/home /nginx/logs;

#日志的后缀为2013080511的格式，日志切分可定位每小时切分一次。

suffix=$(date +%Y%m%d%H);

#重命名

cd $NGX\_LOG\_PATH && mv error.log error.log.$suffix;

#重启nginx，生成新的日志文件.

cd $NGX\_INSTALL\_PATH && ./sbin/nginx -s reload;

3.Mysql的主从配置+mysql索引

* 1. 主从配置：
  2. 需要两台机器，安装mysql，两台机器要在相通的局域网内

主机A: 192.168.1.100

从机B:192.168.1.101

可以有多台从机

1、先登录主机 A

mysql>GRANT REPLICATION SLAVE ON \*.\* TO ‘backup’@’192.168.1.101‘ IDENTIFIED BY ‘123456’;

赋予从机权限，有多台丛机，就执行多次

2、 打开主机A的my.cnf，输入

server-id = 1 #主机标示，整数

log\_bin = /var/log/mysql/mysql-bin.log #确保此文件可写

read-only=0 #主机，读写都可以

binlog-do-db=test #需要备份数据，多个写多行

binlog-ignore-db=mysql #不需要备份的数据库，多个写多行

3、打开从机B的my.cnf，输入:

server-id = 2

log\_bin = /var/log/mysql/mysql-bin.log

master-host=192.168.1.100

master-user=backup

master-pass=123456

master-port=3306

master-connect-retry=60 #如果从服务器发现主服务器断掉，重新连接的时间差(秒)

replicate-do-db =test #只复制某个库

replicate-ignore-db=mysql #不复制某个库

4、同步数据库

把主从库都启动即可自动同步，或者可以把主库的内容导出成SQL，然后在从库中运行一遍即可

5、先重启主机A的mysql，再重启从机B的mysql

6、验证

在主机A中，mysql>show master status\G;

在从机B中，mysql>show slave status\G;

能看到大致这些内容

File: mysql-bin.000001

Position: 1374

Binlog\_Do\_DB: test

Binlog\_Ignore\_DB: mysql

可以在主机A中，做一些INSERT, UPDATE, DELETE 操作，看看主机B中，是否已经被修改

* 1. mysql索引：(创建索引的好处：可以提高查询速度；创建准则：

1.如果表里有几百行记录则可以对其创建索引(表里的记录行数越多索引的效果就越明显)。

2.不要试图对表创建两个或三个以上的索引。

3.为频繁使用的行创建索引。)

1.添加PRIMARY KEY（主键索引）

mysql>ALTER TABLE `table\_name` ADD PRIMARY KEY ( `column` )

2.添加UNIQUE(唯一索引)

mysql>ALTER TABLE `table\_name` ADD UNIQUE ( `column` )

3.添加INDEX(普通索引)

mysql>ALTER TABLE `table\_name` ADD INDEX index\_name ( `column` )

4.添加FULLTEXT(全文索引)

mysql>ALTER TABLE `table\_name` ADD FULLTEXT ( `column`)

5.添加多列索引

mysql>ALTER TABLE `table\_name` ADD INDEX index\_name ( `column1`, `column2`, `column3` )

4.红帽6.5怎么装gcc和gcc++(没有gcc,没有gcc++,无网络,不能换系统)

A) 安装gcc

安装GCC依赖的\*rpm程序,必须按照顺序依次执行：

#rpm -ivh kernel-headers-2.6.18-164.el5.i386.rpm

#rpm -ivh compat-glibc-headers-2.3.4-2.26.i386.rpm

#rpm -ivh compat-glibc-2.3.4-2.26.i386.rpm

#rpm -ivh compat-libstdc++-33-3.2.3-61.i386.rpm

#rpm -ivh compat-libf2c-34-3.4.6-4.i386.rpm

#rpm -ivh compat-libgcc-296-2.96-138.i386.rpm

#rpm -ivh cpp-4.1.2-46.el5.i386.rpm

#rpm -ivh glibc-headers-2.5-42.i386.rpm

mpfr-2.4.1-6.el6.i686.rpm

#rpm -ivh glibc-devel-2.5-42.i386.rpm

#rpm -ivh libgomp-4.4.0-6.el5.i386.rpm

#rpm -ivh libstdc++-devel-4.1.2-46.el5.i386.rpm

rpm -ivh ppl-0.10.2-11.el6.i686.rpm

cloog-ppl-0.15.7-1.2.el6.i686.rpm

#rpm -ivh gcc-4.1.2-46.el5.i386.rpm

#rpm -ivh compat-gcc-34-3.4.6-4.i386.rpm

#rpm -ivh compat-gcc-34-c++-3.4.6-4.i386.rpm

G++的安装：

首先确保之前的包已经安装完毕，之后进行如下操作：

#rpm ivh libstdc++-4.1.1-52.el5.i386.rpm

#rpm ivh gcc-c++-4.1.1-52.el5.i386.rpm

安装完成后，系统就可以运行GCC和G++的命令了。

B)安装gcc++

yum install gcc-c++ libstdc++-devel

5.配置mysql连接数

a)登录数据库:  
set global max\_conntion =1000（即时生效）  
b)修改my.inf中max\_conntion(重启生效) :

在 my.cnf这个文件里加入max\_connecionts=2000

重启mysql服务：service mysqld restart

6.如何搭建服务器？（可互相访问的三台服务器A、B、C）

* 1. 设置主从服务器：在A上搭建Nginx，并分别在ABC上搭建tomcat，在Nginx的配置文件里设置tomcatA为主服务器，tomcatB、C是从服务器
  2. 分别在ABC上安装mysql，设置主从

7.Java如何生成二维码？

所需jar包：QRCode.jar

<http://download.csdn.net/detail/wangpeng047/4008532>

TwoDimensionCode类：二维码操作核心类

**[java]** [view plaincopy](http://blog.csdn.net/about58238/article/details/7494704)

1. **package** qrcode;
3. **import** java.awt.Color;
4. **import** java.awt.Graphics2D;
5. **import** java.awt.image.BufferedImage;
6. **import** java.io.File;
7. **import** java.io.IOException;
8. **import** java.io.InputStream;
9. **import** java.io.OutputStream;
11. **import** javax.imageio.ImageIO;
13. **import** jp.sourceforge.qrcode.QRCodeDecoder;
14. **import** jp.sourceforge.qrcode.exception.DecodingFailedException;
16. **import** com.swetake.util.Qrcode;
18. **public** **class** TwoDimensionCode {
20. /\*\*
21. \* 生成二维码(QRCode)图片
22. \* @param content 存储内容
23. \* @param imgPath 图片路径
24. \*/
25. **public** **void** encoderQRCode(String content, String imgPath) {
26. **this**.encoderQRCode(content, imgPath, "png", 7);
27. }
29. /\*\*
30. \* 生成二维码(QRCode)图片
31. \* @param content 存储内容
32. \* @param output 输出流
33. \*/
34. **public** **void** encoderQRCode(String content, OutputStream output) {
35. **this**.encoderQRCode(content, output, "png", 7);
36. }
38. /\*\*
39. \* 生成二维码(QRCode)图片
40. \* @param content 存储内容
41. \* @param imgPath 图片路径
42. \* @param imgType 图片类型
43. \*/
44. **public** **void** encoderQRCode(String content, String imgPath, String imgType) {
45. **this**.encoderQRCode(content, imgPath, imgType, 7);
46. }
48. /\*\*
49. \* 生成二维码(QRCode)图片
50. \* @param content 存储内容
51. \* @param output 输出流
52. \* @param imgType 图片类型
53. \*/
54. **public** **void** encoderQRCode(String content, OutputStream output, String imgType) {
55. **this**.encoderQRCode(content, output, imgType, 7);
56. }
58. /\*\*
59. \* 生成二维码(QRCode)图片
60. \* @param content 存储内容
61. \* @param imgPath 图片路径
62. \* @param imgType 图片类型
63. \* @param size 二维码尺寸
64. \*/
65. **public** **void** encoderQRCode(String content, String imgPath, String imgType, **int** size) {
66. **try** {
67. BufferedImage bufImg = **this**.qRCodeCommon(content, imgType, size);
68. File imgFile = **new** File(imgPath);
69. // 生成二维码QRCode图片
70. ImageIO.write(bufImg, imgType, imgFile);
71. } **catch** (Exception e) {
72. e.printStackTrace();
73. }
74. }
76. /\*\*
77. \* 生成二维码(QRCode)图片
78. \* @param content 存储内容
79. \* @param output 输出流
80. \* @param imgType 图片类型
81. \* @param size 二维码尺寸
82. \*/
83. **public** **void** encoderQRCode(String content, OutputStream output, String imgType, **int** size) {
84. **try** {
85. BufferedImage bufImg = **this**.qRCodeCommon(content, imgType, size);
86. // 生成二维码QRCode图片
87. ImageIO.write(bufImg, imgType, output);
88. } **catch** (Exception e) {
89. e.printStackTrace();
90. }
91. }
93. /\*\*
94. \* 生成二维码(QRCode)图片的公共方法
95. \* @param content 存储内容
96. \* @param imgType 图片类型
97. \* @param size 二维码尺寸
98. \* @return
99. \*/
100. **private** BufferedImage qRCodeCommon(String content, String imgType, **int** size) {
101. BufferedImage bufImg = **null**;
102. **try** {
103. Qrcode qrcodeHandler = **new** Qrcode();
104. // 设置二维码排错率，可选L(7%)、M(15%)、Q(25%)、H(30%)，排错率越高可存储的信息越少，但对二维码清晰度的要求越小
105. qrcodeHandler.setQrcodeErrorCorrect('M');
106. qrcodeHandler.setQrcodeEncodeMode('B');
107. // 设置设置二维码尺寸，取值范围1-40，值越大尺寸越大，可存储的信息越大
108. qrcodeHandler.setQrcodeVersion(size);
109. // 获得内容的字节数组，设置编码格式
110. **byte**[] contentBytes = content.getBytes("utf-8");
111. // 图片尺寸
112. **int** imgSize = 67 + 12 \* (size - 1);
113. bufImg = **new** BufferedImage(imgSize, imgSize, BufferedImage.TYPE\_INT\_RGB);
114. Graphics2D gs = bufImg.createGraphics();
115. // 设置背景颜色
116. gs.setBackground(Color.WHITE);
117. gs.clearRect(0, 0, imgSize, imgSize);
119. // 设定图像颜色> BLACK
120. gs.setColor(Color.BLACK);
121. // 设置偏移量，不设置可能导致解析出错
122. **int** pixoff = 2;
123. // 输出内容> 二维码
124. **if** (contentBytes.length > 0 && contentBytes.length < 800) {
125. **boolean**[][] codeOut = qrcodeHandler.calQrcode(contentBytes);
126. **for** (**int** i = 0; i < codeOut.length; i++) {
127. **for** (**int** j = 0; j < codeOut.length; j++) {
128. **if** (codeOut[j][i]) {
129. gs.fillRect(j \* 3 + pixoff, i \* 3 + pixoff, 3, 3);
130. }
131. }
132. }
133. } **else** {
134. **throw** **new** Exception("QRCode content bytes length = " + contentBytes.length + " not in [0, 800].");
135. }
136. gs.dispose();
137. bufImg.flush();
138. } **catch** (Exception e) {
139. e.printStackTrace();
140. }
141. **return** bufImg;
142. }
144. /\*\*
145. \* 解析二维码（QRCode）
146. \* @param imgPath 图片路径
147. \* @return
148. \*/
149. **public** String decoderQRCode(String imgPath) {
150. // QRCode 二维码图片的文件
151. File imageFile = **new** File(imgPath);
152. BufferedImage bufImg = **null**;
153. String content = **null**;
154. **try** {
155. bufImg = ImageIO.read(imageFile);
156. QRCodeDecoder decoder = **new** QRCodeDecoder();
157. content = **new** String(decoder.decode(**new** TwoDimensionCodeImage(bufImg)), "utf-8");
158. } **catch** (IOException e) {
159. System.out.println("Error: " + e.getMessage());
160. e.printStackTrace();
161. } **catch** (DecodingFailedException dfe) {
162. System.out.println("Error: " + dfe.getMessage());
163. dfe.printStackTrace();
164. }
165. **return** content;
166. }
168. /\*\*
169. \* 解析二维码（QRCode）
170. \* @param input 输入流
171. \* @return
172. \*/
173. **public** String decoderQRCode(InputStream input) {
174. BufferedImage bufImg = **null**;
175. String content = **null**;
176. **try** {
177. bufImg = ImageIO.read(input);
178. QRCodeDecoder decoder = **new** QRCodeDecoder();
179. content = **new** String(decoder.decode(**new** TwoDimensionCodeImage(bufImg)), "utf-8");
180. } **catch** (IOException e) {
181. System.out.println("Error: " + e.getMessage());
182. e.printStackTrace();
183. } **catch** (DecodingFailedException dfe) {
184. System.out.println("Error: " + dfe.getMessage());
185. dfe.printStackTrace();
186. }
187. **return** content;
188. }
190. **public** **static** **void** main(String[] args) {
191. String imgPath = "G:/TDDOWNLOAD/Michael\_QRCode.png";
192. String encoderContent = "Hello 大大、小小,welcome to QRCode!" + "\nMyblog [ http://sjsky.iteye.com ]" + "\nEMail [ sjsky007@gmail.com ]";
193. TwoDimensionCode handler = **new** TwoDimensionCode();
194. handler.encoderQRCode(encoderContent, imgPath, "png");
195. // try {
196. //OutputStream output = new FileOutputStream(imgPath);
197. //handler.encoderQRCode(content, output);
198. // } catch (Exception e) {
199. //e.printStackTrace();
200. // }
201. System.out.println("========encoder success");
202. String decoderContent = handler.decoderQRCode(imgPath);
203. System.out.println("解析结果如下：");
204. System.out.println(decoderContent);
205. System.out.println("========decoder success!!!");
206. }
207. }

TwoDimensionCodeImage 类：二维码图片对象

**[java]** [view plaincopy](http://blog.csdn.net/about58238/article/details/7494704)

1. **package** qrcode;
3. **import** java.awt.image.BufferedImage;
5. **import** jp.sourceforge.qrcode.data.QRCodeImage;
7. **public** **class** TwoDimensionCodeImage **implements** QRCodeImage {
9. BufferedImage bufImg;
11. **public** TwoDimensionCodeImage(BufferedImage bufImg) {
12. **this**.bufImg = bufImg;
13. }
15. @Override
16. **public** **int** getHeight() {
17. **return** bufImg.getHeight();
18. }
20. @Override
21. **public** **int** getPixel(**int** x, **int** y) {
22. **return** bufImg.getRGB(x, y);
23. }
25. @Override
26. **public** **int** getWidth() {
27. **return** bufImg.getWidth();
28. }
30. }