链滴

如何获取用户真实 IP

作者：chensy

原文链接：https://ld246.com/article/1482946609799

来源网站：链滴

在Java里面，一般情况下：我们直接这样就可以获取用户的IP了</
>
<pre class="brush: java">request.getRemoteAddr()</pre>
但实际上，我们应用的运行环境可能比较复杂，用户访问到我们的
用，中间可能经过：负载均衡代理->应用代理->应用服务器。而代理服务器很多种，有nginx
HAProxy、squid、LVS等等。
所以，应用想要获取到用户的真实IP，要依赖前面的每一个环境，
用户的IP传递过来。
假设：每一层代理都是nginx实现（实际上应该不是）
那实际的环境可能是这样的：

用户IP：183.6.19.19
公司的外网入口IP：183.6.20.20
应用的IP：192.168.21.21（内网IP）

一般来说，通用的配置是这样的
<pre class="brush: plain">proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;</pre>
其中$remote_addr指前一个代理的IP，X-Forwarded-For有一个
准：client1, proxy1, proxy2。如：183.6.19.19,183.6.20.20,183.6.21.21
一个为用户的真实IP
明确了架构，我们程序只要获取header中的X-Forwarded-For的
就可以获取到了。但这种配置有个缺点，X-Forwarded-For很容易被伪造IP
我们也可以自定义header，这样就不容易被伪造IP，比如在入口
代理服务器这样设置
<pre class="brush: plain">proxy_set_header USER-REAL-IP $remote_addr;</pre>
那程序也就可以直接从header中获取用户IP了
那我们的程序就可以这样写了（copy了网上流传比较多的一种写
，稍微修改的）
<pre class="brush: java">   /**
     * 获取当前网络ip
     * @param request
     * @return
     */
    public String getIpAddr(HttpServletRequest request){
            // 优先取自定义的header
                String ipAddress = request.getHeader("USER-REAL-IP");
        // 取不到再取能用的header
                if(ipAddress == null || ipAddress.length() == 0 || "unknown".equalsIgnoreCase(i
Address)) {
            ipAddress = request.getHeader("X-Forwarded-For");
        }
                // 最后取本机配置的IP
        if(ipAddress == null || ipAddress.length() == 0 || "unknown".equalsIgnoreCase(ipAddr
ss)) {
            ipAddress = request.getRemoteAddr();
            if(ipAddress.equals("127.0.0.1") || ipAddress.equals("0:0:0:0:0:0:0:1")){
                //根据网卡取本机配置的IP
                InetAddress inet=null;
                try {
                    inet = InetAddress.getLocalHost();
                } catch (UnknownHostException e) {
                    e.printStackTrace();
                }

```
                ipAddress= inet.getHostAddress();
            }
        }
        //对于通过多个代理的情况，第一个IP为客户端真实IP,多个IP按照','分割
        if(ipAddress!=null &amp;amp;&amp;amp; ipAddress.length()&amp;gt;15){ //"***.***.***.***".length() =
15
            if(ipAddress.indexOf(",")&amp;gt;0){
                ipAddress = ipAddress.substring(0,ipAddress.indexOf(","));
            }
        }
        return ipAddress;
    }</pre>
<p style="padding-left: 30px;">上面的代码其实是很有局限性的，如果网络架构改了，如果自定义eader改了，我们都要修改代码。所以网上流传了很多兼容各种网络架构的代码，但我觉得没必要。竟，一个应用的网络架构的演变还是需要一段时间的，不会频繁改变。而且这种获取IP的方法，应该以在运维层面形成规范，就算是网络架构有调整，但代码服务器设置的header应该保持一致。</p>
<p style="padding-left: 30px;">但实际上，代码不重要，架构怎么变也不重要，重要的是要知道当应用的网络架构，具体问题具体分析，写出合适的代码，没必要为了兼容各种很可能不会有的场景而增加代码的复杂度。</p>
<p style="padding-left: 30px;">&amp;nbsp;</p>
```