https://blog.csdn.net/chao\_1990/article/details/54405759?utm\_source=blogxgwz0

**nginx共享session方式及redis+cookie、ip\_hash方式实现**

2017年01月13日 11:19:05 [冥王x雷利](https://me.csdn.net/chao_1990) 阅读数：25312 标签： [nginx](http://so.csdn.net/so/search/s.do?q=nginx&t=blog)[共享session](http://so.csdn.net/so/search/s.do?q=%E5%85%B1%E4%BA%ABsession&t=blog)[iphash](http://so.csdn.net/so/search/s.do?q=iphash&t=blog)[redis](http://so.csdn.net/so/search/s.do?q=redis&t=blog)[java](http://so.csdn.net/so/search/s.do?q=java&t=blog)更多

个人分类： [技术经验](https://blog.csdn.net/chao_1990/article/category/2268119)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/chao\_1990/article/details/54405759

共享session应用场景，基于分布式，session的原理大致如下(以tomcat为例)，

tomcat有一套自己的session管理机制，每次创建连接的时候都会通过客户端传过来的jsessionid从session连接池中获得sessionId对应的session，如果存在，那么直接从池中获得session连接信息，如果没有通过程序中设置判断是否创建一个新的session信息保存到池中。

nginx贡献session的方式，大体上有ip\_hash(nginx自带)，redis共享session，基于cookie，基于db(各种关系型数据库和非关系型数据库)。

1、基于cookie

简单、方便，每次通过判断cookie中的用户状态信息判断用户的登录状态；但是用户信息要存在客户端，存在安全隐患，除非有相当安全的加密措施，如果加密码负载，也会 增加运算的成本。

2、基于db(各种关系型数据库和非关系型数据库)

登录信息存储到db中，虽然安全，但每次需要验证用户的登录状态是都需要从db中进行IO操作，并发量高是势必会增加数据库的负载。

3、基于redis+cookie共享session

此种方式将将用户的登录信息存储到redis中，因为是基于内存的读取，因此效率不会是响应效率的瓶颈，cookie中存储着jsessionid，不需要加密或处理，只需要存储redis中的key保存统一客户通过cookie中的key可以准确的登录信息或是其他有效的信息，此种方式，cookie的存储不需要加密计算成本，其次redis将信息存储到缓存中，存取效率高，后面会详细介绍此种方式实现过程。

4、基于ip\_hash

此种方式配置最为简单，但是有一定的局限性，其原理就是同一个ip的所有请求都会被nginx进行ip\_hash进行计算，通过结果定位到指定的后台服务器即一个用户如果ip不变，那么每次请求其实请求的都是同一后台服务器；首先最外层的代理要保证源ip在请求 的过程不会被修改，如果你的架构里在最外层方式不单单是nginx服务，而是类似于请求分发的服务服务器，那么意味着一个人的请求可能被定位到不同的服务器上，那么ip\_hash就没有意义了。

5、基于tomcat容器session

此种方式在根本上实现共享session，他的实际情况是通过tomcat管理配置将一个tomct下的session复制到其他的tomcat的session池中，实现真实上的session共享；此种方式需要兼容tomcat配置及需要对其进行扩展，依赖性太强。

已实现的共享方式：

首先配置3个web容器，tomcat1(3301)，tomcat2(3302)、tomcat3(3303)

1、ip\_hash

nginx的配置这里不再说明，主要说明ip\_hash的配置，简直是简单到节点

1. upstream backend {
2. ip\_hash;
3. server localhost:3301 max\_fails=2 fail\_timeout=30s ;
4. server localhost:3302 max\_fails=2 fail\_timeout=30s ;
5. server localhost:3303 max\_fails=2 fail\_timeout=30s ;
6. }

nginx的配置文件nginx.conf引入upstram数据,启动即可。

验证：

过访问web应用，例如http://localhost:8080/login.jsp登录后存储session信息，每次访问这个页面都需要验证session的状态，页面实现的是tomcat1表示，此ip的请求被转发到了tomcat1的服务器上，无论怎么刷新页面，页面显示的都是tomcat1，此时停掉tomcat1，再次刷新页面，页面显示的是tomcat2或是tomcat3的登录页面，此时表示nginx负载目的已经达到，但是session没有共享；局限性显而易见。

2、redis+cookie共享session

原理实现：

1）、登录成功时，将还有redis的key保存到cookie中，将登录用户的信息保存到redis中；

2）、配置拦截器，拦截需要验证登录状态的请求，从cookie中获得redis的key，从redis中获得当前cookie对应的用户信息，对其登录状态进行验证。

验证：

取消nginx的ip\_hash配置，登录web，例如例如http://localhost:8080/login.jsp登录后存储session信息，首页显示tomcat1，此时不需要关闭任何tomcat，直接不停的刷新主页就会发现主页的再tomcat1、tocmat2、tomcat3之间来回切换，表示session已正常共享。个人认为此种方式是比较推荐的。

观点：

从功能的核心来看，其目的就是共享session，究其原因就是客户端和服务之间需要一个唯一的值来建立关联关系，此值就是sessionId，无论请求哪个网站通过工具查看cookie都会有发现，有jsessionid的值，此值在于服务器建立连接以后是基本不会变的(后台主动创建新的session)。及时上面的描述的共享方式5，基于session连接池的session管理，也是要依赖sessionid的，通过查看tomcatsession管理的源码，不难发现其也是通过cookie获得的sessionid，那么如果客户端请强制禁用了cookie，该如何获得jsessionId? 之前查阅过资源(未亲自证实)，当浏览器禁用cookie后，请求的地址会被重定向，重定向后会在请求的url中拼接上jsessionid，这个容器就能获得sessionid，也就可以验证session 状态；此种场景有待证实。