链滴

【转】使用 Spring Session 做分布式会话管理

作者：honglan

原文链接：https://ld246.com/article/1496468513224

来源网站：链滴

在 Web 项目开发中，会话管理是一个很重要的部分，用于存储与用户相关的数据。通常是由符合 session 规范的容器来负责存储管理，也就是一旦容器关闭，重启会导致会话失效。因此打造一个高用性的系统，必须将 session 管理从容器中独立出来。而这实现方案有很多种，下面简单介绍下：</
>
　　第一种是使用容器扩展来实现，大家比较容易接受的是通过容器插件来实现，比如基于 Tom at 的 tomcat-redis-session-anager，基于 Jetty 的 jetty-session-redi 等等。好处是对项目来说是透明的，无需改动代码。不过前者目前还不支持 Tomcat 8，或者不太完善。个人觉得由于过于依赖容器，一旦容器升级或者更换意味着又得从新来过。并且代码不在目中，对开发者来说维护也是个问题。
　　第二种是自己写一套会话管理的工具类，包括 Session 管理和 Cookie 管理，在需要使用会的时候都从自己的工具类中获取，而工具类后端存储可以放到 Redis 中。很显然这个方案灵活性最大但开发需要一些额外的时间。并且系统中存在两套 Session 方案，很容易弄错而导致取不到数据。</
>
　　第三种是使用框架的会话管理工具，也就是本文要说的 spring-session，可以理解是替换了 Serv et 那一套会话管理，既不依赖容器，又不需要改动代码，并且是用了 spring-data-redis 那一套连接，可以说是最完美的解决方案。当然，前提是项目要使用 Spring Framework 才行。
　　这里简单记录下整合的过程：
　　如果项目之前没有整合过 spring-data-redis 的话，这一步需要先做，在 maven 中添加这个依赖：
|
<code><``dependency``></code>
<code><``groupId``>org.springframework.data</code>
<code><``artifactId``>spring-data-redis</code>
<code><``version``>1.5.2.RELEASE</code>
``
<code><``dependency``></code>
<code><``groupId``>org.springframework.session</code>
<code><``artifactId``>spring-session</code>
<code><``version``>1.0.2.RELEASE</code>
``
|
　　再在 applicationContext.xml 中添加以下 bean，用于定义 redis 的连接池和初始化 redis 版操作类，自行替换其中的相关变量。
|
``
<code><``bean</code> <code>id``=``"jedisPoolConfig"</code> <code>class``=``"red s.clients.jedis.JedisPoolConfig"``></code>
``
<code><``bean</code> <code>id``=``"jedisConnectionFactory"</code>
<code>class``=``"org.springframework.data.redis.connection.jedis.JedisConnectionFactor "``></code>
<code><``property</code> <code>name``=``"hostName"</code> <code>value``=``" {redis.host}"</code> <code>/></code>
<code><``property</code> <code>name``=``"port"</code> <code>value``=``"${redis port}"</code> <code>/></code>
<code><``property</code> <code>name``=``"password"</code> <code>value``=``"${ edis.pass}"</code> <code>/></code>
<code><``property</code> <code>name``=``"timeout"</code> <code>value``=``"${r dis.timeout}"</code> <code>/></code>

<code><``property</code> <code>name``=``"poolConfig"</code> <code>ref``=``"jedsPoolConfig"</code> <code>/></code>
<code><``property</code> <code>name``=``"usePool"</code> <code>value``=``"tru"</code> <code>/></code>
``
<code><``bean</code> <code>id``=``"redisTemplate"</code> <code>class``=``"org.pringframework.data.redis.core.StringRedisTemplate"``></code>
<code><``property</code> <code>name``=``"connectionFactory"</code> <code>ref`=``"jedisConnectionFactory"</code> <code>/></code>
``
``
<code><``bean</code> <code>id``=``"redisHttpSessionConfiguration"</code>
<code>class``=``"org.springframework.session.data.redis.config.annotation.web.http.RediHttpSessionConfiguration"``></code>
<code><``property</code> <code>name``=``"maxInactiveIntervalInSeconds"</code> <code>value``=``"1800"</code> <code>/></code>
``
|
　　这里前面几个 bean 都是操作 redis 时候使用的，最后一个 bean 才是 spring-session 需用到的，其中的 id 可以不写或者保持不变，这也是一个约定优先配置的体现。这个 bean 中又会自产生多个 bean，用于相关操作，极大的简化了我们的配置项。其中有个比较重要的是 springSessionepositoryFilter，它将在下面的代理 filter 中被调用到。maxInactiveIntervalInSeconds 表示超时时，默认是 1800 秒。写上述配置的时候我个人习惯采用 xml 来定义，官方文档中有采用注解来声明一配置类。
　　然后是在 web.xml 中添加一个 session 代理 filter，通过这个 filter 来包装 Servlet 的 getSession()。需要注意的是这个 filter 需要放在所有 filter 链最前面。
|
``
<code><``filter``></code>
<code><``filter-name``>springSessionRepositoryFilter</code>
<code><``filter-class``>org.springframework.web.filter.DelegatingFilterProxy</code>
``
<code><``filter-mapping``></code>
<code><``filter-name``>springSessionRepositoryFilter</code>
<code><``url-pattern``>/*</code>
``
|
　　这样便配置完毕了，需要注意的是，spring-session 要求 Redis Server 版本不低于 2.8。
　　验证：使用 redis-cli 就可以查看到 session key 了，且浏览器 Cookie 中的 jsessionid 已替换为 session。
|
<code>127.0.0.1:6379> KEYS *</code>
<code>1) "spring:session:expirations:1440922740000"</code>
<code>2) "spring:session:sessions:35b48cb4-62f8-440c-afac-9c7e3cfe98d3"</code>

|