链滴

用 Spring Session on Redis 来管理 HttpSession

作者：tomaer

原文链接：https://ld246.com/article/1470663768474

来源网站：链滴

传统 HttpSession 有什么缺陷?

传统的 HttpSession 是存储在实现了 Servlet 标准的容器中的，其实现逻辑完全由容器自身完成，session 的整个生命周期，只谈产生和销毁，第一次发出请求由容器产生，过了有效时间或者是容器重启者关闭失效,表面上看这样视乎是没有问题的，但是随着用户数的增多，软件架构发生变化，这样的 session 方案视乎已经不能完全满足需求

- 用户数的增多会导致容器内部存储session数量增大，从而在 jvm 中产生大量的数据，GC 回收效降低
- 多 node 服务架构，粘性 session 问题，比如负载均衡之后节点（nginx反向代理，LVS负载等) sesson 共享

从上面的 2 个方面来看传统的 HttpSession 已经逐渐的不能胜任大型站点架构方案，我们需要找出的 session 方案。

Session 方案

- Tomcat Session on Redis or Memcached
- Jetty Session on Redis
- Spring Session

上面的几种方案，或者是 N 种方案，都是基于容器实现的，对容器的依赖度比较高。在之前的 XXX 件中曾经使用第一种方案，但是这种方案是完全针对某一特定容器，跨容器部署的时候就显得力不从，而且维护起来十分麻烦，曾把单 node 的 Redis 改到 redis-cluster 上用了很多时间，中间也出了多问题，所以不推荐针对某一容器的这种 session 方案。推荐 Spring Session 的原因是 Spring Session 是完全实现了 Servlet 标准的 Session 方案，并不关心到底是使用什么容器。

参考文档

- http://docs.spring.io/spring-session/docs/current/reference/html5/
- http://docs.spring.io/spring-session/docs/current/reference/html5/guides/httpsession-xml.tml
- http://redis.io/documentation

Minimum Requirements

- Java 5+
- If you are running in a Servlet Container (not required), Servlet 2.5+
- If you are using other Spring libraries (not required), the minimum required version is Sprin 3.2.14. While we re-run all unit tests against Spring 3.2.x, we recommend using the latest Sprng 4.x version when possible.
- @EnableRedisHttpSession requires Redis 2.8+. This is necessary to support Session Expirati n

Setup

● 添加Maven依赖

pom.xml

```
<dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session</artifactId>
    <version>1.2.1.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-data-redis</artifactId>
    <version>1.2.1.RELEASE</version>
</dependency>
```

● 添加 springSessionRepositoryFilter

由于spring-data-redis用到了spring框架,所以需要加载spring配置文件并且添加spring-web依赖

pom.xml

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>4.2.5.RELEASE</version>
</dependency>
```

web.xml

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring-session-redis-single.xml</param-value>
</context-param>
 <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
 </listener>
<!-- 注springSessionRepositoryFilter必须放在最前面 -->
<filter>
   <filter-name>springSessionRepositoryFilter</filter-name>
   <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
   <filter-name>springSessionRepositoryFilter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

● 编写 /WEB-INF/spring-session-redis-single.xml

下面是spring官方给出的示例配置,并没有发现从哪里配置Redis的相关信息

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2
01/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springfr
```

```
mework.org/schema/tx" xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/sch
ma/beans/spring-beans-4.2.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/sch
ma/context/spring-context-4.2.xsd
        http://www.springframework.org/schema/aop http://www.springframework.org/schem
/aop/spring-aop-4.2.xsd
        http://www.springframework.org/schema/tx http://www.springframework.org/schema/
x/spring-tx-4.2.xsd">
    <context:annotation-config/>
    <bean class="org.springframework.session.data.redis.config.annotation.web.http.RedisHtt
SessionConfiguration"/>
    <bean class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory"/>
</beans>
```

经过研究得到如下配置

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2
01/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springfr
mework.org/schema/tx" xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/sch
ma/beans/spring-beans-4.2.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/sch
ma/context/spring-context-4.2.xsd
        http://www.springframework.org/schema/aop http://www.springframework.org/schem
/aop/spring-aop-4.2.xsd
        http://www.springframework.org/schema/tx http://www.springframework.org/schema/
x/spring-tx-4.2.xsd">
    <context:annotation-config/>
    <bean id="jedisPoolConfig" class="redis.clients.jedis.JedisPoolConfig">
        <property name="maxTotal" value="30"/>
        <property name="maxIdle" value="10"/>
        <property name="minIdle" value="1"/>
        <property name="maxWaitMillis" value="30000"/>
        <property name="testOnBorrow" value="true"/>
        <property name="testOnReturn" value="false"/>
        <property name="testWhileIdle" value="false"/>
    </bean>
    <bean id="jedisConnectionFactory" class="org.springframework.data.redis.connection.jedis
JedisConnectionFactory">
    <property name="hostName" value="127.0.0.1" />
    <property name="port" value="6379" />
    <property name="password" value="" />
    <property name="timeout" value="3000" />
    <property name="poolConfig" ref="jedisPoolConfig" />
    <property name="usePool" value="true" />
    </bean>
    <bean id="redisTemplate" class="org.springframework.data.redis.core.RedisTemplate">
    <property name="connectionFactory" ref="jedisConnectionFactory"/>
```

```
    </bean>
    <bean class="org.springframework.session.data.redis.config.annotation.web.http.RedisHttSessionConfiguration">
    <property name="maxInactiveIntervalInSeconds" value="1800" />
    </bean>
</beans>
```

深入研究

- Session的跨域问题, cookiepath 和 domain
- 修改Session存在cookie中名字，怕与其他程序有影响
- org.springframework.session.web.http.SessionRepositoryFilter
- org.springframework.session.web.http.CookieHttpSessionStrategy
- org.springframework.session.web.http.DefaultCookieSerializer

翻阅spring-session的源代码,DefaultCookieSerializer的writeCookieValue方法,getCookiePath方法getDomainName方法可以得知spring-session中没有对跨域问题进行处理，但是他允许我们IOC属来修改实现逻辑

```
<bean class="org.springframework.session.web.http.DefaultCookieSerializer">
    <!-- 我去掉了这个地方的value值，请根据自己需要填写 -->
    <property name="cookiePath" value="" />
    <property name="domainName" value="" />
    <property name="cookieName" value="" />
</bean>
```

- 如何支持Redis Cluster (没有经过测试,由于Azure的Redis没有看到提供Redis Cluster的方式,顾暂不考虑,后期可做深入研究)
- org.springframework.data.redis.connection.jedis.JedisConnectionFactory
- org.springframework.data.redis.connection.RedisClusterConfiguration
- org.springframework.data.redis.connection.RedisNode

```
<bean id="clusterNodes1" class="org.springframework.data.redis.connection.RedisNode">
    <!-- 可以采用property给每个属性赋值,但是目前并不知道那个节点是Master Node,所以这个部分给Srping自己完成 -->
    <constructor-arg index="0" name="host" value="" />
    <constructor-arg index="1" name="port" value="" />
</bean>
<bean id="clusterConfig" class="org.springframework.data.redis.connection.RedisClusterConfguration">
    <property name="clusterNodes">
        <set>
            <ref bean="clusterNodes1" />
            ...
        </set>
    </property>
</bean>
<bean id="jedisConnectionFactory" class="org.springframework.data.redis.connection.jedis.JdisConnectionFactory">
    <property name="timeout" value="300" />
```

```
    <property name="poolConfig" ref="jedisPoolConfig" />
    <property name="password" value="" />
    <property name="usePool" value="true" />
    <property name="clusterConfig" ref="clusterConfig" />
</bean>
```

- Spring Session 支持其他的存储方案
- MongoDB
- JDBC

Testing Spring Session on Redis

下面的内容更新于2016年10月26日 早9时,注意配置文件里面是用的单实例的Redis,如果用集成环境的edis,自己稍加改造就出来了,配置文件上面已经给出了,需要注意的东西我会放在打赏区

- 下载源代码 http://7bvamo.com1.z0.glb.clouddn.com/spring-session.zip (源代码已经更新)
- 修改/spring-session/src/main/resources/META-INF/configuration/environments/application properties 中对于redis的配置和相关的cookie配置
- mvn clean package
- 启动2个不同端口的容器
- 同一个浏览器访问不同端口的 /login 看看什么情况,注意观察redis中的变化和浏览器cookie部分
- 换一个浏览器访问任意一个端口的/login 再看看redis中的变化
- 删除掉redis中的所有数据, 改变之前访问过的浏览器地址为/login再看看什么情况