# 孙猴子

生如夏花





博客园 首页 新随笔 联系 管理 订阅 🞹

随笔-119 文章-0 评论-17

### 使用Redis做MyBatis的二级缓存

#### 1. 介绍

使用mybatis时可以使用二级缓存提高查询速度,进而改善用户体验。

使用redis做mybatis的二级缓存可是内存可控<如将单独的服务器部署出来用于二级缓存>,管理方便。

#### 2. 使用思路

- 2.1 配置redis.xml 设置redis服务连接各参数
- 2.1 在配置文件中使用 <setting> 标签,设置开启二级缓存;
- 2.2 在mapper.xml 中使用<cache type="com.demo.RedisCacheClass" /> 将cache映射到指定的 RedisCacheClass类中;
  - 2.3 映射类RedisCacheClass 实现 MyBatis包中的Cache类,并重写其中各方法;

在重写各方法体中,使用redisFactory和redis服务建立连接,将缓存的数据加载到指定的redis内存中(putObject方法)或将redis服务中的数据从缓存中读取出来(getObject方法);

在redis服务中写入和加载数据时需要借用spring-data-redis.jar中JdkSerializationRedisSerializer.class中的序列化(serialize)和反序列化方法(deserialize),此为包中封装的redis默认的序列化方法;

2.4 映射类中的各方法重写完成后即可实现mybatis数据二级缓存到redis服务中;

#### 3. 代码实践

3.1 配置redis.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"</pre>
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:p="http://www.springframework.org/schema/p"
   xmlns:mvc="http://www.springframework.org/schema/mvc"
   xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
   xmlns:context="http://www.springframework.org/schema/context"
   xmlns:task="http://www.springframework.org/schema/task"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
   http://www.springframework.org/schema/mvc
   http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd
   http://www.springframework.org/schema/aop
   http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
   http://www.springframework.org/schema/tx
   http://www.springframework.org/schema/tx/spring-tx.xsd
   http://www.springframework.org/schema/context
   http://www.springframework.org/schema/context/spring-context-3.0.xsd
   http://www.springframework.org/schema/task
   http://www.springframework.org/schema/task/spring-task-3.0.xsd" >
   <!-- enable autowire -->
   <context:annotation-config />
   <task:annotation-driven/>
   <context:component-scan base-package="demo.util,demo.salesorder,demo.person" />
    <!-- Configures the @Controller programming model 必须加上这个,不然请求controller时会出现
no mapping url错误-->
   <mvc:annotation-driven />
```

昵称: 孙猴子 园龄: 3年 粉丝: 21 关注: 3 +加关注

5 30 8 9 10 11 12 6 17 15 16 19 20 21 22 23 26 28 29 30 31 2 6 8 9

2019年1月

# \_\_ 搜索

找找看

### 学用链接

我的随笔

我的评论

我的参与 最新评论

我的标签

# 主我的标签

**Java**(13)

MySQL(12)

Spring(10)

SQL(9)

redis(9)

WEB(9)

JavaScript(9)

WebService(7)

mongoDB(6)

eclipse(6) 更多

# 随笔分类

ActiveMQ(3)

Apache(1)

CSS(3)

git(2)

HTML(1)

Java(18)

JavaScript(12)

JSON(1)

JSP(2)

Junit(1)

Linux(6) log4j(4)

Maven(4)

mongoDB(5)

mybatis(2)

```
<!-- 引入数据库配置文件 -->
   <bean id="propertyConfigurer"</pre>
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
       property name="locations">
           st>
              <value>classpath:sysconfig/jdbc.properties</value>
              <value>classpath:sysconfig/redis.properties</value>
           </list>
       </property>
   </bean>
   <!-- JDBC -->
   destroy-method="close"
       p:driverClassName="${jdbc.driverClassName}"
       p:url="${jdbc.databaseurl}"
       p:username="${jdbc.username}"
       p:password="${jdbc.password}" >
       cproperty name="maxActive">
          <value>${jdbc.maxActive}</value>
       </property>
       property name="initialSize">
          <value>${jdbc.initialSize}</value>
       </property>
       property name="maxWait">
          <value>${jdbc.maxWait}</value>
       </property>
       property name="maxIdle">
           <value>${jdbc.maxIdle}</value>
       </property>
       property name="minIdle">
          <value>${jdbc.minIdle}</value>
       </property>
       <!-- 只要下面两个参数设置成小于8小时(MySql默认),就能避免MySql的8小时自动断开连接问题 -->
       property name="timeBetweenEvictionRunsMillis">
          <value>18000000
       /property><!-- 5小时 -->
       roperty name="minEvictableIdleTimeMillis">
           <value>10800000
       /property><!-- 3小时 -->
       property name="validationQuery">
          <value>SELECT 1</value>
       </property>
       property name="testOnBorrow">
          <value>true</value>
       </property>
   </bean>
   <!-- define the SqlSessionFactory -->
   <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
       roperty name="dataSource" ref="defaultDataSource" />
       <!-- 可以单独指定mybatis的配置文件,或者写在本文件里面。 用下面的自动扫描装配(推荐)或者单独
mapper -->
       <property name="configLocation" value="classpath:sysconfig/mybatis-config.xml" />
   </bean>
   <!-- 自动扫描并组装MyBatis的映射文件和接口-->
   <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
       cproperty name="basePackage" value="demo.*.data" />
       <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"></property>
   </bean>
   <!-- JDBC END -->
   <!-- redis数据源 -->
   <bean id="poolConfig" class="redis.clients.jedis.JedisPoolConfig">
       cproperty name="maxIdle" value="${redis.maxIdle}" />
       cyroperty name="maxTotal" value="${redis.maxActive}" />
       cproperty name="maxWaitMillis" value="${redis.maxWait}" />
       cyroperty name="testOnBorrow" value="${redis.testOnBorrow}" />
   </bean>
   <!-- Spring-redis连接池管理工厂 -->
   <bean id="jedisConnectionFactory"</pre>
class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory">
```

```
MySQL(12)
Network(1)
redis(9)
Spring(10)
SOL(9)
Tomcat(2)
UML(1)
WEB(11)
WebService(7)
Windwos(2)
数据库(2)
算法(1)
2018年2月 (2)
2018年1月 (2)
2017年10月 (1)
2017年7月 (3)
2017年4月 (2)
2017年3月 (4)
2017年2月 (21)
2017年1月 (21)
2016年12月 (22)
2016年11月 (13)
2016年10月 (8)
2016年9月 (3)
2016年8月 (5)
2016年7月 (4)
2016年6月 (8)
```

## ➡ 最新评论

1. Re:Jquery 移除 html中绑定的onClick事件

解决了我的问题,谢谢!

--台北的冬天看雨

2. Re:redis 安装启动及设置密码<windows> 没有效果哇

--bihang

3. Re:web过滤器中获取请求的参数(content -type:multipart/form-data)

HttpServletRequest转化成 MultipartHttpServletRequest数据丢失了,不知道你遇没遇到这么情况!目前不知道什么原因。

--不会写java

4. Re:web过滤器中获取请求的参数(content -type:multipart/form-data)

终于解决了问题,postman访问接口的时候 过滤XXS攻击的时候没过滤掉,后来才知道 是这么个问题。

--大楠子

5. Re:Jquery 移除 html中绑定的onClick事件

怎么恢复

--liqinglin

# **阅读排行榜**

- 1. JS trim()方法使用(129448)
- 2. redis 安装启动及设置密码<windows>(75 201)
- 3. web过滤器中获取请求的参数(content-ty pe:multipart/form-data)(31571)
- 4. 使用Redis做MyBatis的二级缓存(24401)
- 5. ActiveMQ (一): 安装启动及测试(23401)

# • 评论排行榜

- 1. Jquery 移除 html中绑定的onClick事件(7)
- 2. web过滤器中获取请求的参数(content-ty pe:multipart/form-data)(3)
- 3. ActiveMQ (一): 安装启动及测试(2)
- 4. redis 安装启动及设置密码<windows>(2)

### 3.2 mybatis.xml 配置开启二级缓存

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
   PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
   "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
   <!-- 配置mybatis的缓存,延迟加载等等一系列属性 -->
   <settings>
       <!-- 全局映射器启用缓存 *主要将此属性设置完成即可-->
       <setting name="cacheEnabled" value="true"/>
       <!-- 查询时,关闭关联对象即时加载以提高性能 -->
       <setting name="lazyLoadingEnabled" value="false"/>
       <!-- 对于未知的SQL查询,允许返回不同的结果集以达到通用的效果 -->
       <setting name="multipleResultSetsEnabled" value="true"/>
       <!-- 设置关联对象加载的形态,此处为按需加载字段(加载字段由SQL指定),不会加载关联表的所有字段,
以提高性能 -->
       <setting name="aggressiveLazyLoading" value="true"/>
   </settings>
</configuration>
```

### 3.3 在mapper.xml中映射缓存类RedisCacheClass

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="demo.person.data.UserMapper">
<cache type="demo.redis.cache.RedisCache"/> <!-- *映射语句 -->
<select id="getPersonList" parameterType="map" resultType="Person">
    select '
     from person
    <where>
       1=1
        <if test="user_name!=null">
            and user_name=#{user_name}
        </if>
    </where>
</select>
<insert id="addPerson" parameterType="Person" keyProperty="id" useGeneratedKeys="true">
    insert into person(
       login_id,
        user_name,
        gender,
        birthday,
        remark
```

5. 使用Redis做MyBatis的二级缓存(1)

## **业**推荐排行榜

- 1. Jquery 移除 html中绑定的onClick事件(3)
- 2. Spring 框架的设计理念与设计模式分析 (3)
- 3. redis 安装启动及设置密码<windows>(3)
- 4. ActiveMQ (一): 安装启动及测试(1)
- 5. JavaWeb项目的classpath说明(1)

```
)values(
    #{login_id},
    #{user_name},
    #{gender},
    #{birthday},
    #{remark}
    )
</insert>
```

#### 3.4 实现Mybatis中的Cache接口

Cache.class源码:

```
/*

* Copyright 2009-2012 the original author or authors.

* http://www.apache.org/licenses/LICENSE-2.0*/
package org.apache.ibatis.cache;
import java.util.concurrent.locks.ReadWriteLock;

public interface Cache {

String getId();
int getSize();

void putObject(Object key, Object value);

Object getObject(Object key);

Object removeObject(Object key);

void clear();

ReadWriteLock getReadWriteLock();

}
```

### RedisCache.java

```
package demo.redis.cache;
import java.util.concurrent.locks.ReadWriteLock;
import java.util.concurrent.locks.ReentrantReadWriteLock;
import org.apache.ibatis.cache.Cache;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.redis.connection.jedis.JedisConnection;
import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;
{\color{blue} \textbf{import} org.spring framework.data.red is.serializer.Jdk Serialization Red is Serializer;}
{\color{red} \underline{\mathsf{import}}} \ \ \mathsf{org.springframework.data.red} \\ \mathsf{is.serializer.Red} \\ \mathsf{is.Serializer};
{\color{red} \textbf{import}} \ \ \textbf{redis.clients.jedis.exceptions.JedisConnectionException;}
public class RedisCache implements Cache //实现类
    private static final Logger logger = LoggerFactory.getLogger(RedisCache.class);
    private static JedisConnectionFactory jedisConnectionFactory;
    private final String id;
```

```
* The {@code ReadWriteLock}.
   private final ReadWriteLock readWriteLock = new ReentrantReadWriteLock();
   public RedisCache(final String id) {
       if (id == null) {
           throw new IllegalArgumentException("Cache instances require an ID");
       logger.debug("MybatisRedisCache:id=" + id);
       this.id = id;
   }
   @Override
   public void clear()
       JedisConnection connection = null;
       try
           connection = jedisConnectionFactory.getConnection(); //连接清除数据
           connection.flushDb();
           connection.flushAll();
       catch (JedisConnectionException e)
           e.printStackTrace();
       finally
       {
           if (connection != null) {
               connection.close();
       }
   }
   @Override
   public String getId()
       return this.id;
   @Override
   public Object getObject(Object key)
       Object result = null;
       JedisConnection connection = null;
       try
           connection = jedisConnectionFactory.getConnection();
           RedisSerializer<Object> serializer = new JdkSerializationRedisSerializer();
//借用spring_data_redis.jar中的JdkSerializationRedisSerializer.class
           result = serializer.deserialize(connection.get(serializer.serialize(key)));
//利用其反序列化方法获取值
       catch (JedisConnectionException e)
       {
           e.printStackTrace();
       finally
           if (connection != null) {
               connection.close();
       return result;
   }
   @Override
   public ReadWriteLock getReadWriteLock()
       return this.readWriteLock;
   @Override
   public int getSize()
       int result = 0;
```

```
JedisConnection connection = null;
        try
        {
            connection = jedisConnectionFactory.getConnection();
            result = Integer.valueOf(connection.dbSize().toString());
        catch (JedisConnectionException e)
            e.printStackTrace();
        }
        finally
        {
            if (connection != null) {
               connection.close();
        return result;
    }
    @Override
    public void putObject(Object key, Object value)
        JedisConnection connection = null;
        try
        {
            logger.info(">>>>>>>>>>>>putObject:"+key+"="+value);
            connection = jedisConnectionFactory.getConnection();
            RedisSerializer<Object> serializer = new JdkSerializationRedisSerializer();
//借用spring_data_redis.jar中的JdkSerializationRedisSerializer.class
            connection.set(serializer.serialize(key), serializer.serialize(value)); //利用
其序列化方法将数据写入redis服务的缓存中
        catch (JedisConnectionException e)
            e.printStackTrace();
        finally
            if (connection != null) {
                connection.close();
        }
    }
    @Override
    public Object removeObject(Object key)
        JedisConnection connection = null
        Object result = null;
        try
            connection = jedisConnectionFactory.getConnection();
            RedisSerializer<Object> serializer = new JdkSerializationRedisSerializer();
            result =connection.expire(serializer.serialize(key), 0);
        catch (JedisConnectionException e)
            e.printStackTrace();
        }
        finally
            if (connection != null) {
                connection.close();
        return result;
    public static void setJedisConnectionFactory(JedisConnectionFactory
jedisConnectionFactory) {
        RedisCache.jedisConnectionFactory = jedisConnectionFactory;
    }
}
```

### 4. 总结

通过重写Cache类中的方法,将mybatis中默认的缓存空间映射到redis空间中。

#### 分类: mybatis,redis

标签: redis, mybatis













<u>孙猴子</u> <u> 关注 - 3</u> 粉丝 - 21

«上一篇: MySQL 类型转换

€推荐



» 下一篇: mysql 基础命令

posted @ 2017-02-07 15:32 孙猴子 阅读(24401) 评论(1) 编辑 收藏

#### 评论

#1楼 2017-08-24 11:13 | 梁沛健 🖂

66

redis.xml 配置当中最底下的这句:

<bean class="demo.util.UTF8StringBeanPostProcessor"></bean> 哪蹦出来的,有什么用,有这个文件的源码吗兄弟?

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

🤜 注册用户登录后才能发表评论,请 <u>登录</u> 或 <u>注册</u>,<u>访问</u>网站首页。

【推荐】超50万VC++源码: 大型组态工控、电力仿真CAD与GIS源码库!



### 相关博文:

- · 使用redis做mybaties的二级缓存(2)-Mybatis 二级缓存小心使用
- ·Mybatis使用Redis二级缓存
- ·Redis集成到Spring做mybatis做二级缓存
- redis作为mybatis的二级缓存
- · Spring + MySQL + Mybatis + Redis【二级缓存】



#### 最新新闻:

- ·应用商店8大趋势: 消费支出高速增长 苹果商店支出远超谷歌
- ·马斯克回应澳洲官员:澳洲山脉下挖50公里隧道耗资近10亿美元
- · 小偷瞄准特斯拉汽车 旧金山湾区过去几周发生数十起破窗盗窃案
- · 蛋壳公寓2亿美元全资收购爱上租 为长租公寓行业最大收购案
- ·顺丰回应"高负债问题":直营模式财务数据与轻资产公司不可比
- » 更多新闻...