L_O_V_E_Java

请坚持，请珍惜

博客园 闪存 首页 新随笔 联系 管理 订阅

随笔- 35 文章- 0 评论- 5

ssm+redis整合(通过cache方式)

这几天的研究ssm redis 终于进入主题了，今天参考了网上一些文章搭建了一下ssm+redis整合，特别记录下来以便以后可以查询使用，有什么不足请大牛们提点

项目架构

昵称：L_O_V_E_Java
园龄：4年2个月
粉丝：5
关注：0
+加关注

< 2018年5月 >

日	一	二	三	四	五	六
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

我的标签

android(9)
aop(4)
JAVA(4)
redis(3)
ssm+redis整合(3)
ssm整合(2)
webservice(1)
组件(1)
activiti(1)
js(1)
更多

随笔档案

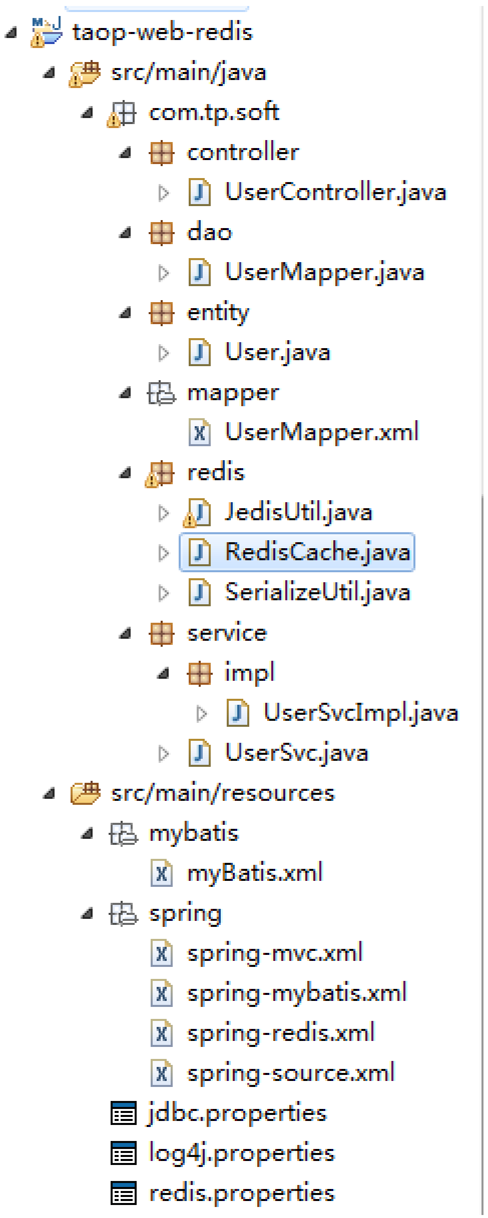
2017年9月 (1)
2017年8月 (1)
2017年7月 (13)
2016年5月 (4)
2014年12月 (1)
2014年9月 (5)
2014年8月 (8)
2014年6月 (2)

最新评论

1. Re:ssm+redis整合(通过cache方式)
问题是
你这张表的增加删除修改　要如何清空相关联的别的缓存?
功能实现倒不难　可以一一进行配置　但是不知道有没有可以懒省事的一套规则　可以让程序员无需关心这一点。
--辉辉?

2. Re:ssm+redis整合(通过cache方式)
@deel_feelssm搭建完成后验证可以查到User信息么?...
--习惯沉淀

3. Re:ssm+redis整合(通过cache方式)
望解答：项目按照你的搭建好，运行没有报错，而查询出来的user对象为空。请问是什



```
taop-web-redis
  src/main/java
    com.tp.soft
      controller
        UserController.java
      dao
        UserMapper.java
      entity
        User.java
      mapper
        UserMapper.xml
      redis
        JedisUtil.java
        RedisCache.java
        SerializeUtil.java
      service
        impl
          UserSvcImpl.java
        UserSvc.java
  src/main/resources
    mybatis
      myBatis.xml
    spring
      spring-mvc.xml
      spring-mybatis.xml
      spring-redis.xml
      spring-source.xml
    jdbc.properties
    log4j.properties
    redis.properties
```

1、pom.xml

么回事，没有报错，又应该去哪里找问题呢？我qq 836543420

--deel_feel

4. Re:常用webservice接口

nice!!!

--青菜L萝卜

5. Re:PHP5.5.13 + Apache2.4.7安装配置流程详解

想要jar的我可以邮箱发给你们哈

--L_O_V_E_Java

阅读排行榜

1. ssm+redis整合(通过cache方式)(4649)
2. JAVA的UDP协议交互信息(860)
3. android中的HttpURLConnection和HttpClient实现app与pc数据交互(471)
4. ssm+redis整合(通过aop自定义注解方式)(402)
5. android线程登录(395)

评论排行榜

1. ssm+redis整合(通过cache方式)(3)
2. PHP5.5.13 + Apache2.4.7安装配置流程详解(1)
3. 常用webservice接口(1)

推荐排行榜

1. js处理iframe中子页面与父页面里面对象的取得的解决方案(1)
2. xml资源getStringArray(R.array.xxx)方法(1)
3. 常用webservice接口(1)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.tp.soft</groupId>
  <artifactId>taop-web-redis</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>taop-web-redis Maven Webapp</name>
  <url>http://maven.apache.org</url>
 <properties>
        <!-- spring版本号 -->
        <spring.version>4.3.2.RELEASE</spring.version>
        <!-- mybatis版本号 -->
        <mybatis.version>3.2.8</mybatis.version>
        <!-- log4j日志文件管理包版本 -->
        <slf4j.version>1.7.7</slf4j.version>
        <log4j.version>1.2.17</log4j.version>
        <oracle.version>10.2.0.5.0</oracle.version>
    </properties>

     <dependencies>
        <!-- 添加Spring-core包 -->
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <!-- 添加spring-context包 -->
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
```

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${spring.version}</version>
</dependency>
<!-- 添加spring-tx包 -->
<!-- https://mvnrepository.com/artifact/org.springframework/spring-tx -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
    <version>${spring.version}</version>
</dependency>
<!-- 添加spring-jdbc包 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${spring.version}</version>
</dependency>
<!-- 为了方便进行单元测试，添加spring-test包 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>${spring.version}</version>
</dependency>
 <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aop</artifactId>
    <version>${spring.version}</version>
</dependency>
```

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-expression</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-beans</artifactId>
    <version>${spring.version}</version>
</dependency>
   <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>${spring.version}</version>
</dependency>
<!--添加aspectjweaver包 -->
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.8.9</version>
</dependency>
<!-- 添加mybatis的核心包 -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>${mybatis.version}</version>
</dependency>
<!-- 添加mybatis与Spring整合的核心包 -->
<dependency>
```

```
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>1.3.0</version>
        </dependency>
        <!-- 添加servlet3.0核心包 -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>javax.servlet.jsp-api</artifactId>
            <version>2.3.2-b01</version>
        </dependency>
        <!-- jstl -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>
        <!-- 添加druid连接池包 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>1.0.24</version>
        </dependency>
        <!-- 添加junit单元测试包 -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
```

```
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>${log4j.version}</version>
</dependency>

<dependency>
    <groupId>com.oracle</groupId>
    <artifactId>ojdbc14</artifactId>
    <version>${oracle.version}</version>
</dependency>
<dependency>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>1.2</version>
</dependency>
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.3</version>
</dependency>

<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>2.9.0</version>
</dependency>

<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-redis</artifactId>
    <version>1.6.2.RELEASE</version>
</dependency>
```

```
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-ehcache</artifactId>
            <version>1.0.0</version>
        </dependency>
     </dependencies>
  <build>
    <finalName>maven-tp-web</finalName>
  </build>
</project>
```

2、jdbc.properties

```
#JDBCdriver
jdbc.driverClassName=oracle.jdbc.driver.OracleDriver

#JDBC
jdbc.url=你的连接服务器
jdbc.username=数据库账号
jdbc.password=数据库密码
jdbc.publickey=阿里秘钥



jdbc.dbcp.initialSize=10
jdbc.dbcp.maxActive=500
jdbc.dbcp.maxIdle=10
jdbc.dbcp.minIdle=1
jdbc.dbcp.maxWait=120000
```

3、log4j.properties 可以使用mybatis自带

4、redis.properties

```
redis.host = 192.168.76.76
redis.port = 6379
redis.pass = admin
redis.maxIdle = 200
redis.maxActive = 1024
redis.maxWait = 10000
redis.testOnBorrow = true
```

5、spring-source.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-4.3.xsd">

    <!-- 引入dbconfig.properties属性文件 -->
    <context:property-placeholder location="classpath:*.properties" />

    <!-- 自动扫描(自动注入), 扫描me.gacl.service这个包以及它的子包的所有使用@Service注解标注的类 -->
    <context:component-scan base-package="com.tp.soft" />
</beans>
```

6、spring-redis.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
  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:util="http://www.springframework.org/schema/util"
  xmlns:aop="http://www.springframework.org/schema/aop"
  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-4.3.xsd
      http://www.springframework.org/schema/util
      http://www.springframework.org/schema/util/spring-util-4.3.xsd
      http://www.springframework.org/schema/mvc
      http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
      http://www.springframework.org/schema/aop
      http://www.springframework.org/schema/aop/spring-aop-4.3.xsd
      http://www.springframework.org/schema/context
      http://www.springframework.org/schema/context/spring-context-4.3.xsd">


    <!-- 连接池基本参数配置，类似数据库连接池 -->
     <context:property-placeholder location="classpath*:redis.properties" />

    <bean id="poolConfig" class="redis.clients.jedis.JedisPoolConfig">
        <property name="maxIdle" value="${redis.maxIdle}" />
        <property name="testOnBorrow" value="${redis.testOnBorrow}"/>
    </bean>

    <!-- 连接池配置，类似数据库连接池 -->
    <bean id="connectionFactory"
class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory" >
        <property name="hostName" value="${redis.host}"></property>
        <property name="port" value="${redis.port}"></property>
        <property name="password" value="${redis.pass}"></property>
        <property name="poolConfig"  ref="poolConfig"></property>
    </bean>

    <!-- 调用连接池工厂配置 -->
    <bean id="redisTemplate" class=" org.springframework.data.redis.core.RedisTemplate">
        <property name="connectionFactory" ref="connectionFactory"></property>

       <!--  如果不配置Serializer，那么存储的时候智能使用String，如果用User类型存储，那么会提示错误
User can't cast
        to String！！！   -->
         <property name="keySerializer">
            <bean
            class="org.springframework.data.redis.serializer.StringRedisSerializer" />
        </property>
        <property name="valueSerializer">
            <bean
```

```
class="org.springframework.data.redis.serializer.JdkSerializationRedisSerializer" />
        </property>
    </bean>
</beans>
```

7、spring-mybatis.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.3.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">


    <!--Druid 连接池配置-->
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init"
destroy-method="close">
        <!-- 基本属性 url、user、password -->
        <property name="url" value="${jdbc.url}" />
        <property name="username" value="${jdbc.username}" />
        <property name="password" value="${jdbc.password}" />
        <property name="filters" value="stat,config" />
         <property name="connectionProperties"
```

```
value="config.decrypt=true;config.decrypt.key=${jdbc.publickey}" />

        <!-- 配置初始化大小、最小、最大 -->
        <property name="initialSize" value="1" />
        <property name="minIdle" value="1" />
        <property name="maxActive" value="40" />

        <!-- 配置获取连接等待超时的时间 -->
        <property name="maxWait" value="60000" />

        <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->
        <property name="timeBetweenEvictionRunsMillis" value="60000" />

        <!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->
        <property name="minEvictableIdleTimeMillis" value="300000" />

        <property name="validationQuery" value="SELECT 'x' FROM DUAL" />
        <property name="testWhileIdle" value="true" />
        <property name="testOnBorrow" value="false" />
        <property name="testOnReturn" value="false" />

        <!-- 打开PSCache，并且指定每个连接上PSCache的大小 -->
        <property name="poolPreparedStatements" value="true" />
        <property name="maxPoolPreparedStatementPerConnectionSize" value="20" />

        <!-- 超过时间限制是否回收 -->
        <property name="removeAbandoned" value="true" />
        <!-- 超时时间；单位为秒。180秒=3分钟 -->
        <property name="removeAbandonedTimeout" value="180" />
        <!-- 关闭abanded连接时输出错误日志 -->
        <property name="logAbandoned" value="true" />

        <!-- 配置监控统计拦截的filters，去掉后监控界面sql无法统计 -->
        <!-- property name="filters" value="stat" /-->
```

```
    </bean>


    <!-- 配置sqlSessionFactory -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 实例化sqlSessionFactory时需要使用上述配置好的数据源以及SQL映射文件 -->
        <property name="dataSource" ref="dataSource" />
        <!-- 自动扫描me/gacl/mapping/目录下的所有SQL映射的xml文件, 省掉Configuration.xml里的手工配置 value="classpath:com/mapper/*.xml"指的是classpath(类路径)下com.mapping包中的所有xml文件
            UserMapper.xml位于com.mapping包下，这样UserMapper.xml就可以被自动扫描 -->
         <property name="mapperLocations" value="classpath:com/tp/soft/mapper/*.xml" />
         <property name="configLocation" value="classpath:mybatis/myBatis.xml"/>
    </bean>

    <!-- 配置扫描器 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- 扫描com.dao这个包以及它的子包下的所有映射接口类 -->
        <property name="basePackage" value="com.tp.soft.dao" />
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />
    </bean>
</beans>
```

8、myBatis.xml

```
<?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>
    <settings>
        <!-- 日志开启 -->
        <setting name="logImpl" value="LOG4J"/>
        <!-- 二级缓存开启 -->
        <setting name="cacheEnabled" value="true"/>
        <setting name="lazyLoadingEnabled" value="false"/>
        <setting name="aggressiveLazyLoading" value="true"/>
    </settings>
    <!-- 配置映射类的别名 -->

    <typeAliases>
        <!-- 配置entity下的所有别名 别名首字母小写 -->
        <package name="com.tp.soft.entity" />
    </typeAliases>
</configuration>
```

9、User.java

```
package com.tp.soft.entity;

import java.io.Serializable;

public class User implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = -1695973853274402680L;

    private int userid;

    private String login_name;
    private String login_pwd;


    public User() {

    }


    public User(int userid, String login_name, String login_pwd) {
        super();
        this.userid = userid;
        this.login_name = login_name;
        this.login_pwd = login_pwd;
    }


    public int getUserid() {
        return userid;
    }


    public void setUserid(int userid) {
        this.userid = userid;
    }


    public String getLogin_name() {
        return login_name;
    }


    public void setLogin_name(String login_name) {
        this.login_name = login_name;
    }


    public String getLogin_pwd() {
        return login_pwd;
    }


    public void setLogin_pwd(String login_pwd) {
        this.login_pwd = login_pwd;
    }


}
```

10、几个工具类

redisUtil.java 连接池类

```
package com.tp.soft.redis;

import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisPool;
import redis.clients.jedis.JedisPoolConfig;

public class JedisUtil {
    private static String ADDR = "192.168.76.76";
    private static int PORT = 6379;
    private static String AUTH = "admin";
    private static int MAX_ACTIVE = 1024;

    private static int MAX_IDLE = 200;

    private static int MAX_WAIT = 10000;

    private static int TIMEOUT = 10000;

    private static boolean TEST_ON_BORROW = true;

    private static JedisPool jedisPool = null;

    static {
        try{
            JedisPoolConfig config = new JedisPoolConfig();
            config.setMaxIdle(MAX_IDLE);
            config.setMaxWaitMillis(MAX_WAIT);
            config.setTestOnBorrow(TEST_ON_BORROW);
            jedisPool = new JedisPool(config,ADDR,PORT,TIMEOUT,AUTH);
        }catch (Exception e) {
            e.printStackTrace();
        }
    }

    public synchronized static Jedis getJedis(){
        try{
            if(jedisPool != null){
                Jedis jedis = jedisPool.getResource();
                return jedis;
            }else{
                return null;
            }
        }catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    public static void returnResource(final Jedis jedis){
        if(jedis != null){
            jedisPool.returnResource(jedis);
        }
    }
}
```

SerializeUtil.java 序列化类

```
package com.tp.soft.redis;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class SerializeUtil {
    public static byte[] serialize(Object object) {
        ObjectOutputStream oos = null;
        ByteArrayOutputStream baos = null;
        try {
            // 序列化
            baos = new ByteArrayOutputStream();
            oos = new ObjectOutputStream(baos);
            oos.writeObject(object);
            byte[] bytes = baos.toByteArray();
            return bytes;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    public static Object unserialize(byte[] bytes) {
        if (bytes == null)
            return null;
        ByteArrayInputStream bais = null;
        try {
            // 反序列化
            bais = new ByteArrayInputStream(bytes);
            ObjectInputStream ois = new ObjectInputStream(bais);
            return ois.readObject();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

RedisCache.java 缓存类

```
package com.tp.soft.redis;

import java.util.concurrent.locks.ReadWriteLock;
import java.util.concurrent.locks.ReentrantReadWriteLock;
import org.apache.ibatis.cache.Cache;

/*
 * 使用第三方缓存服务器，处理二级缓存
 */
public class RedisCache implements Cache {
    private final ReadWriteLock readWriteLock = new ReentrantReadWriteLock();

    private String id;

    public RedisCache(final String id) {
        if (id == null) {
            throw new IllegalArgumentException("Cache instances require an ID");
        }
        this.id = id;

    }

    public String getId() {
        return this.id;
    }

    public void putObject(Object key, Object value) {
        JedisUtil.getJedis().set(SerializeUtil.serialize(key.toString()),
                SerializeUtil.serialize(value));

    }

    public Object getObject(Object key) {
        Object value = SerializeUtil.unserialize(JedisUtil.getJedis().get(
                SerializeUtil.serialize(key.toString())));
        return value;
    }

    public Object removeObject(Object key) {
        return JedisUtil.getJedis().expire(
                SerializeUtil.serialize(key.toString()), 0);

    }

    public void clear() {
        JedisUtil.getJedis().flushDB();
    }

    public int getSize() {
        return Integer.valueOf(JedisUtil.getJedis().dbSize().toString());
    }

    public ReadWriteLock getReadWriteLock() {
        return readWriteLock;
    }

}
```

11、UserMapper.xml

如果设置useCache="false" 则不进入二级redis缓存

```
<?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="com.tp.soft.dao.UserMapper">
```

```
    <!-- 缓存类配置 -->
    <cache type="com.tp.soft.redis.RedisCache" />

    <select id="getUserById" parameterType="int" resultType="user" useCache="true">
        select * from AU_USER where userid = #{id}
    </select>
</mapper>
```

12、UserMapper.java

```
package com.tp.soft.dao;

import com.tp.soft.entity.User;

public interface UserMapper {
    public User getUserById(int id);
}
```

13、UserSvc.java

```
package com.tp.soft.service;

import com.tp.soft.entity.User;

public interface UserSvc {
    public User getUser(int id);
}
```

14、UserSvcImpl.java

```
package com.tp.soft.service.impl;

import javax.annotation.Resource;

import org.springframework.dao.DataAccessException;
import org.springframework.stereotype.Service;

import com.tp.soft.dao.UserMapper;
import com.tp.soft.entity.User;
import com.tp.soft.service.UserSvc;

@Service("userService")
public class UserSvcImpl implements UserSvc{

    @Resource
    private UserMapper userMapper;

    public User getUser(int id) {
        User user = null;
        try{
            user = userMapper.getUserById(id);
        }catch (DataAccessException e) {
            System.out.println(e.getLocalizedMessage());
        }
        return user;
    }


}
```

15、配置springmvc

```
<beans xmlns="http://www.springframework.org/schema/beans"
  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:util="http://www.springframework.org/schema/util"
  xmlns:aop="http://www.springframework.org/schema/aop"
  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-4.3.xsd
      http://www.springframework.org/schema/util
      http://www.springframework.org/schema/util/spring-util-4.3.xsd
      http://www.springframework.org/schema/mvc
      http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
      http://www.springframework.org/schema/aop
      http://www.springframework.org/schema/aop/spring-aop-4.3.xsd
      http://www.springframework.org/schema/context
      http://www.springframework.org/schema/context/spring-context-4.3.xsd">

<!-- 设置注解扫描包 只扫描@Controller注解 -->
    <context:component-scan base-package="com.tp.soft" use-default-filters="false">
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Controller" />
    </context:component-scan>
```

```
      <mvc:annotation-driven />
    <!-- 静态资源 -->
    <mvc:default-servlet-handler/>

    <!-- 默认视图解析器 -->
    <bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver" p:order="3">
        <property name="suffix" value=".jsp" />
        <property name="prefix" value="/WEB-INF/pages/"></property>
    </bean>

    <!-- SpringMVC上传文件时,需配置MultipartResolver处理器 -->
    <bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver"
p:defaultEncoding="UTF-8">
        <!-- 指定所上传文件的总大小不能超过104857600......注意maxUploadSize属性的限制不是针对单个文
件,而是所有文件的容量之和 -->
        <property name="maxUploadSize" value="104857600" />
        <property name="maxInMemorySize" value="4096"/>
    </bean>
</beans>
```

16、配置web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xmlns:jsp="http://java.sun.com/xml/ns/javaee/jsp"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
```

```
<display-name>ssm整合(maven)</display-name>


<!-- spring配置文件整合配置 -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:spring/spring-source.xml, classpath:spring/spring-mybatis.xml</param-value>
  </context-param>

<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<!-- springmvc配置 -->
<servlet>
    <servlet-name>spring</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:spring/spring-mvc.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>spring</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

<!-- druid 监控 -->
<servlet>
     <servlet-name>DruidStatView</servlet-name>
     <servlet-class>com.alibaba.druid.support.http.StatViewServlet</servlet-class>
```

```
 </servlet>
 <servlet-mapping>
     <servlet-name>DruidStatView</servlet-name>
     <url-pattern>/druid/*</url-pattern>
 </servlet-mapping>
 <filter>
  <filter-name>druidWebStatFilter</filter-name>
  <filter-class>com.alibaba.druid.support.http.WebStatFilter</filter-class>
  <init-param>
   <param-name>exclusions</param-name>
   <param-value>/public/*,*.js,*.css,/druid*,*.jsp,*.swf</param-value>
  </init-param>
  <init-param>
   <param-name>principalSessionName</param-name>
   <param-value>sessionInfo</param-value>
  </init-param>
  <init-param>
   <param-name>profileEnable</param-name>
   <param-value>true</param-value>
  </init-param>
 </filter>
 <filter-mapping>
  <filter-name>druidWebStatFilter</filter-name>
  <url-pattern>/*</url-pattern>
 </filter-mapping>

</web-app>
```

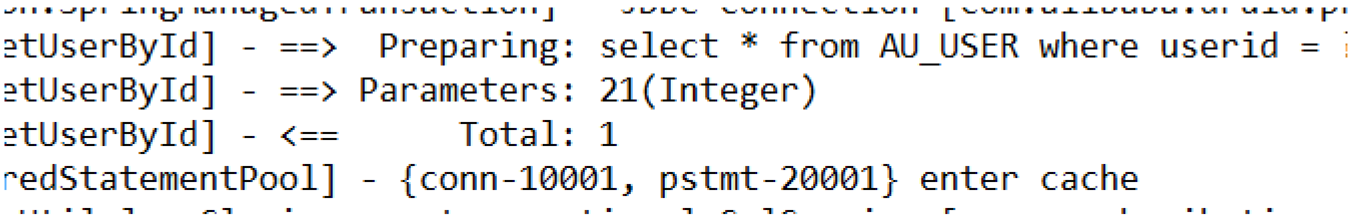
17、UserController.java

```
package com.tp.soft.controller;

import java.util.HashMap;
import java.util.Map;

import javax.annotation.Resource;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import com.tp.soft.entity.User;
import com.tp.soft.service.UserSvc;

@Controller
public class UserController {

    @Resource
    private UserSvc userSvc;

    @RequestMapping(value="/toQueryUser")
    public ModelAndView toQueryUser(){
        User user = userSvc.getUser(21);
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("user", user);
        return new ModelAndView("/pc/userTest", map);
    }
}
```

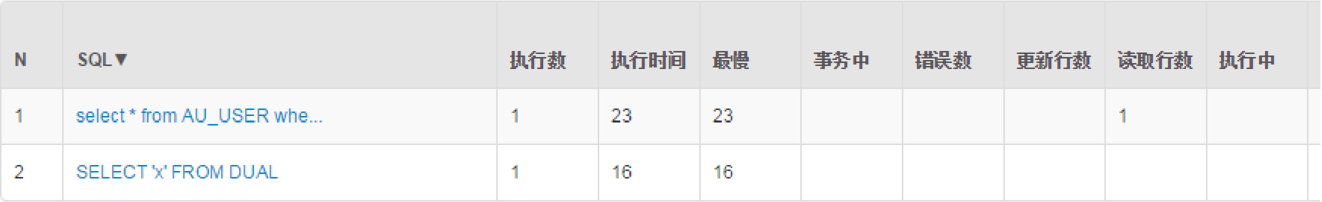
18、访问

http://localhost/taop-web-redis/toQueryUser 第一次

结果



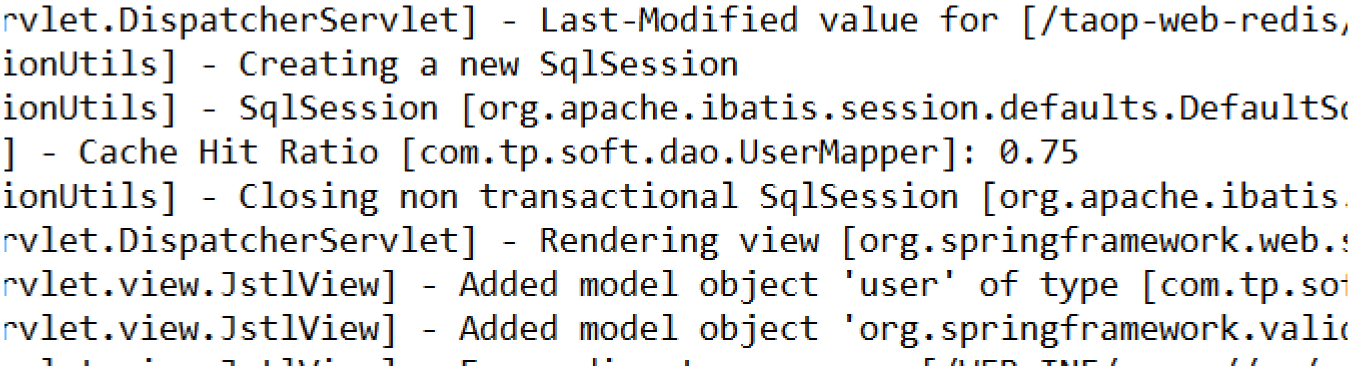
```
on.springmanaged.transaction] - JDBC Connection [com.alibaba.druid.p
etUserById] - ==>  Preparing: select * from AU_USER where userid = 
etUserById] - ==> Parameters: 21(Integer)
etUserById] - <==      Total: 1
redStatementPool] - {conn-10001, pstmt-20001} enter cache
```



N	SQL▼	执行数	执行时间	最慢	事务中	错误数	更新行数	读取行数	执行中
1	select * from AU_USER whe...	1	23	23				1	
2	SELECT 'x' FROM DUAL	1	16	16					

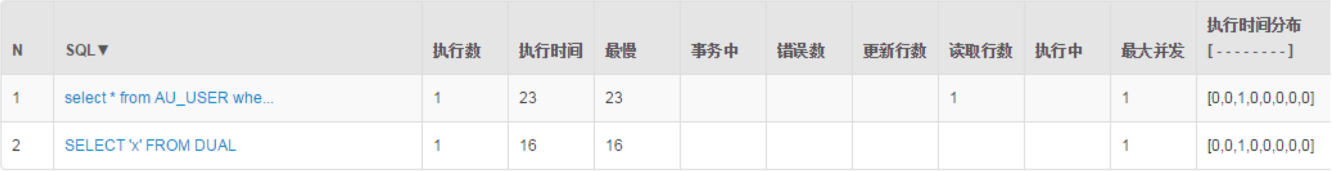
访问第二次

结果



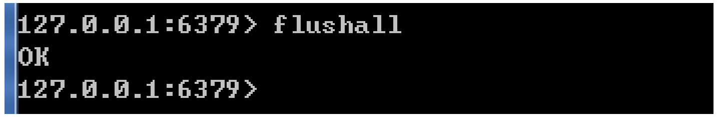
```
rvlet.DispatcherServlet] - Last-Modified value for [/taop-web-redis/
ionUtils] - Creating a new SqlSession
ionUtils] - SqlSession [org.apache.ibatis.session.defaults.DefaultSq
] - Cache Hit Ratio [com.tp.soft.dao.UserMapper]: 0.75
ionUtils] - Closing non transactional SqlSession [org.apache.ibatis.
rvlet.DispatcherServlet] - Rendering view [org.springframework.web.s
rvlet.view.JstlView] - Added model object 'user' of type [com.tp.sof
rvlet.view.JstlView] - Added model object 'org.springframework.valid
```

不再打印sql



N	SQL▼	执行数	执行时间	最慢	事务中	错误数	更新行数	读取行数	执行中	最大并发	执行时间分布 [--------]
1	select * from AU_USER whe...	1	23	23				1		1	[0,0,1,0,0,0,0,0]
2	SELECT 'x' FROM DUAL	1	16	16						1	[0,0,1,0,0,0,0,0]

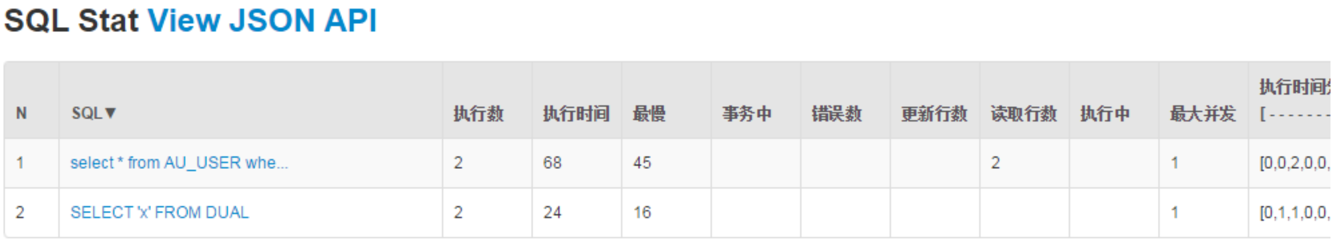
当清空缓存数据



```
127.0.0.1:6379> flushall
OK
127.0.0.1:6379>
```

再请求链接

结果



SQL Stat View JSON API

N	SQL▼	执行数	执行时间	最慢	事务中	错误数	更新行数	读取行数	执行中	最大并发	执行时间 [-------
1	select * from AU_USER whe...	2	68	45				2		1	[0,0,2,0,0,
2	SELECT 'x' FROM DUAL	2	24	16						1	[0,1,1,0,0,

到此 搭建结束，后期将强化项目架构。

标签: ssm+redis整合

好文要顶 关注我 收藏该文

0 0

L_O_V_E_Java
关注 - 0
粉丝 - 5

+加关注

« 上一篇：ssm 整合（方案二 maven）
» 下一篇：ssm+redis整合之redis连接池注入

posted @ 2017-07-13 16:55 L_O_V_E_Java 阅读(4649) 评论(3) 编辑 收藏

发表评论

#1楼 2017-09-20 21:24 | deel_feel

望解答：项目按照你的搭建好，运行没有报错，而查询出来的user对象为空。请问是什么回事，没有报错，又应该去哪里找问题呢？我qq 836543420

支持(0) 反对(0)

#2楼 2017-12-21 14:43 | 习惯沉淀

@ deel_feel
ssm搭建完成后验证可以查到User信息么?

支持(0) 反对(0)

#3楼 2018-04-11 10:49 | 辉辉?

问题是
你这张表的增加删除修改 要如何清空相关联的别的缓存?
功能实现倒不难 可以一一进行配置 但是不知道有没有可以懒省事的一套规则 可以让程序员无需关心这一点。

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 登录 或 注册，访问网站首页。

最新IT新闻:
- “数学王子”高斯诞辰241周年 谷歌涂鸦纪念
- YouTube原创内容负责人：Netflix速度太快追不上
- 美法官：阿里巴巴无权禁止迪拜加密货币公司使用相似商标名
- iPhone X销量未达预期 但横向比较仍是全球最销量手机