**4-7 基于ShardingJdbc的分库分表实践**

**qiyu-live-user-provider**

引入相关依赖：

|  |
| --- |
| XML <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.28</version> </dependency>  <dependency>  <groupId>org.apache.shardingsphere</groupId>  <artifactId>shardingsphere-jdbc-core</artifactId>  <version>5.3.2</version> </dependency>  <dependency>  <groupId>com.baomidou</groupId>  <artifactId>mybatis-plus-boot-starter</artifactId>  <version>3.5.3</version> </dependency> |

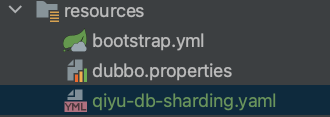
引入依赖之后，我们需要先创建用户中台的数据源配置文件：

接着是数据源的配置项内容，关于数据源的各项属性配置如下文所示：

application.yml配置：

|  |
| --- |
| Java spring:  datasource:  driver-class-name: org.apache.shardingsphere.driver.ShardingSphereDriver  url: jdbc:shardingsphere:classpath:qiyu-db-sharding.yaml |

接下来是我们的ShardingJdbc配置文件( qiyu-db-sharding.yaml ):



qiyu-db-sharding.yaml配置如下：

|  |
| --- |
| Java dataSources:  user: ##新表，重建的分表  dataSourceClassName: com.zaxxer.hikari.HikariDataSource  driver-class-name: com.mysql.cj.jdbc.Driver  jdbcUrl: jdbc:mysql://cloud.db.com:3306/qiyu\_live\_user?useUnicode=true&characterEncoding=utf8  username: qiyu\_user\_28721  password: qiyu\_87123Live@34  idle-timeout: 5 #单位：分钟  minimum-idle: 100  pool-name: qiyu-db-user  maximum-pool-size: 1000  connection-timeout: 30000  connection-init-sql: SELECT 1  connection-test-query: SELECT 1  rules:  - !SINGLE  defaultDataSource: user ## 不分表分分库的默认数据源  - !SHARDING  tables:  t\_user:  actualDataNodes: user.t\_user\_${(0..99).collect(){it.toString().padLeft(2,'0')}}  tableStrategy:  standard:  shardingColumn: user\_id  shardingAlgorithmName: t\_user-inline   shardingAlgorithms:  t\_user-inline:  type: INLINE  props:  algorithm-expression: t\_user\_${(user\_id % 100).toString().padLeft(2,'0')}  props:  sql-show: true |

用户PO对象内容：

|  |
| --- |
| Java package org.qiyu.live.user.provider.dao.po;  import com.baomidou.mybatisplus.annotation.IdType; import com.baomidou.mybatisplus.annotation.TableId; import com.baomidou.mybatisplus.annotation.TableName;  import java.util.Date;  /\*\*  \* @Author idea  \* @Date: Created in 15:54 2023/5/7  \* @Description  \*/ @TableName("t\_user") public class UserPO {   @TableId(type = IdType.INPUT)  private Long userId;  private String nickName;  private String trueName;  private String avatar;  private Integer sex;  private Integer workCity;  private Integer bornCity;  private Date bornDate;  private Date createTime;  private Date updateTime;   public Long getUserId() {  return userId;  }   public void setUserId(Long userId) {  this.userId = userId;  }   public String getNickName() {  return nickName;  }   public void setNickName(String nickName) {  this.nickName = nickName;  }   public String getTrueName() {  return trueName;  }   public void setTrueName(String trueName) {  this.trueName = trueName;  }   public String getAvatar() {  return avatar;  }   public void setAvatar(String avatar) {  this.avatar = avatar;  }   public Integer getSex() {  return sex;  }   public void setSex(Integer sex) {  this.sex = sex;  }   public Integer getWorkCity() {  return workCity;  }   public void setWorkCity(Integer workCity) {  this.workCity = workCity;  }   public Integer getBornCity() {  return bornCity;  }   public void setBornCity(Integer bornCity) {  this.bornCity = bornCity;  }   public Date getBornDate() {  return bornDate;  }   public void setBornDate(Date bornDate) {  this.bornDate = bornDate;  }   public Date getCreateTime() {  return createTime;  }   public void setCreateTime(Date createTime) {  this.createTime = createTime;  }   public Date getUpdateTime() {  return updateTime;  }   public void setUpdateTime(Date updateTime) {  this.updateTime = updateTime;  }   @Override  public String toString() {  return "UserPO{" +  "userId=" + userId +  ", nickName='" + nickName + '\'' +  ", trueName='" + trueName + '\'' +  ", avatar='" + avatar + '\'' +  ", sex=" + sex +  ", workCity=" + workCity +  ", bornCity=" + bornCity +  ", bornDate=" + bornDate +  ", createTime=" + createTime +  ", updateTime=" + updateTime +  '}';  } } |

**qiyu-live-common-interface**

公共组件的依赖都放在了这里面，实现一个bean拷贝的封装工具类：

|  |
| --- |
| Java package org.qiyu.live.common.interfaces.utils;  import org.springframework.beans.BeanInstantiationException; import org.springframework.beans.BeanUtils;  import java.util.ArrayList; import java.util.List;  /\*\*  \* @Author idea  \* @Date: Created in 16:06 2023/5/7  \* @Description  \*/ public class ConvertBeanUtils {   /\*\*  \* 将一个对象转成目标对象  \*  \* @param source  \* @param targetClass  \* @param <T>  \* @return  \*/  public static <T> T convert(Object source, Class<T> targetClass) {  if (source == null) {  return null;  }  T t = newInstance(targetClass);  BeanUtils.copyProperties(source, t);  return t;  }   /\*\*  \* 将List对象转换成目标对象，注意实现是ArrayList  \*  \* @param targetClass  \* @param <K>  \* @param <T>  \* @return  \*/  public static <K, T> List<T> convertList(List<K> sourceList, Class<T> targetClass) {  if (sourceList == null) {  return null;  }  List targetList = new ArrayList((int)(sourceList.size()/0.75) + 1);  for (K source : sourceList) {  targetList.add(convert(source, targetClass));  }  return targetList;  }   private static <T> T newInstance(Class<T> targetClass) {  try {  return targetClass.newInstance();  } catch (Exception e) {  throw new BeanInstantiationException(targetClass, "instantiation error", e);  }  } } |

|  |
| --- |
| Java package org.qiyu.live.user.provider.config;  import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.boot.ApplicationRunner; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration;  import javax.sql.DataSource; import java.sql.Connection;  /\*\*  \* @Author idea  \* @Date: Created in 18:06 2023/5/7  \* @Description  \*/ @Configuration public class ShardingJdbcDatasourceAutoInitConnectionConfig {   private static final Logger LOGGER = LoggerFactory.getLogger(ShardingJdbcDatasourceAutoInitConnectionConfig.class);   @Bean  public ApplicationRunner runner(DataSource dataSource) {  return args -> {  LOGGER.info("dataSource: {}", dataSource);  Connection connection = dataSource.getConnection();  };  } } |