4-17 | 分布式 id 发号器的实现

数据库配置

创建公共的数据库：

```
Java
CREATE DATABASE qiyu_live_common  CHARACTER  set utf8mb3
COLLATE=utf8_bin;
```

设计发号器的表结构：

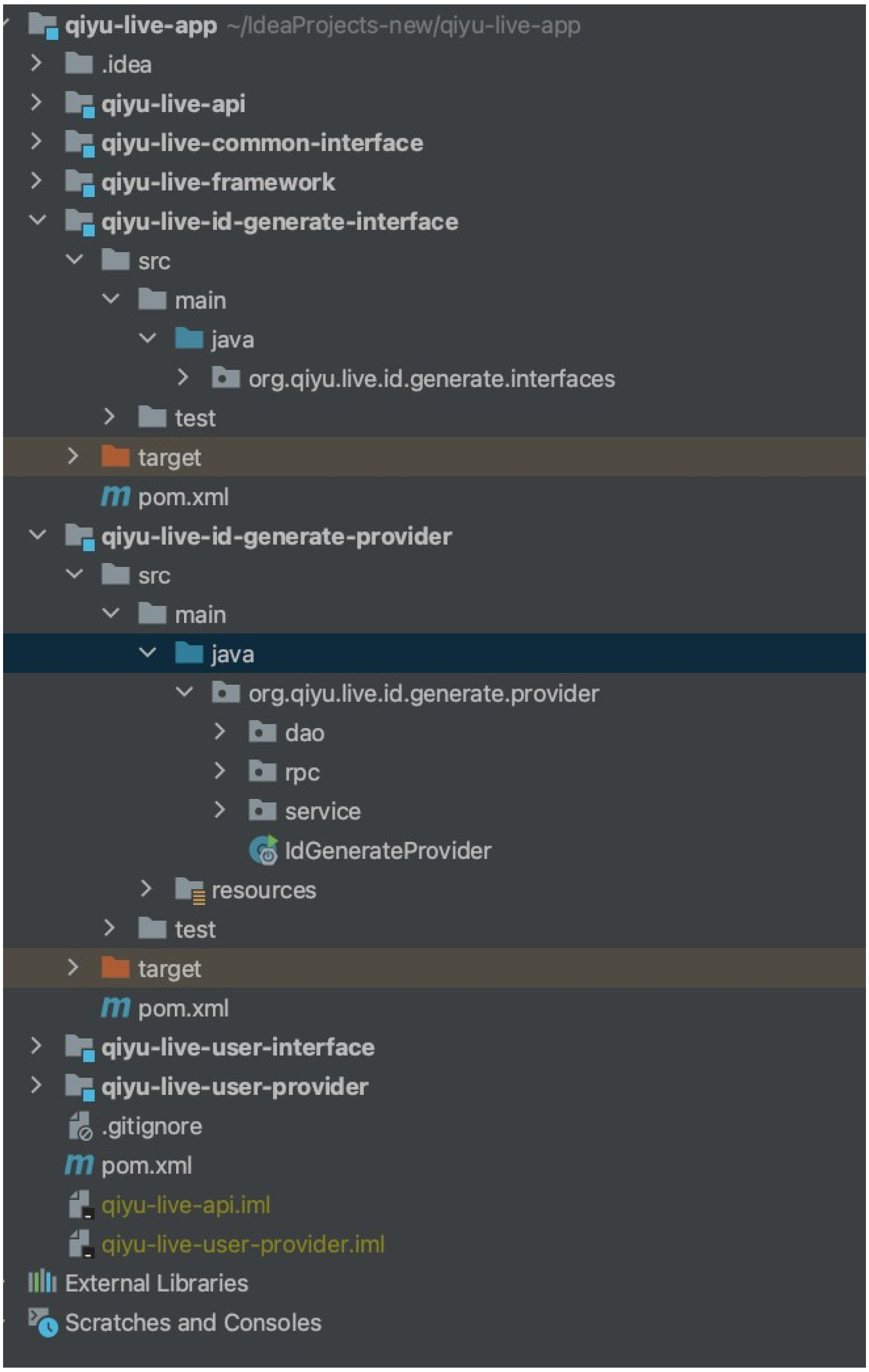
```
Java
CREATE TABLE `t_id_generate_config` (
  `id` int NOT NULL AUTO_INCREMENT COMMENT '主键 id',
  `remark` varchar(255) CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci DEFAULT NULL COMMENT '描述',
  `next_threshold` bigint DEFAULT NULL COMMENT '当前 id 所在阶段的阈
值',
  `init_num` bigint DEFAULT NULL COMMENT '初始化值',
  `current_start` bigint DEFAULT NULL COMMENT '当前 id 所在阶段的开始
值',
  `step` int DEFAULT NULL COMMENT 'id 递增区间',
  `is_seq` tinyint DEFAULT NULL COMMENT '是否有序（0 无序，1 有序）',
  `id_prefix` varchar(60) CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci DEFAULT NULL COMMENT '业务前缀码，如果没有则返回
时不携带',
  `version` int NOT NULL DEFAULT '0' COMMENT '乐观锁版本号',
  `create_time` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '创建时
间',
  `update_time` datetime DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT '更新时间',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

然后往表中初始化一条记录：

```
Java
INSERT INTO `t_id_generate_config` (`id`, `remark`,
`next_threshold`, `init_num`, `current_start`, `step`, `is_seq`,
`id_prefix`, `version`, `create_time`, `update_time`)
VALUES
        (1, '用户 id 生成策略', 10050, 10000, 10000, 50, 0,
'user_id', 0, '2023-05-23 12:38:21', '2023-05-23 23:31:45');
```

初始化项目结构

初始化项目工程，qiyu-live-id-generate-interface 模块和 qiyu-live-id-generate-provider 模块，这两个模块其实就是一个普通的 dubbo 服务组成部分。



```
qiyu-live-app ~/IdeaProjects-new/qiyu-live-app
  .idea
  qiyu-live-api
  qiyu-live-common-interface
  qiyu-live-framework
  qiyu-live-id-generate-interface
    src
      main
        java
          org.qiyu.live.id.generate.interfaces
      test
    target
    pom.xml
  qiyu-live-id-generate-provider
    src
      main
        java
          org.qiyu.live.id.generate.provider
            dao
            rpc
            service
            IdGenerateProvider
        resources
      test
    target
    pom.xml
  qiyu-live-user-interface
  qiyu-live-user-provider
  .gitignore
  pom.xml
  qiyu-live-api.iml
  qiyu-live-user-provider.iml
External Libraries
Scratches and Consoles
```

然后再 provider 层引入以下依赖：

```XML
<properties>
    <mybatis-plus.version>3.5.3</mybatis-plus.version>
    <dubbo.version>3.2.0-beta.3</dubbo.version>
</properties>

<dependencies>
    <dependency>
        <groupId>com.baomidou</groupId>
        <artifactId>mybatis-plus-boot-starter</artifactId>
        <version>${mybatis-plus.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.dubbo</groupId>
        <artifactId>dubbo-spring-boot-starter</artifactId>
        <version>${dubbo.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <exclusions>
            <exclusion>
                <artifactId>log4j-to-slf4j</artifactId>
                <groupId>org.apache.logging.log4j</groupId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>${qiyu-mysql.version}</version>
    </dependency>
    <dependency>
        <groupId>com.alibaba.cloud</groupId>
        <artifactId>spring-cloud-starter-alibaba-nacos-
discovery</artifactId>
    </dependency>
    <dependency>
        <groupId>org.idea</groupId>
        <artifactId>qiyu-live-id-generate-interface</artifactId>
        <version>1.0-SNAPSHOT</version>
        <scope>compile</scope>
    </dependency>
</dependencies>
```

相关的 application.yaml 配置：

```YAML
spring:
  application:
    name: qiyu-live-id-generate-provider
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    #访问主库
    url: jdbc:mysql://cloud.db:8808/qiyu_live_common?
useUnicode=true&characterEncoding=utf8
    username: root
    password: root

dubbo:
  application:
    name: ${spring.application.name}
  registry:
    address: nacos://127.0.0.1:8848?namespace=qiyu-live-
test&&username=qiyu&&password=qiyu
  protocol:
    name: dubbo
    port: 9098
```

bootstrap.yml 配置；

```YAML
spring:
  cloud:
    nacos:
      username: qiyu
      password: qiyu
      discovery:
        server-addr: qiyu.test.com:8848
        namespace: qiyu-live-test
```

接口定义

定义我们的接口对象：

qiyu-live-id-generate-interfaces 层定义接口，代码如下：

```Java
package org.qiyu.live.id.generate.interfaces;

```

```
public interface IdBuilderRpc {

    /**
     * 根据本地步长度来生成唯一id(区间性递增)
     *
     * @return
     */
    Long increaseSeqId(int code);

    /**
     * 生成的是非连续性id
     *
     * @param code
     * @return
     */
    Long increaseUnSeqId(int code);

    /**
     * 根据本地步长度来生成唯一id(区间性递增)
     *
     * @param code
     * @return
     */
    String increaseSeqStrId(int code);
}
```

qiyu-live-id-generate-provider 层进行接口的 RPC 类实现：

```Java
package org.qiyu.live.id.generate.provider.rpc;

import jakarta.annotation.Resource;
import org.apache.dubbo.config.annotation.DubboService;
import org.qiyu.live.id.generate.interfaces.IdBuilderRpc;
import
org.qiyu.live.id.generate.provider.service.IdBuilderService;

/**
 * @Author idea
 * @Date: Created in 19:54 2023/5/23
 * @Description
 */
@DubboService
```

```
public class IdBuilderRpcImpl implements IdBuilderRpc {

    @Resource
    private IdBuilderService idBuilderService;

    @Override
    public Long increaseSeqId(int code) {
        return idBuilderService.increaseSeqId(code);
    }

    @Override
    public Long increaseUnSeqId(int code) {
        return idBuilderService.increaseUnSeqId(code);
    }

    @Override
    public String increaseSeqStrId(int code) {
        return idBuilderService.increaseSeqStrId(code);
    }
}
```

然后是内部的 service 层实现，service 层的接口定义：

```Java
package org.qiyu.live.id.generate.provider.service;

/**
 * @Author idea
 * @Date: Created in 15:47 2023/5/24
 * @Description
 */
public interface IdGenerateService {

    /**
     * 生成有序id
     *
     * @param code
     * @return
     */
    Long getSeqId(Integer code);

    /**
     * 生成无序id
     *
     * @param code
     * @return
     */
    Long getUnSeqId(Integer code);
}
```

Service 层核心实现

service 层的实现代码：

```
Java
package org.qiyu.live.id.generate.provider.service.impl;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import jakarta.annotation.Resource;
import org.qiyu.live.common.interfaces.ConvertBeanUtils;
import
org.qiyu.live.id.generate.provider.dao.mapper.IdBuilderMapper;
import org.qiyu.live.id.generate.provider.dao.po.IdBuilderPO;
import
org.qiyu.live.id.generate.provider.service.IdGenerateService;
import org.qiyu.live.id.generate.provider.service.bo.LocalSeqIdBO;
import
org.qiyu.live.id.generate.provider.service.bo.LocalUnSeqIdBO;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.stereotype.Service;

import java.util.*;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.concurrent.ThreadLocalRandom;
import java.util.concurrent.atomic.AtomicLong;
import java.util.concurrent.locks.ReentrantLock;
import java.util.stream.Collectors;

/**
 * @Author idea
 * @Date: Created in 15:47 2023/5/24
 * @Description
 */
@Service
public class IdGenerateServiceImpl implements IdGenerateService,
InitializingBean {



    @Resource
    private IdBuilderMapper idBuilderMapper;
    private static Map<Integer, LocalSeqIdBO> localSeqIdMap = new
ConcurrentHashMap<>();
    private static Map<Integer, LocalUnSeqIdBO> localUnSeqIdMap =
new ConcurrentHashMap<>();
    private static final int SEQ_FLAG = 1;
    private static final int RETRY_TIMES = 3;
    private static final Logger LOGGER =
LoggerFactory.getLogger(IdGenerateServiceImpl.class);
    private static ReentrantLock SEQ_LOCK = new ReentrantLock();
    private static ReentrantLock UN_SEQ_LOCK = new
ReentrantLock();

    //采用了局部有序性去进行设计，不能保证 id 段完全用完
    @Override
    public Long getSeqId(Integer code) {
        LocalSeqIdBO localSeqIdBO = localSeqIdMap.get(code);
        if (localSeqIdBO == null) {
            LOGGER.error("[getSeqId] code is error,{}", code);
            return null;
        }
        long returnId =
localSeqIdBO.getCurrentValue().getAndIncrement();
        if (returnId - localSeqIdBO.getCurrentStart() >
localSeqIdBO.getStep() * 0.75) {
            //进行一个本地 id 段的更新操作
            this.refreshLocalSeqId(localSeqIdBO.getId());
        }
        return returnId;
    }

    @Override
    public Long getUnSeqId(Integer code) {
        LocalUnSeqIdBO localUnSeqIdBO = localUnSeqIdMap.get(code);
        if (localUnSeqIdBO == null) {
            LOGGER.error("[getUnSeqId] code is error,{}", code);
            return null;
        }
        Long unSeqId = localUnSeqIdBO.getIdQueue().poll();
        if (localUnSeqIdBO.getIdQueue().size() <
localUnSeqIdBO.getStep() * 0.25) {
            this.refreshLocalUnSeqId(localUnSeqIdBO.getId());
        }
        return unSeqId;
    }

    /**
     * 刷新无序 id 段，加载到本地内存中
     *
     * @param code
     */
    private void refreshLocalUnSeqId(Integer code) {
        boolean lockStatus = false;
        try {
            lockStatus = UN_SEQ_LOCK.tryLock();
            if (lockStatus) {
                for (int i = 0; i < RETRY_TIMES; i++) {
                    IdBuilderPO idBuilderPO =
idBuilderMapper.selectById(code);
                    if (idBuilderPO == null) {
                        LOGGER.error("[refreshLocalSeqId] code is
error,{}", code);
                        return;
                    }
                    long nextThreshold =
idBuilderPO.getNextThreshold() + idBuilderPO.getStep();
                    long currentStart =
idBuilderPO.getNextThreshold();
                    int result =
idBuilderMapper.updateCurrentThreshold(nextThreshold,
                            currentStart, idBuilderPO.getId(),
idBuilderPO.getVersion());
                    if (result < 1) {
                        continue;
                    }
                    LocalUnSeqIdBO localUnSeqIdBO = new
LocalUnSeqIdBO();
                    localUnSeqIdBO.setId(idBuilderPO.getId());
                    localUnSeqIdBO.setStep(idBuilderPO.getStep());

localUnSeqIdBO.setNextThreshold(nextThreshold);
                    localUnSeqIdBO.setCurrentStart(currentStart);

localUnSeqIdBO.setRandomIdInQueue(currentStart,nextThreshold);
```

```
                    localUnSeqIdMap.put(idBuilderPO.getId(),
localUnSeqIdBO);
                    break;
                }
            }
        } catch (Exception e) {
            LOGGER.error("[refreshLocalUnSeqId] code is {}, error
is ", e, code);
        } finally {
            if (lockStatus) {
                UN_SEQ_LOCK.unlock();
            }
        }

    }

    /**
     * 刷新有序 id 段，加载到本地内存中
     *
     * @param code
     */
    private void refreshLocalSeqId(Integer code) {
        boolean lockStatus = false;
        try {
            lockStatus = SEQ_LOCK.tryLock();
            //防止多线程进入下方程序逻辑中
            if (lockStatus) {
                for (int i = 0; i < RETRY_TIMES; i++) {
                    IdBuilderPO idBuilderPO =
idBuilderMapper.selectById(code);
                    if (idBuilderPO == null) {
                        LOGGER.error("[refreshLocalSeqId] code is
error,{}", code);
                        return;
                    }
                    long nextThreshold =
idBuilderPO.getNextThreshold() + idBuilderPO.getStep();
                    long currentStart =
idBuilderPO.getNextThreshold();
                    AtomicLong currentValue = new
AtomicLong(idBuilderPO.getNextThreshold());
                    int result =
idBuilderMapper.updateCurrentThreshold(nextThreshold,
                            currentStart, idBuilderPO.getId(),
idBuilderPO.getVersion());
                    if (result < 1) {
                        continue;
                    }
                    LocalSeqIdBO localSeqIdBO = new
LocalSeqIdBO();
                    localSeqIdBO.setId(idBuilderPO.getId());
                    localSeqIdBO.setCurrentValue(currentValue);
                    localSeqIdBO.setCurrentStart(currentStart);
                    localSeqIdBO.setNextThreshold(nextThreshold);
                    localSeqIdBO.setStep(idBuilderPO.getStep());
                    localSeqIdMap.put(idBuilderPO.getId(),
localSeqIdBO);
                    break;
                }
            }
        } catch (Exception e) {
            LOGGER.error("[refreshLocalSeqId] code is {},error is
", e, code);
        } finally {
            if (lockStatus) {
                SEQ_LOCK.unlock();
            }
        }
    }

    @Override
    public void afterPropertiesSet() {
        //在启动之前，将mysql的id配置初始化到本地内存中
        List<IdBuilderPO> idBuilderPOList =
idBuilderMapper.selectAll();
        for (IdBuilderPO idBuilderPO : idBuilderPOList) {
            int updateStatus =
idBuilderMapper.updateNewVersion(idBuilderPO.getId());
            if (updateStatus > 0) {
                if (idBuilderPO.getIsSeq() == SEQ_FLAG) {
                    LocalSeqIdBO localSeqIdBO = new
LocalSeqIdBO();
                    localSeqIdBO.setId(idBuilderPO.getId());
                    localSeqIdBO.setStep(idBuilderPO.getStep());

localSeqIdBO.setNextThreshold(idBuilderPO.getNextThreshold() +
idBuilderPO.getStep());
```

```
localSeqIdBO.setCurrentStart(idBuilderPO.getNextThreshold());
                    AtomicLong currentValue = new
AtomicLong(idBuilderPO.getNextThreshold());
                    localSeqIdBO.setCurrentValue(currentValue);
                    localSeqIdMap.put(idBuilderPO.getId(),
localSeqIdBO);
                } else {
                    LocalUnSeqIdBO localUnSeqIdBO = new
LocalUnSeqIdBO();
                    localUnSeqIdBO.setId(idBuilderPO.getId());
                    localUnSeqIdBO.setStep(idBuilderPO.getStep());

localUnSeqIdBO.setNextThreshold(idBuilderPO.getNextThreshold() +
idBuilderPO.getStep());

localUnSeqIdBO.setCurrentStart(idBuilderPO.getNextThreshold());

localUnSeqIdBO.setRandomIdInQueue(idBuilderPO.getCurrentStart(),id
BuilderPO.getNextThreshold());
                    localUnSeqIdMap.put(idBuilderPO.getId(),
localUnSeqIdBO);
                }
            }
        }
    }
}
```

相关 BO 对象的实现：

```Java
package org.qiyu.live.id.generate.provider.service.bo;

import java.util.concurrent.atomic.AtomicInteger;
import java.util.concurrent.atomic.AtomicLong;

/**
 * @Author idea
 * @Date: Created in 15:49 2023/5/24
 * @Description
 */
public class LocalSeqIdBO {

    //mysql 配置的 id
    private Integer id;
    //对应分布式 id 的配置说明
    private String desc;
    //当前在本地内存的 id 值
    private AtomicLong currentValue;
    //本地内存记录 id 段的开始位置
    private Long currentStart;
    //本地内存记录 id 段的结束位置
    private Long nextThreshold;
    //步长
    private Integer step;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getDesc() {
        return desc;
    }

    public void setDesc(String desc) {
        this.desc = desc;
    }

    public Integer getStep() {
        return step;
    }

    public void setStep(Integer step) {
        this.step = step;
    }

    public Long getCurrentStart() {
        return currentStart;
    }

    public void setCurrentStart(Long currentStart) {
        this.currentStart = currentStart;
    }

    public Long getNextThreshold() {
        return nextThreshold;
    }

    public void setNextThreshold(Long nextThreshold) {
        this.nextThreshold = nextThreshold;
    }

    public AtomicLong getCurrentValue() {
        return currentValue;
    }

    public void setCurrentValue(AtomicLong currentValue) {
        this.currentValue = currentValue;
    }

    @Override
    public String toString() {
        return "LocalSeqIdBO{" +
                "id=" + id +
                ", desc='" + desc + '\'' +
                ", currentValue=" + currentValue +
                ", currentStart=" + currentStart +
                ", nextThreshold=" + nextThreshold +
                ", step=" + step +
                '}';
    }
}
```

```
Java
package org.qiyu.live.id.generate.provider.service.bo;

import java.util.Collections;
import java.util.LinkedList;
import java.util.List;
import java.util.concurrent.ConcurrentLinkedQueue;

/**
 * @Author idea
 * @Date: Created in 19:34 2023/5/24
 * @Description
 */
public class LocalUnSeqIdBO {
```

```
//mysql 配置的 id
private Integer id;
//对应分布式 id 的配置说明
private String desc;
//链表记录 id 值
private ConcurrentLinkedQueue<Long> idQueue;
//本地内存记录 id 段的开始位置
private Long currentStart;
//本地内存记录 id 段的结束位置
private Long nextThreshold;
//步长
private Integer step;

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getDesc() {
    return desc;
}

public void setDesc(String desc) {
    this.desc = desc;
}

public ConcurrentLinkedQueue<Long> getIdQueue() {
    return idQueue;
}

public void setIdQueue(ConcurrentLinkedQueue<Long> idQueue) {
    this.idQueue = idQueue;
}

public void setRandomIdInQueue(long begin,long end) {
    List<Long> idList = new LinkedList<>();
    for (long j = begin; j < end; j++) {
        idList.add(j);
    }
    //把队列的元素进行打乱
```

```
        Collections.shuffle(idList);
        ConcurrentLinkedQueue idQueue = new
ConcurrentLinkedQueue();
        idQueue.addAll(idList);
        this.setIdQueue(idQueue);
    }

    public Long getCurrentStart() {
        return currentStart;
    }

    public void setCurrentStart(Long currentStart) {
        this.currentStart = currentStart;
    }

    public Long getNextThreshold() {
        return nextThreshold;
    }

    public void setNextThreshold(Long nextThreshold) {
        this.nextThreshold = nextThreshold;
    }

    public Integer getStep() {
        return step;
    }

    public void setStep(Integer step) {
        this.step = step;
    }
}
```

接下来我们需要按照表结构，去定义我们的 PO 对象：

```java
Java
package org.qiyu.live.id.generate.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 19:59 2023/5/23
 * @Description
 */
@TableName("t_id_builder_config")
public class IdBuilderPO {

    @TableId(type = IdType.AUTO)
    private Integer id;

    /**
     * id 备注描述
     */
    private String remark;

    /**
     * 初始化值
     */
    private long initNum;

    /**
     * 步长
     */
    private int step;

    /**
     * 是否是有序的 id
     */
    private int isSeq;

    /**
     * 当前 id 所在阶段的开始值
     */
    private long currentStart;

    /**
     * 当前 id 所在阶段的阈值
     */
    private long nextThreshold;

    /**
```

```
     * 业务代码前缀
     */
    private String idPrefix;

    /**
     * 乐观锁版本号
     */
    private int version;

    private Date createTime;

    private Date updateTime;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getRemark() {
        return remark;
    }

    public void setRemark(String remark) {
        this.remark = remark;
    }

    public long getInitNum() {
        return initNum;
    }

    public void setInitNum(long initNum) {
        this.initNum = initNum;
    }

    public int getStep() {
        return step;
    }

    public void setStep(int step) {
        this.step = step;
    }
    public long getCurrentStart() {
        return currentStart;
    }

    public void setCurrentStart(long currentStart) {
        this.currentStart = currentStart;
    }

    public long getNextThreshold() {
        return nextThreshold;
    }

    public void setNextThreshold(long nextThreshold) {
        this.nextThreshold = nextThreshold;
    }

    public String getIdPrefix() {
        return idPrefix;
    }

    public void setIdPrefix(String idPrefix) {
        this.idPrefix = idPrefix;
    }

    public int getVersion() {
        return version;
    }

    public void setVersion(int version) {
        this.version = version;
    }

    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;
    }

    public int getIsSeq() {
        return isSeq;
    }

    public void setIsSeq(int isSeq) {
        this.isSeq = isSeq;
    }

    @Override
    public String toString() {
        return "IdBuilderPO{" +
                "id=" + id +
                ", remark='" + remark + '\'' +
                ", initNum=" + initNum +
                ", step=" + step +
                ", isSeq=" + isSeq +
                ", currentStart=" + currentStart +
                ", nextThreshold=" + nextThreshold +
                ", idPrefix='" + idPrefix + '\'' +
                ", version=" + version +
                ", createTime=" + createTime +
                ", updateTime=" + updateTime +
                '}';
    }
}
```

引入我们的 mapper 对象：

```Java
package org.qiyu.live.id.generate.provider.dao.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Param;
import org.apache.ibatis.annotations.Select;
import org.apache.ibatis.annotations.Update;
import org.qiyu.live.id.generate.provider.dao.po.IdBuilderPO;

import java.util.List;
```

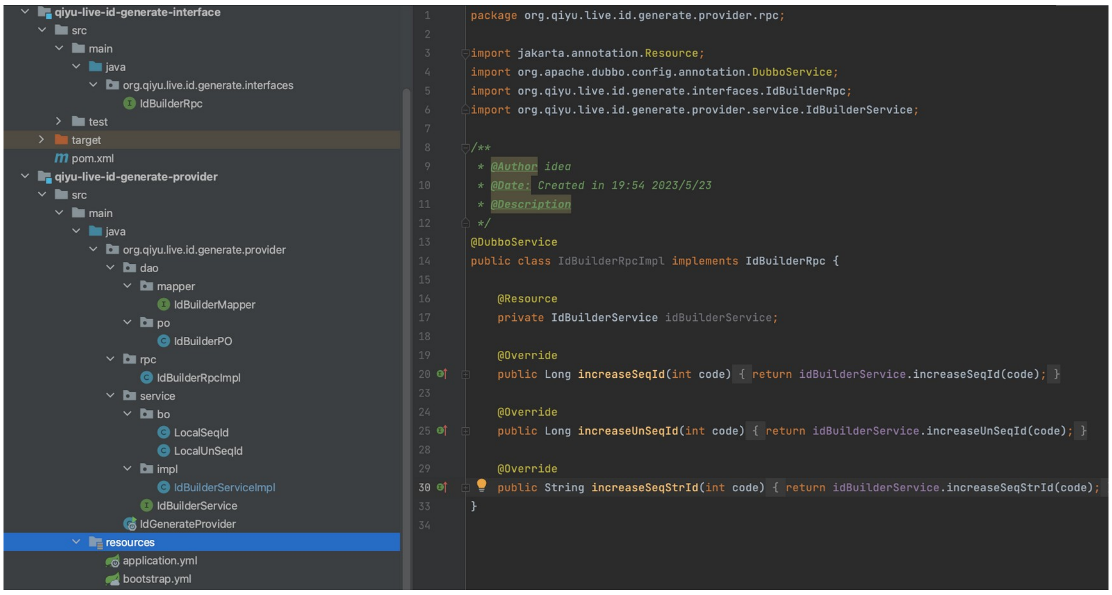
```java
/**
 * @Author idea
 * @Date: Created in 15:46 2023/5/24
 * @Description
 */
@Mapper
public interface IdBuilderMapper extends BaseMapper<IdBuilderPO> {

    @Update("update t_id_builder_config set current_start = 
next_threshold,next_threshold=current_start+step,version=version+1 
where id=#{code}")
    Integer updateNewVersion(@Param("code") Integer code);

    @Update("UPDATE t_id_builder_config set 
next_threshold=#{nextThreshold},current_start=#{currentStart},vers
ion=version+1 where id=#{id} and version=#{version}")
    Integer updateCurrentThreshold(@Param("nextThreshold") long 
nextThreshold, @Param("currentStart") long currentStart, 
@Param("id") int id, @Param("version") int version);

    @Select("select * from t_id_builder_config")
    List<IdBuilderPO> selectAll();
}
```

完成以上几个步骤之后，我们就可以去测试本项目的 rpc 接口了。



```
qiyu-live-id-generate-interface
  src
    main
      java
        org.qiyu.live.id.generate.interfaces
          IdBuilderRpc
    test
  target
  pom.xml
qiyu-live-id-generate-provider
  src
    main
      java
        org.qiyu.live.id.generate.provider
          dao
            mapper
              IdBuilderMapper
            po
              IdBuilderPO
          rpc
            IdBuilderRpcImpl
          service
            bo
              LocalSeqId
              LocalUnSeqId
            impl
              IdBuilderServiceImpl
            IdBuilderService
          IdGenerateProvider
      resources
        application.yml
        bootstrap.yml
```

```java
package org.qiyu.live.id.generate.provider.rpc;

import jakarta.annotation.Resource;
import org.apache.dubbo.config.annotation.DubboService;
import org.qiyu.live.id.generate.interfaces.IdBuilderRpc;
import org.qiyu.live.id.generate.provider.service.IdBuilderService;

/**
 * @Author idea
 * @Date: Created in 19:54 2023/5/23
 * @Description
 */
@DubboService
public class IdBuilderRpcImpl implements IdBuilderRpc {

    @Resource
    private IdBuilderService idBuilderService;

    @Override
    public Long increaseSeqId(int code) { return idBuilderService.increaseSeqId(code); }

    @Override
    public Long increaseUnSeqId(int code) { return idBuilderService.increaseUnSeqId(code); }

    @Override
    public String increaseSeqStrId(int code) { return idBuilderService.increaseSeqStrId(code);
}

```