day10_数据保存&排重&文档解析

1 数据保存准备

1.1 ip代理池

1.1.1 需求分析

针对于ip代理池的管理，包括了增删改查，设置可用ip和不可用ip

1.1.2 实体类

ClIpPool类

com.heima.model.crawler.pojos.ClIpPool

```
@Data
public class ClIpPool {

    private Integer id;

    private String supplier;

    private String ip;
    /**
     * 端口号
     */
    private int port;

    /**
     * 用户名
     */
    private String username;

    /**
     * 密码
     */
    private String password;

    /**
     * 错误码
     */
    private Integer code;

    /**
     * 耗时
     */
    private Integer duration;

    /**
```

```
     * 错误信息
     */
    private String error;

    private Boolean isEnable;


    private String ranges;


    private Date createdTime;

    public Integer getId() {
        return id;
    }

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

    public String getSupplier() {
        return supplier;
    }

    public void setSupplier(String supplier) {
        this.supplier = supplier;
    }

    public String getIp() {
        return ip;
    }

    public void setIp(String ip) {
        this.ip = ip;
    }

    public int getPort() {
        return port;
    }

    public void setPort(int port) {
        this.port = port;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
```

```
    }

    public Boolean getEnable() {
        return isEnable;
    }

    public void setEnable(Boolean enable) {
        isEnable = enable;
    }

    public String getRanges() {
        return ranges;
    }

    public void setRanges(String ranges) {
        this.ranges = ranges;
    }

    public Date getCreatedTime() {
        return createdTime;
    }

    public void setCreatedTime(Date createdTime) {
        this.createdTime = createdTime;
    }


    public Integer getCode() {
        return code;
    }

    public void setCode(Integer code) {
        this.code = code;
    }

    public Integer getDuration() {
        return duration;
    }

    public void setDuration(Integer duration) {
        this.duration = duration;
    }

    public String getError() {
        return error;
    }

    public void setError(String error) {
        this.error = error;
    }
}
```

ClIpPoolMapper

com.heima.model.mappers.crawerls.ClIpPoolMapper

```
public interface ClIpPoolMapper {
```

```
    int deleteByPrimaryKey(Integer id);

    int insert(ClIpPool record);

    int insertSelective(ClIpPool record);

    ClIpPool selectByPrimaryKey(Integer id);

    int updateByPrimaryKeySelective(ClIpPool record);

    int updateByPrimaryKey(ClIpPool record);

    /**
     * 查询所有数据
     *
     * @param record
     * @return
     */
    List<ClIpPool> selectList(ClIpPool record);

    /**
     * 查询可用的列表
     *
     * @param record
     * @return
     */
    List<ClIpPool> selectAvailableList(ClIpPool record);
}
```

1.1.3 mapper接口定义

ClIpPoolMapper.xml

mappers/crawerls/ClIpPoolMapper.xml

```
<?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.heima.model.mappers.crawerls.ClIpPoolMapper">
    <resultMap id="BaseResultMap" type="com.heima.model.crawler.pojos.ClIpPool">
        <id column="id" property="id"/>
        <result column="supplier" property="supplier"/>
        <result column="ip" property="ip"/>
        <result column="port" property="port"/>
        <result column="username" property="username"/>
        <result column="password" property="password"/>
        <result column="code" property="code"/>
        <result column="duration" property="duration"/>
        <result column="error" property="error"/>
        <result column="is_enable" property="isEnable"/>
        <result column="ranges" property="ranges"/>
        <result column="created_time" property="createdTime"/>
    </resultMap>

    <sql id="Base_Column_where">
        <where>
```

```
            <if test="supplier!=null and supplier!=''">
                and supplier = #{supplier}
            </if>
            <if test="ip!=null and ip!=''">
                and ip = #{ip}
            </if>
            <if test="port!=null and port!=''">
                and port = #{port}
            </if>
            <if test="username!=null and username!=''">
                and username = #{username}
            </if>
            <if test="password!=null and password!=''">
                and password = #{password}
            </if>
            <if test="code!=null and code!=''">
                and code = #{code}
            </if>
            <if test="duration!=null and duration!=''">
                and duration = #{duration}
            </if>
            <if test="error!=null and error!=''">
                and error = #{error}
            </if>
            <if test="isEnable!=null and isEnable!=''">
                and is_enable = #{isEnable}
            </if>
            <if test="ranges!=null and ranges!=''">
                and ranges = #{ranges}
            </if>
        </where>
    </sql>
    <sql id="Base_Column_List">
    id, supplier, ip, port,username,password,code,duration,error,is_enable,
ranges, created_time
  </sql>
    <select id="selectList" resultMap="BaseResultMap">
        select
        <include refid="Base_Column_List"/>
        from cl_ip_pool
        <include refid="Base_Column_where"/>
    </select>
    <select id="selectAvailableList" resultMap="BaseResultMap">
        select
        <include refid="Base_Column_List"/>
        from cl_ip_pool
        <where>
            and is_enable = true
            <if test="duration!=null and duration!=''">
                <![CDATA[
                and duration <= #{duration}
                 ]]>
            </if>
        </where>
        order by duration
    </select>
    <select id="selectByPrimaryKey" resultMap="BaseResultMap"
parameterType="java.lang.Integer">
```

```
        select
        <include refid="Base_Column_List"/>
        from cl_ip_pool
        where id = #{id}
    </select>
    <delete id="deleteByPrimaryKey" parameterType="java.lang.Integer">
    delete from cl_ip_pool
    where id = #{id}
  </delete>
    <insert id="insert" parameterType="com.heima.model.crawler.pojos.ClIpPool"
            useGeneratedKeys="true" keyProperty="id">
    insert into cl_ip_pool (id, supplier, ip,
port,username,password,code,duration,error,
      is_enable, ranges, created_time
      )
    values (#{id}, #{supplier}, #{ip}, #{port}, #{username}, #{password},#
{code},#{duration},#{error},
      #{isEnable}, #{ranges}, #{createdTime}
      )
  </insert>
    <insert id="insertSelective"
parameterType="com.heima.model.crawler.pojos.ClIpPool" keyProperty="id"
useGeneratedKeys="true">

        insert into cl_ip_pool
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="id != null">
                id,
            </if>
            <if test="supplier != null">
                supplier,
            </if>
            <if test="ip != null">
                ip,
            </if>
            <if test="port != null">
                port,
            </if>
            <if test="username != null">
                username,
            </if>
            <if test="password != null">
                password,
            </if>
            <if test="code != null">
                code,
            </if>
            <if test="duration != null">
                duration,
            </if>
            <if test="error != null">
                error,
            </if>
            <if test="isEnable != null">
                is_enable,
            </if>
            <if test="ranges != null">
                ranges,
```

```
            </if>
            <if test="createdTime != null">
                created_time,
            </if>
        </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="id != null">
                #{id},
            </if>
            <if test="supplier != null">
                #{supplier},
            </if>
            <if test="ip != null">
                #{ip},
            </if>
            <if test="port != null">
                #{port},
            </if>
            <if test="username != null">
                #{username},
            </if>
            <if test="password != null">
                #{password},
            </if>
            <if test="code != null">
                #{code},
            </if>
            <if test="duration != null">
                #{duration},
            </if>
            <if test="error != null">
                #{error},
            </if>
            <if test="isEnable != null">
                #{isEnable},
            </if>
            <if test="ranges != null">
                #{ranges},
            </if>
            <if test="createdTime != null">
                #{createdTime},
            </if>
        </trim>
    </insert>
    <update id="updateByPrimaryKeySelective"
parameterType="com.heima.model.crawler.pojos.ClIpPool">

        update cl_ip_pool
        <set>
            <if test="supplier != null">
                supplier = #{supplier},
            </if>
            <if test="ip != null">
                ip = #{ip},
            </if>
            <if test="port != null">
                port = #{port},
            </if>
```

```
            <if test="username != null">
                username = #{username},
            </if>
            <if test="password != null">
                password = #{password},
            </if>
            <if test="code != null">
                code = #{code},
            </if>
            <if test="duration != null">
                duration = #{duration},
            </if>
            <if test="error != null">
                error = #{error},
            </if>
            <if test="isEnable != null">
                is_enable = #{isEnable},
            </if>
            <if test="ranges != null">
                ranges = #{ranges},
            </if>
            <if test="createdTime != null">
                created_time = #{createdTime},
            </if>
        </set>
        where id = #{id}
    </update>
    <update id="updateByPrimaryKey"
parameterType="com.heima.model.crawler.pojos.ClIpPool">

    update cl_ip_pool
    set supplier = #{supplier},
      ip = #{ip},
      port = #{port},
      username = #{username},
      password = #{password},
      code = #{code},
      duration = #{duration},
      error = #{error},
      is_enable = #{isEnable},
      ranges = #{ranges},
      created_time = #{createdTime}
    where id = #{id}
  </update>
</mapper>
```

1.1.4 service代码

CrawlerIpPoolService

com.heima.crawler.service.CrawlerIpPoolService

```
public interface CrawlerIpPoolService {

    /**
     * 保存方法
     *
```

```
     * @param clIpPool
     */
    public void saveCrawlerIpPool(ClIpPool clIpPool);

    /**
     * 检查代理Ip 是否存在
     *
     * @param host
     * @param port
     * @return
     */
    public boolean checkExist(String host, int port);

    /**
     * 更新方法
     *
     * @param clIpPool
     */
    public void updateCrawlerIpPool(ClIpPool clIpPool);

    /**
     * 查询所有数据
     *
     * @param clIpPool
     */
    public List<ClIpPool> queryList(ClIpPool clIpPool);

    /**
     * 获取可用的列表
     *
     * @return
     */
    public List<ClIpPool> queryAvailableList(ClIpPool clIpPool);


    public void delete(ClIpPool clIpPool);


    void unvailableProxy(CrawlerProxy proxy, String errorMsg);
}
```

CrawlerIpPoolServiceImpl

com.heima.crawler.service.impl.CrawlerIpPoolServiceImpl

```
@Service
public class CrawlerIpPoolServiceImpl implements CrawlerIpPoolService {
    @Autowired
    private ClIpPoolMapper clIpPoolMapper;

    @Override
    public void saveCrawlerIpPool(ClIpPool clIpPool) {
        clIpPoolMapper.insertSelective(clIpPool);
    }

    @Override
    public boolean checkExist(String host, int port) {
```

```
        ClIpPool clIpPool = new ClIpPool();
        clIpPool.setIp(host);
        clIpPool.setPort(port);
        List<ClIpPool> clIpPoolList = clIpPoolMapper.selectList(clIpPool);
        if (null != clIpPoolList && !clIpPoolList.isEmpty()) {
            return true;
        }
        return false;
    }

    @Override
    public void updateCrawlerIpPool(ClIpPool clIpPool) {
        clIpPoolMapper.updateByPrimaryKey(clIpPool);
    }

    @Override
    public List<ClIpPool> queryList(ClIpPool clIpPool) {
        return clIpPoolMapper.selectList(clIpPool);
    }

    @Override
    public List<ClIpPool> queryAvailableList(ClIpPool clIpPool) {
        return clIpPoolMapper.selectAvailableList(clIpPool);
    }

    @Override
    public void delete(ClIpPool clIpPool) {
        clIpPoolMapper.deleteByPrimaryKey(clIpPool.getId());
    }

    @Override
    public void unvailableProxy(CrawlerProxy proxy, String errorMsg) {
        ClIpPool clIpPoolQuery = new ClIpPool();
        clIpPoolQuery.setIp(proxy.getHost());
        clIpPoolQuery.setPort(proxy.getPort());
        clIpPoolQuery.setEnable(true);
        List<ClIpPool> clIpPoolList = clIpPoolMapper.selectList(clIpPoolQuery);
        if (null != clIpPoolList && !clIpPoolList.isEmpty()) {
            for (ClIpPool clIpPool : clIpPoolList) {
                clIpPool.setEnable(false);
                clIpPool.setError(errorMsg);
                clIpPoolMapper.updateByPrimaryKey(clIpPool);
            }
        }
    }
}
```

1.1.5 测试

```
@SpringBootTest
@RunWith(SpringRunner.class)
public class CrawlerIpPoolServiceTest {

    @Autowired
    private CrawlerIpPoolService crawlerIpPoolService;

    @Test
```

```
    public void testSaveCrawlerIpPool(){
        ClIpPool clIpPool = new ClIpPool();
        clIpPool.setIp("2222.3333.444.5555");
        clIpPool.setPort(1111);
        clIpPool.setEnable(true);
        clIpPool.setCreatedTime(new Date());
        crawlerIpPoolService.saveCrawlerIpPool(clIpPool);
    }


    @Test
    public void testCheckExist(){
        boolean b =
crawlerIpPoolService.checkExist("2222.3333.444.555555666555", 1111);
        System.out.println(b);
    }
}
```

1.2 爬虫文章图文附加信息

1.2.1 需求分析

爬虫文章的附加信息，比如文章的点赞，转发，评论量统计，方便后期追踪数据，反向爬取数据

1.2.2 实体类

ClNewsAdditional

```
/**
 * 回复
 */
@Data
public class ClNewsAdditional {
    private Integer id;
    private Integer newsId;
    private String url;
    private Integer readCount;
    private Integer likes;
    private Integer comment;
    private Integer forward;
    private Integer unlikes;
    private Integer collection;
    private Date createdTime;
    private Date count;
    private Date updatedTime;
    private Date nextUpdateTime;
    private Integer updateNum;
}
```

1.2.3 mapper接口定义

ClNewsAdditionalMapper

com.heima.model.mappers.crawerls.ClNewsAdditionalMapper

```java
public interface ClNewsAdditionalMapper {

    int deleteByPrimaryKey(Integer id);

    int insert(ClNewsAdditional record);

    int insertSelective(ClNewsAdditional record);

    ClNewsAdditional selectByPrimaryKey(Integer id);

    int updateByPrimaryKeySelective(ClNewsAdditional record);

    int updateByPrimaryKey(ClNewsAdditional record);

    /**
     * 按条件查询所有数据
     *
     * @param record
     * @return
     */
    List<ClNewsAdditional> selectList(ClNewsAdditional record);

    /**
     * 获取需要更新的数据
     * @return
     */
    List<ClNewsAdditional> selectListByNeedUpdate(Date currentDate);
}
```

ClNewsAdditionalMapper.xml

mappers/crawerls/ClNewsAdditionalMapper.xml

```xml
<?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.heima.model.mappers.crawerls.ClNewsAdditionalMapper">
    <resultMap id="BaseResultMap"
type="com.heima.model.crawler.pojos.ClNewsAdditional">
        <id column="id" property="id"/>
        <result column="news_id" property="newsId"/>
        <result column="url" property="url"/>
        <result column="read_count" property="readCount"/>
        <result column="likes" property="likes"/>
        <result column="comment" property="comment"/>
        <result column="forward" property="forward"/>
        <result column="unlikes" property="unlikes"/>
        <result column="collection" property="collection"/>
        <result column="created_time" property="createdTime"/>
```

```
        <result column="count" property="count"/>
        <result column="updated_time" property="updatedTime"/>
        <result column="update_num" property="updateNum"/>
        <result column="next_update_time" property="nextUpdateTime"/>
    </resultMap>
    <sql id="Base_Column_List">
    id, news_id, url, read_count, likes, comment, forward, unlikes, collection,
created_time,
    count, updated_time,update_num,next_update_time
  </sql>
    <sql id="Base_Column_where">
        <where>
            <if test="newsId!=null and newsId!=''">
                and news_id = #{newsId}
            </if>
            <if test="url!=null and url!=''">
                and url = #{url}
            </if>
            <if test="readCount!=null and readCount!=''">
                and read_count = #{readCount}
            </if>
            <if test="readCount!=null and readCount!=''">
                and read_count = #{readCount}
            </if>
        </where>
    </sql>
    <select id="selectList" resultMap="BaseResultMap">
        select
        <include refid="Base_Column_List"/>
        from cl_news_additional
        <include refid="Base_Column_where"/>
    </select>
    <select id="selectListByNeedUpdate" resultMap="BaseResultMap"
parameterType="java.util.Date">
        select
        <include refid="Base_Column_List"/>
        from cl_news_additional
        <where>
            <![CDATA[
            and update_num < 5
            and next_update_time <= #{currentDate}
            ]]>
        </where>
    </select>
    <select id="selectByPrimaryKey" resultMap="BaseResultMap"
parameterType="java.lang.Integer">
        select
        <include refid="Base_Column_List"/>
        from cl_news_additional
        where id = #{id}
    </select>
    <delete id="deleteByPrimaryKey" parameterType="java.lang.Integer">
    delete from cl_news_additional
    where id = #{id}
  </delete>
    <insert id="insert"
parameterType="com.heima.model.crawler.pojos.ClNewsAdditional"
            keyProperty="id" useGeneratedKeys="true">
```

```
    insert into cl_news_additional (id, news_id, url,
      read_count, likes, comment,
      forward, unlikes, collection,
      created_time, count, updated_time,update_num,next_update_time
      )
    values (#{id}, #{newsId}, #{url},
      #{readCount}, #{likes}, #{comment},
      #{forward}, #{unlikes}, #{collection},
      #{createdTime}, #{count}, #{updatedTime},#{updateNum},#{nextUpdateTime}
      )
  </insert>
    <insert id="insertSelective"
parameterType="com.heima.model.crawler.pojos.ClNewsAdditional"
            keyProperty="id" useGeneratedKeys="true">

        insert into cl_news_additional
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="id != null">
                id,
            </if>
            <if test="newsId != null">
                news_id,
            </if>
            <if test="url != null">
                url,
            </if>
            <if test="readCount != null">
                read_count,
            </if>
            <if test="likes != null">
                likes,
            </if>
            <if test="comment != null">
                comment,
            </if>
            <if test="forward != null">
                forward,
            </if>
            <if test="unlikes != null">
                unlikes,
            </if>
            <if test="collection != null">
                collection,
            </if>
            <if test="createdTime != null">
                created_time,
            </if>
            <if test="count != null">
                count,
            </if>
            <if test="updatedTime != null">
                updated_time,
            </if>
            <if test="updateNum != null">
                update_num,
            </if>

            <if test="nextUpdateTime != null">
```

```
                next_update_time,
            </if>

        </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="id != null">
                #{id},
            </if>
            <if test="newsId != null">
                #{newsId},
            </if>
            <if test="url != null">
                #{url},
            </if>
            <if test="readCount != null">
                #{readCount},
            </if>
            <if test="likes != null">
                #{likes},
            </if>
            <if test="comment != null">
                #{comment},
            </if>
            <if test="forward != null">
                #{forward},
            </if>
            <if test="unlikes != null">
                #{unlikes},
            </if>
            <if test="collection != null">
                #{collection},
            </if>
            <if test="createdTime != null">
                #{createdTime},
            </if>
            <if test="count != null">
                #{count},
            </if>
            <if test="updatedTime != null">
                #{updatedTime},
            </if>
            <if test="updateNum != null">
                #{updateNum},
            </if>
            <if test="nextUpdateTime != null">
                #{nextUpdateTime},
            </if>
        </trim>
    </insert>
    <update id="updateByPrimaryKeySelective"
            parameterType="com.heima.model.crawler.pojos.ClNewsAdditional">
        update cl_news_additional
        <set>
            <if test="newsId != null">
                news_id = #{newsId},
            </if>
            <if test="url != null">
                url = #{url},
```

```
            </if>
            <if test="readCount != null">
                read_count = #{readCount},
            </if>
            <if test="likes != null">
                likes = #{likes},
            </if>
            <if test="comment != null">
                comment = #{comment},
            </if>
            <if test="forward != null">
                forward = #{forward},
            </if>
            <if test="unlikes != null">
                unlikes = #{unlikes},
            </if>
            <if test="collection != null">
                collection = #{collection},
            </if>
            <if test="createdTime != null">
                created_time = #{createdTime},
            </if>
            <if test="count != null">
                count = #{count},
            </if>
            <if test="updatedTime != null">
                updated_time = #{updatedTime},
            </if>
            <if test="updateNum != null">
                update_num = #{updateNum},
            </if>
            <if test="nextUpdateTime != null">
                next_update_time = #{nextUpdateTime},
            </if>
        </set>
        where id = #{id}
    </update>
    <update id="updateByPrimaryKey"
parameterType="com.heima.model.crawler.pojos.ClNewsAdditional">
    update cl_news_additional
    set news_id = #{newsId},
      url = #{url},
      read_count = #{readCount},
      likes = #{likes},
      comment = #{comment},
      forward = #{forward},
      unlikes = #{unlikes},
      collection = #{collection},
      created_time = #{createdTime},
      count = #{count},
      updated_time = #{updatedTime},
       update_num = #{updateNum},
      next_update_time = #{nextUpdateTime}
    where id = #{id}
  </update>
</mapper>
```

1.2.4 service代码

CrawlerNewsAdditionalService

com.heima.crawler.service.CrawlerNewsAdditionalService

```
public interface CrawlerNewsAdditionalService {

    void saveAdditional(ClNewsAdditional clNewsAdditional);

    public List<ClNewsAdditional> queryListByNeedUpdate(Date currentDate);

    List<ClNewsAdditional> queryList(ClNewsAdditional clNewsAdditional);

    public boolean checkExist(String url);

    public ClNewsAdditional getAdditionalByUrl(String url);

    /**
     * 是否是已存在的URL
     *
     * @return
     */
    public boolean isExistsUrl(String url);

    public void updateAdditional(ClNewsAdditional clNewsAdditional);

    public List<ParseItem> toParseItem(List<ClNewsAdditional> additionalList);

    public List<ParseItem> queryIncrementParseItem(Date currentDate);
}
```

CrawlerNewsAdditionalServiceImpl

com.heima.crawler.service.impl.CrawlerNewsAdditionalServiceImpl

```
@Service
public class CrawlerNewsAdditionalServiceImpl implements
CrawlerNewsAdditionalService {

    @Autowired
    private ClNewsAdditionalMapper clNewsAdditionalMapper;

    public List<ClNewsAdditional> queryList(ClNewsAdditional clNewsAdditional) {
        return clNewsAdditionalMapper.selectList(clNewsAdditional);
    }

    /**
     * 获取待更新的数据
     *
     * @return
     */
    public List<ClNewsAdditional> queryListByNeedUpdate(Date currentDate) {
        return clNewsAdditionalMapper.selectListByNeedUpdate(currentDate);
    }

    @Override
```

```
    public ClNewsAdditional getAdditionalByUrl(String url) {
        ClNewsAdditional clNewsAdditional = new ClNewsAdditional();
        clNewsAdditional.setUrl(url);
        List<ClNewsAdditional> additionalList = queryList(clNewsAdditional);
        if (null != additionalList && !additionalList.isEmpty()) {
            return additionalList.get(0);
        }
        return null;
    }


    /**
     * 是否是已存在的URL
     *
     * @return
     */
    public boolean isExistsUrl(String url) {
        boolean isExistsUrl = false;
        if (StringUtils.isNotEmpty(url)) {
            ClNewsAdditional clNewsAdditional = getAdditionalByUrl(url);
            if (null != clNewsAdditional) {
                isExistsUrl = true;
            }
        }
        return isExistsUrl;
    }

    @Override
    public boolean checkExist(String url) {
        ClNewsAdditional clNewsAdditional = new ClNewsAdditional();
        clNewsAdditional.setUrl(url);
        List<ClNewsAdditional> clNewsAdditionalList =
clNewsAdditionalMapper.selectList(clNewsAdditional);
        if (null != clNewsAdditionalList && !clNewsAdditionalList.isEmpty()) {
            return true;
        }
        return false;
    }


    @Override
    public void updateAdditional(ClNewsAdditional clNewsAdditional) {
        clNewsAdditionalMapper.updateByPrimaryKeySelective(clNewsAdditional);
    }

    @Override
    public void saveAdditional(ClNewsAdditional clNewsAdditional) {
        clNewsAdditionalMapper.insertSelective(clNewsAdditional);
    }

    /**
     * 转换为ParseItem
     *
     * @param additionalList
     * @return
     */
    public List<ParseItem> toParseItem(List<ClNewsAdditional> additionalList) {
        List<ParseItem> parseItemList = new ArrayList<ParseItem>();
```

```
        if (null != additionalList && !additionalList.isEmpty()) {
            for (ClNewsAdditional additional : additionalList) {
                ParseItem parseItem = toParseItem(additional);
                if (null != parseItem) {
                    parseItemList.add(parseItem);
                }
            }
        }
        return parseItemList;
    }

    private ParseItem toParseItem(ClNewsAdditional additional) {
        CrawlerParseItem crawlerParseItem = null;
        if (null != additional) {
            crawlerParseItem = new CrawlerParseItem();
            crawlerParseItem.setUrl(additional.getUrl());
        }
        return crawlerParseItem;
    }

    /**
     * 获取增量统计数据
     * @return
     */
    public List<ParseItem> queryIncrementParseItem(Date currentDate) {
        List<ClNewsAdditional> clNewsAdditionalList =
queryListByNeedUpdate(currentDate);
        List<ParseItem> parseItemList = toParseItem(clNewsAdditionalList);
        return parseItemList;
    }
}
```

1.2.5 测试

```
@SpringBootTest
@RunWith(SpringRunner.class)
public class CrawlerNewsAdditionalServiceTest {

    @Autowired
    private CrawlerNewsAdditionalService crawlerNewsAdditionalService;

    @Test
    public void testQueryList(){
        ClNewsAdditional clNewsAdditional = new ClNewsAdditional();

 clNewsAdditional.setUrl("https://blog.csdn.net/weixin_43976602/article/details/
96971651");
        List<ClNewsAdditional> clNewsAdditionals =
crawlerNewsAdditionalService.queryList(clNewsAdditional);
        System.out.println(clNewsAdditionals);
    }

    @Test
    public void testCheckExist(){
        boolean b =
crawlerNewsAdditionalService.checkExist("https://blog.csdn.net/weixin_43976602/a
rticle/details/96971651");
```

```
        System.out.println(b);
    }
}
```

1.3 爬虫文章图文评论信息表

1.3.1 需求分析

保存文章的评论信息

1.3.2 实体类

com.heima.model.crawler.pojos.ClNewsComment

```
/**
 * 文章评论
 */
@Data
public class ClNewsComment implements Serializable {
    /**
     * 主键
     */
    private Integer id;
    /**
     * 文章ID
     */
    private Integer newsId;

    /**
     * 评论人
     */
    private String username;

    /**
     * 评论内容
     */
    private String content;

    /**
     * 评论日期
     */
    private Date commentDate;

    /**
     * 创建日期
     */
    private Date createdDate;
}
```

1.3.3 mapper接口定义

ClNewsCommentMapper

com.heima.model.mappers.crawerls.ClNewsCommentMapper

```
public interface ClNewsCommentMapper {

    int deleteByPrimaryKey(Integer id);

    int insert(ClNewsComment record);

    int insertSelective(ClNewsComment record);

    ClNewsComment selectByPrimaryKey(Integer id);

    int updateByPrimaryKeySelective(ClNewsComment record);

    int updateByPrimaryKey(ClNewsComment record);

    /**
     * 按条件查询所有数据
     *
     * @param record
     * @return
     */
    List<ClNewsComment> selectList(ClNewsComment record);
}
```

ClNewsCommentMapper.xml

mappers/admin/ClNewsCommentMapper.xml

```
<?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.heima.model.mappers.crawerls.ClNewsCommentMapper">
    <resultMap id="BaseResultMap"
type="com.heima.model.crawler.pojos.ClNewsComment">
        <id column="id" property="id"/>
        <result column="news_id" property="newsId"/>
        <result column="username" property="username"/>
        <result column="content" property="content"/>
        <result column="comment_date" property="commentDate"/>
        <result column="created_date" property="createdDate"/>
    </resultMap>
    <sql id="Base_Column_List">
      id,news_id,username,content,comment_date,created_date
    </sql>
    <sql id="Base_Column_where">
        <where>
            <if test="newsId!=null and newsId!=''">
                news_id = #{newsId}
            </if>
```

```
            <if test="username!=null and username!=''">
                username = #{username}
            </if>
            <if test="content!=null and content!=''">
                content = #{content}
            </if>
            <if test="commentDate!=null and commentDate!=''">
                comment_date = #{commentDate}
            </if>
            <if test="createdDate!=null and createdDate!=''">
                created_date = #{createdDate}
            </if>
        </where>
    </sql>
    <select id="selectList" resultMap="BaseResultMap">
        select
        <include refid="Base_Column_List"/>
        from cl_news_comments
        <include refid="Base_Column_where"/>
    </select>
    <select id="selectByPrimaryKey" resultMap="BaseResultMap"
parameterType="java.lang.Integer">
        select
        <include refid="Base_Column_List"/>
        from cl_news_comments
        where id = #{id}
    </select>
    <delete id="deleteByPrimaryKey" parameterType="java.lang.Integer">
        delete from cl_news_comments
        where id = #{id}
    </delete>
    <insert id="insert"
parameterType="com.heima.model.crawler.pojos.ClNewsAdditional">

        insert into cl_news_comments (id, news_id, username,
        content, comment_date, created_time
        )
        values (#{id}, #{newsId}, #{username},
        #{content}, #{commentDate}, #{createTime}
        )
    </insert>
    <insert id="insertSelective"
parameterType="com.heima.model.crawler.pojos.ClNewsAdditional">

        insert into cl_news_comments
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="id != null">
                id,
            </if>
            <if test="newsId != null">
                news_id,
            </if>
            <if test="username != null">
                username,
            </if>
            <if test="content != null">
                content,
            </if>
```

```
            <if test="commentDate != null">
                comment_date,
            </if>
            <if test="createdDate != null">
                created_date,
            </if>
        </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="id != null">
                #{id},
            </if>
            <if test="newsId != null">
                #{newsId},
            </if>
            <if test="username != null">
                #{username},
            </if>
            <if test="content != null">
                #{content},
            </if>
            <if test="commentDate != null">
                #{commentDate},
            </if>
            <if test="createdDate != null">
                #{createdDate},
            </if>
        </trim>
    </insert>
    <update id="updateByPrimaryKeySelective"
            parameterType="com.heima.model.crawler.pojos.ClNewsAdditional">
        update cl_news_comments
        <set>
            <if test="newsId != null">
                news_id = #{newsId},
            </if>
            <if test="username != null">
                username = #{username},
            </if>
            <if test="content != null">
                content = #{content},
            </if>
            <if test="commentDate != null">
                comment_date = #{commentDate},
            </if>
            <if test="createdDate != null">
                created_date = #{createdDate},
            </if>
        </set>
        where id = #{id}
    </update>
    <update id="updateByPrimaryKey"
parameterType="com.heima.model.crawler.pojos.ClNewsAdditional">

        update cl_news_comments
        set news_id = #{newsId},
        username = #{username},
        content = #{content},
        comment_date = #{commentDate},
```

```
        created_date = #{createdDate},
        where id = #{id}
    </update>
</mapper>
```

1.3.4 service代码

CrawlerNewsCommentService

com.heima.crawler.service.CrawlerNewsCommentService

```
public interface CrawlerNewsCommentService {
    public void saveClNewsComment(ClNewsComment clNewsComment);
}
```

ClNewsCommentServiceImpl

com.heima.crawler.service.impl.ClNewsCommentServiceImpl

```
@Service
public class ClNewsCommentServiceImpl implements CrawlerNewsCommentService {
    @Autowired
    private ClNewsCommentMapper clNewsCommentMapper;

    @Override
    public void saveClNewsComment(ClNewsComment clNewsComment) {
        clNewsCommentMapper.insertSelective(clNewsComment);
    }
}
```

1.3.5 测试

1.4 爬虫文章

1.4.1 需求分析

文章的增删改查

1.4.2 实体类

ClNews

com.heima.model.crawler.pojos.ClNews

```
/**
 * 文章
 */
public class ClNews {
    private Integer id;
    private Integer taskId;
    private String title;
    private String name;
```

```
    private int type;
    private Integer channelId;
    private String labels;
    private Date originalTime;
    private Date createdTime;
    private Date submitedTime;
    private Byte status;
    private Date publishTime;
    private String reason;
    private Integer articleId;
    private Integer no;
    private String content;
    private String labelIds;
    public String getUnCompressContent() {
        if (StringUtils.isNotEmpty(content)) {
            return ZipUtils.gunzip(content);
        }
        return content;
    }


    public Integer getId() {
        return id;
    }

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

    public Integer getTaskId() {
        return taskId;
    }

    public void setTaskId(Integer taskId) {
        this.taskId = taskId;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getType() {
        return type;
    }

    public void setType(int type) {
```

```
        this.type = type;
    }

    public Integer getChannelId() {
        return channelId;
    }

    public void setChannelId(Integer channelId) {
        this.channelId = channelId;
    }

    public String getLabels() {
        return labels;
    }

    public void setLabels(String labels) {
        this.labels = labels;
    }

    public Date getOriginalTime() {
        return originalTime;
    }

    public void setOriginalTime(Date originalTime) {
        this.originalTime = originalTime;
    }

    public Date getCreatedTime() {
        return createdTime;
    }

    public void setCreatedTime(Date createdTime) {
        this.createdTime = createdTime;
    }

    public Date getSubmitedTime() {
        return submitedTime;
    }

    public void setSubmitedTime(Date submitedTime) {
        this.submitedTime = submitedTime;
    }

    public Byte getStatus() {
        return status;
    }

    public void setStatus(Byte status) {
        this.status = status;
    }

    public Date getPublishTime() {
        return publishTime;
    }

    public void setPublishTime(Date publishTime) {
        this.publishTime = publishTime;
    }
```

```
    public String getReason() {
        return reason;
    }

    public void setReason(String reason) {
        this.reason = reason;
    }

    public Integer getArticleId() {
        return articleId;
    }

    public void setArticleId(Integer articleId) {
        this.articleId = articleId;
    }

    public Integer getNo() {
        return no;
    }

    public void setNo(Integer no) {
        this.no = no;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    public String getLabelIds() {
        return labelIds;
    }

    public void setLabelIds(String labelIds) {
        this.labelIds = labelIds;
    }
}
```

1.4.3 mapper接口定义

ClNewsMapper

com.heima.model.mappers.crawerls.ClNewsMapper

```
public interface ClNewsMapper {

    int deleteByPrimaryKey(Integer id);

    int insert(ClNews record);

    int insertSelective(ClNews record);

    ClNews selectByPrimaryKey(Integer id);
```

```
    int updateByPrimaryKeySelective(ClNews record);

    int updateStatus(ClNews record);

    int updateByPrimaryKeyWithBLOBs(ClNews record);

    int updateByPrimaryKey(ClNews record);
    /**
     * 按条件查询所有数据
     *
     * @param record
     * @return
     */
    List<ClNews> selectList(ClNews record);

    void deleteByUrl(String url);

    ClNews selectByIdAndStatus(ClNews param);
}
```

ClNewsMapper.xml

mappers/crawerls/ClNewsMapper.xml

```
<?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.heima.model.mappers.crawerls.ClNewsMapper">
    <resultMap id="BaseResultMap" type="com.heima.model.crawler.pojos.ClNews">
        <id column="id" property="id"/>
        <result column="task_id" property="taskId"/>
        <result column="title" property="title"/>
        <result column="name" property="name"/>
        <result column="type" property="type"/>
        <result column="channel_id" property="channelId"/>
        <result column="labels" property="labels"/>
        <result column="label_ids" property="labelIds"/>
        <result column="original_time" property="originalTime"/>
        <result column="created_time" property="createdTime"/>
        <result column="submited_time" property="submitedTime"/>
        <result column="status" property="status"/>
        <result column="publish_time" property="publishTime"/>
        <result column="reason" property="reason"/>
        <result column="article_id" property="articleId"/>
        <result column="no" property="no"/>
    </resultMap>
    <resultMap id="ResultMapWithBLOBs"
type="com.heima.model.crawler.pojos.ClNews"
               extends="BaseResultMap">
        <result column="content" property="content" jdbcType="LONGVARCHAR"/>
    </resultMap>
    <sql id="Base_Column_List">
    id, task_id, title, name, type, channel_id, labels,label_ids original_time,
created_time,
    submited_time, status, publish_time, reason, article_id, no
  </sql>
```

```
    <sql id="Blob_Column_List">
    ,content
  </sql>
    <sql id="Base_Column_where">
        <where>
            <if test="id!=null and id!=''">
                and id = #{id}
            </if>
            <if test="title!=null and title!=''">
                and title = #{title}
            </if>
            <if test="name!=null and name!=''">
                and name = #{name}
            </if>
            <if test="type!=null and type!=''">
                and type = #{type}
            </if>
            <if test="status!=null and status!=''">
                and status = #{status}
            </if>
        </where>
    </sql>
    <select id="selectList" resultMap="ResultMapWithBLOBs">
        select
        <include refid="Base_Column_List"/>
        <include refid="Blob_Column_List"/>
        from cl_news
        <include refid="Base_Column_where"/>
    </select>
    <select id="selectByIdAndStatus" resultMap="ResultMapWithBLOBs">
        select
        <include refid="Base_Column_List"/>
        <include refid="Blob_Column_List"/>
        from cl_news
        <include refid="Base_Column_where"/>
    </select>
    <select id="selectByPrimaryKey" resultMap="ResultMapWithBLOBs"
parameterType="java.lang.Integer">
        select
        <include refid="Base_Column_List"/>
        ,
        <include refid="Blob_Column_List"/>
        from cl_news
        where id = #{id}
    </select>
    <delete id="deleteByPrimaryKey" parameterType="java.lang.Integer">
    delete from cl_news
    where url = #{id}
  </delete>
    <delete id="deleteByUrl" parameterType="java.lang.String">
    delete from cl_news
    where id = #{url}
  </delete>
    <insert id="insert" parameterType="com.heima.model.crawler.pojos.ClNews"
useGeneratedKeys="true"
            keyProperty="id">
    insert into cl_news (id, task_id,title,
      name, type, channel_id,
```

```
      labels,label_ids, original_time, created_time,
      submited_time, status, publish_time,
      reason, article_id, no,
      content)
    values (#{id}, #{taskId},#{title},
      #{name}, #{type}, #{channelId},
      #{labels}, #{labelIds}, #{originalTime}, #{createdTime},
      #{submitedTime}, #{status,jdbcType=TINYINT}, #{publishTime},
      #{reason}, #{articleId}, #{no},
      #{content,jdbcType=LONGVARCHAR})
  </insert>
    <insert id="insertSelective"
parameterType="com.heima.model.crawler.pojos.ClNews"
            keyProperty="id" useGeneratedKeys="true">
        insert into cl_news
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="id != null">
                id,
            </if>
            <if test="taskId != null">
                task_id,
            </if>
            <if test="title != null">
                title,
            </if>
            <if test="name != null">
                name,
            </if>
            <if test="type != null">
                type,
            </if>
            <if test="channelId != null">
                channel_id,
            </if>
            <if test="labels != null">
                labels,
            </if>
            <if test="labelIds != null">
                label_ids,
            </if>
            <if test="originalTime != null">
                original_time,
            </if>
            <if test="createdTime != null">
                created_time,
            </if>
            <if test="submitedTime != null">
                submited_time,
            </if>
            <if test="status != null">
                status,
            </if>
            <if test="publishTime != null">
                publish_time,
            </if>
            <if test="reason != null">
                reason,
            </if>
```

```
            <if test="articleId != null">
                article_id,
            </if>
            <if test="no != null">
                no,
            </if>
            <if test="content != null">
                content,
            </if>
        </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="id != null">
                #{id},
            </if>
            <if test="taskId != null">
                #{taskId},
            </if>

            <if test="title != null">
                #{title},
            </if>
            <if test="name != null">
                #{name},
            </if>
            <if test="type != null">
                #{type},
            </if>
            <if test="channelId != null">
                #{channelId},
            </if>
            <if test="labels != null">
                #{labels},
            </if>
            <if test="labelIds != null">
                #{labelIds},
            </if>
            <if test="originalTime != null">
                #{originalTime},
            </if>
            <if test="createdTime != null">
                #{createdTime},
            </if>
            <if test="submitedTime != null">
                #{submitedTime},
            </if>
            <if test="status != null">
                #{status,jdbcType=TINYINT},
            </if>
            <if test="publishTime != null">
                #{publishTime},
            </if>
            <if test="reason != null">
                #{reason},
            </if>
            <if test="articleId != null">
                #{articleId},
            </if>
            <if test="no != null">
```

```
                #{no},
            </if>
            <if test="content != null">
                #{content,jdbcType=LONGVARCHAR},
            </if>
        </trim>
    </insert>
    <update id="updateStatus"
parameterType="com.heima.model.crawler.pojos.ClNews">
        update cl_news
        <set>
            <if test="submitedTime != null">
                submited_time = #{submitedTime},
            </if>
            <if test="status != null">
                status = #{status,jdbcType=TINYINT},
            </if>
            <if test="publishTime != null">
                publish_time = #{publishTime},
            </if>
            <if test="reason != null">
                reason = #{reason},
            </if>
            <if test="no != null">
                no = #{no},
            </if>
        </set>
        where id = #{id}
    </update>
    <update id="updateByPrimaryKeySelective"
parameterType="com.heima.model.crawler.pojos.ClNews">
        update cl_news
        <set>
            <if test="taskId != null">
                task_id = #{taskId},
            </if>
            <if test="title != null">
                title = #{title},
            </if>
            <if test="name != null">
                name = #{name},
            </if>
            <if test="type != null">
                type = #{type},
            </if>
            <if test="channelId != null">
                channel_id = #{channelId},
            </if>
            <if test="labels != null">
                labels = #{labels},
            </if>
            <if test="labelIds != null">
                label_ids = #{labelIds},
            </if>
            <if test="originalTime != null">
                original_time = #{originalTime},
            </if>
            <if test="createdTime != null">
```

```
                created_time = #{createdTime},
            </if>
            <if test="submitedTime != null">
                submited_time = #{submitedTime},
            </if>
            <if test="status != null">
                status = #{status,jdbcType=TINYINT},
            </if>
            <if test="publishTime != null">
                publish_time = #{publishTime},
            </if>
            <if test="reason != null">
                reason = #{reason},
            </if>
            <if test="articleId != null">
                article_id = #{articleId},
            </if>
            <if test="no != null">
                no = #{no},
            </if>
            <if test="content != null">
                content = #{content,jdbcType=LONGVARCHAR},
            </if>
        </set>
        where id = #{id}
    </update>
    <update id="updateByPrimaryKeyWithBLOBs"
parameterType="com.heima.model.crawler.pojos.ClNews">
    update cl_news
    set task_id = #{taskId},
      title = #{title},
      name = #{name},
      type = #{type},
      channel_id = #{channelId},
      labels = #{labels},
      label_ids = #{labelIds},
      original_time = #{originalTime},
      created_time = #{createdTime},
      submited_time = #{submitedTime},
      status = #{status,jdbcType=TINYINT},
      publish_time = #{publishTime},
      reason = #{reason},
      article_id = #{articleId},
      no = #{no},
      content = #{content,jdbcType=LONGVARCHAR}
    where id = #{id}
  </update>
    <update id="updateByPrimaryKey"
parameterType="com.heima.model.crawler.pojos.ClNews">
    update cl_news
    set task_id = #{taskId},
      url = #{url},
      title = #{title},
      name = #{name},
      type = #{type},
      channel_id = #{channelId},
      labels = #{labels},
      label_ids = #{labelIds},
```

```
      reading_num = #{readingNum},
      original_time = #{originalTime},
      created_time = #{createdTime},
      submited_time = #{submitedTime},
      status = #{status,jdbcType=TINYINT},
      publish_time = #{publishTime},
      reason = #{reason},
      article_id = #{articleId},
      no = #{no}
    where id = #{id}
  </update>
</mapper>
```

1.4.4 service代码

CrawlerNewsService

com.heima.crawler.service.CrawlerNewsService

```
public interface CrawlerNewsService {
    public void saveNews(ClNews clNews);
    public void updateNews(ClNews clNews);
    public void deleteByUrl(String url);
    public List<ClNews> queryList(ClNews clNews);
}
```

CrawlerNewsServiceImp

com.heima.crawller.service.impl.CrawlerNewsServiceImpl

```
@Service
public class CrawlerNewsServiceImpl implements CrawlerNewsService {
    @Autowired
    private ClNewsMapper clNewsMapper;
    @Override
    public void saveNews(ClNews clNews) {
        clNewsMapper.insertSelective(clNews);
    }
    @Override
    public void deleteByUrl(String url) {
        clNewsMapper.deleteByUrl(url);
    }
    @Override
    public List<ClNews> queryList(ClNews clNews) {
        return clNewsMapper.selectList(clNews);
    }
    @Override
    public void updateNews(ClNews clNews) {
        clNewsMapper.updateByPrimaryKey(clNews);
    }
}
```

1.4.5 测试

2 排重

> 下载后的内容需要进行排重，一个URL不能重复下载，这里用到了redis排重分两步
>
> - redis排重
> - 数据库排重

2.1集成redis

(1)heima-leadnews-common模块集成redis

redis.properties

```
#redis config
spring.redis.host=127.0.0.1
spring.redis.port=6379
spring.redis.password=123456
spring.redis.timeout=90000
spring.redis.lettuce.pool.max-active=8
spring.redis.lettuce.pool.max-idle=8
spring.redis.lettuce.pool.max-wait=-1
spring.redis.lettuce.pool.min-idle=0
```

创建配置类：com.heima.common.redis.RedisConfiguration

```
@Configuration
@ConfigurationProperties(prefix = "spring.redis")
@PropertySource("classpath:redis.properties")
public class RedisConfiguration extends RedisAutoConfiguration {
}
```

(2)heima-leadnews-crawler微服务中引入redis的配置

创建配置类：com.heima.crawler.config.RedisConfig

```
@Configuration
@ComponentScan("com.heima.common.redis")
public class RedisConfig {
}
```

2.2 DbAndRedisScheduler类

> 该类的主要作用是为了防重复

```
/**
 * URL防重复
 */
@Log4j2
public class DbAndRedisScheduler extends RedisScheduler implements ProcessFlow {
```

```
    @Autowired
    private CrawlerHelper crawlerHelper;

    @Autowired
    private CrawlerNewsAdditionalService crawlerNewsAdditionalService;

    public DbAndRedisScheduler(String host) {
        super(host);
    }

    public DbAndRedisScheduler(JedisPool pool) {
        super(pool);
    }

    /**
     * 是否重复
     * @param request request请求
     * @param task 任务
     * @return
     */
    @Override
    public boolean isDuplicate(Request request, Task task) {
        String handelType = crawlerHelper.getHandelType(request);
        boolean isExist = false;
        //正向统计才尽心排重
        if (CrawlerEnum.HandelType.FORWARD.name().equals(handelType)) {
            log.info("URL排重开始，URL:{},documentType:{}", request.getUrl(),
handelType);
            isExist = super.isDuplicate(request, task);
            if (!isExist) {
                isExist =
crawlerNewsAdditionalService.isExistsUrl(request.getUrl());
            }
            log.info("URL排重结束，URL:{}，handelType:{},isExist：{}",
request.getUrl(), handelType, isExist);
        } else {
            log.info("反向抓取，不进行URL排重");
        }
        return isExist;
    }


    @Override
    public void handel(ProcessFlowData processFlowData) {

    }

    @Override
    public CrawlerEnum.ComponentType getComponentType() {
        return CrawlerEnum.ComponentType.SCHEDULER;
    }

    @Override
    public int getPriority() {
        return 123;
    }
}
```

交给spring管理,在CrawlerConfig配置中配置初始化当前类

```
@Value("${redis.host}")
private String redisHost;
@Value("${redis.port}")
private int reidsPort;
@Value("${redis.timeout}")
private int reidstimeout;
@Value("${redis.password}")
private String reidsPassword;

@Bean
public DbAndRedisScheduler getDbAndRedisScheduler() {
    GenericObjectPoolConfig genericObjectPoolConfig = new
GenericObjectPoolConfig();
    JedisPool jedisPool = new JedisPool(genericObjectPoolConfig, redisHost,
reidsPort, reidstimeout, null, 0);
    return new DbAndRedisScheduler(jedisPool);
}
```

2.3 测试

直接使用ProcessingFlowManagerTest这个类测试即可

3 解析模块

3.1 前置数据

3.1.1 HtmlStyle类

com.heima.model.crawler.core.label.HtmlStyle

```
public class HtmlStyle {
    private Map<String, String> styleMap = new HashMap<>();

    public void addStyle(String key, String value) {
        styleMap.put(key, value);
    }

    public void addStyle(Map<String, String> map) {
        styleMap.putAll(map);
    }

    public String getCssStyle() {
        StringBuilder sb = new StringBuilder();
        for (Map.Entry<String, String> entry : styleMap.entrySet()) {

 sb.append(entry.getKey()).append(":'").append(entry.getValue()).append("',");
        }
        return StringUtils.removeEnd(sb.toString(), ",");
    }

    public Map<String, String> getStyleMap() {
        return styleMap;
```

```
    }

    public void setStyleMap(Map<String, String> styleMap) {
        this.styleMap = styleMap;
    }
}
```

3.1.2 AbstractHtmlParsePipeline抽象类

com.heima.crawler.process.parse.AbstractHtmlParsePipeline

```
/**
 * Html 解析抽抽象类，定义了公用的方法以及抽象模板
 * <p>
 * <p>
 * Pipeline负责抽取结果的处理，包括计算、持久化到文件、数据库等。WebMagic默认提供了“输出到控制台”和“保存到文件”两种结果处理方案。
 * <p>
 * Pipeline定义了结果保存的方式，如果你要保存到指定数据库，则需要编写对应的Pipeline。对于一类需求一般只需编写一个Pipeline。
 *
 * @param <T>
 */
@Log4j2
public abstract class AbstractHtmlParsePipeline<T> extends AbstractProcessFlow
implements Pipeline {

    @Autowired
    private CrawlerHelper crawlerHelper;


    public void handel(ProcessFlowData processFlowData) {

    }


    /**
     * 这里传入的是处理好的对象
     * 对结构数据进行清洗以及存储
     * 是 Pipeline的主要入口方法
     *
     * @param resultItems ResultItems保存了抽取结果，它是一个Map结构，
     * @param task
     */
    @Override
    public void process(ResultItems resultItems, Task task) {
        long currentTime = System.currentTimeMillis();
        String url = resultItems.getRequest().getUrl();
        String documentType =
crawlerHelper.getDocumentType(resultItems.getRequest());
        String handelType =
crawlerHelper.getHandelType(resultItems.getRequest());
        log.info("开始解析抽取后的数据，url:{},handelType:{}", url, handelType);
        if (!CrawlerEnum.DocumentType.PAGE.name().equals(documentType)) {
```

```
            log.error("不符合的文档类型，url:{},documentType:{},handelType:{}",
url, documentType, handelType);
            return;
        }
        ParseItem parseItem =
crawlerHelper.getParseItem(resultItems.getRequest());
        if (null != parseItem && StringUtils.isNotEmpty(url)) {
            Map<String, Object> grabParameterMap = resultItems.getAll();
            preParameterHandel(grabParameterMap);
            if (url.equals(parseItem.getInitialUrl())) {
                //通过泛着进行设置属性
                ReflectUtils.setPropertie(parseItem, grabParameterMap, true);
                //设置处理类型

 parseItem.setHandelType(crawlerHelper.getHandelType(resultItems.getRequest()));
                handelHtmlData((T) parseItem);
            }
        }
        log.info("解析抽取后的数据完成，url:{},handelType:{},耗时：{}", url,
handelType, System.currentTimeMillis() - currentTime);
    }

    /**
     * 前置参数处理
     */
    public abstract void preParameterHandel(Map<String, Object> parameter);

    /**
     * html 数据处理以及清洗存储
     *
     * @param t
     */
    public abstract void handelHtmlData(T t);


    /**
     * 获取解析表达式
     *
     * @return
     */
    public String getParseExpression() {
        return "p,pre,h1,h2,h3,h4,h5";
    }

    /**
     * 获取默认的Html 样式
     *
     * @return
     */
    public Map<String, HtmlStyle> getDefHtmlStyleMap() {
        Map<String, HtmlStyle> styleMap = new HashMap<String, HtmlStyle>();
        //h1 数据添加
        HtmlStyle h1Style = new HtmlStyle();
        h1Style.addStyle("font-size", "22px");
        h1Style.addStyle("line-height", "24px");
        styleMap.put("h1", h1Style);
        //h2 数据添加
        HtmlStyle h2Style = new HtmlStyle();
```

```
        h2Style.addStyle("font-size", "18px");
        h2Style.addStyle("line-height", "20px");
        styleMap.put("h2", h2Style);
        //h3 数据添加
        HtmlStyle h3Style = new HtmlStyle();
        h3Style.addStyle("font-size", "16px");
        h3Style.addStyle("line-height", "18px");
        styleMap.put("h3", h3Style);
        //h4 数据添加
        HtmlStyle h4Style = new HtmlStyle();
        h4Style.addStyle("font-size", "14px");
        h4Style.addStyle("line-height", "16px");
        styleMap.put("h4", h4Style);
        //h5 数据添加
        HtmlStyle h5Style = new HtmlStyle();
        h5Style.addStyle("font-size", "12px");
        h5Style.addStyle("line-height", "14px");
        styleMap.put("h5", h5Style);
        //h6 数据添加
        HtmlStyle h6Style = new HtmlStyle();
        h6Style.addStyle("font-size", "10px");
        h6Style.addStyle("line-height", "12px");
        styleMap.put("h6", h6Style);
        return styleMap;
    }

    /**
     * 获取组件类型
     *
     * @return
     */
    @Override
    public CrawlerEnum.ComponentType getComponentType() {
        return CrawlerEnum.ComponentType.PIPELINE;
    }
}
```

3.2 线程池操作类CrawlerThreadPool

com.heima.crawler.process.thread.CrawlerThreadPool

```
/**
 * 线程池处理类
 */
@Log4j2
public class CrawlerThreadPool {
    /**
     * 线程池最大连接数 IO密集型 2n+1
     */

    private static final int threadNum =
Runtime.getRuntime().availableProcessors();
    /**
     * 创建一个阻塞队列
     */
```

```
    private static final ArrayBlockingQueue queue = new
ArrayBlockingQueue<Runnable>(10000);
    /**
     * 创建一个线程池
     * 核心线程数 1
     * 最大线程数 2n+1
     * 等待超时时间60秒
     * 阻塞队列大小 10000
     */
    private static final ExecutorService executorService = new
ThreadPoolExecutor(1, threadNum,
            60L, TimeUnit.SECONDS, queue) {

        @Override
        protected void beforeExecute(Thread t, Runnable r) {
            log.info("线程池开始执行任务，threadName:{},线程池堆积数量：{}",
t.getName(), queue.size());
        }

        @Override
        protected void afterExecute(Runnable r, Throwable t) {
            log.info("线程池开始执行完成");
            if (null != t) {
                log.error(t.getLocalizedMessage());
            }
        }
    };


    /**
     * 提交一个线程
     *
     * @param runnable
     */
    public static void submit(Runnable runnable) {
        log.info("线程池添加任务,线程池堆积任务数量：{},最大线程数:{}", queue.size(),
threadNum);
        executorService.execute(runnable);
    }


}
```

3.3 Gzip压缩工具类 ZipUtils

com.heima.model.crawler.core.parse.ZipUtils

```
/**
 * 字符串压缩
 */
public class ZipUtils {
```

```
    /**
     * 使用gzip进行压缩
     */
    public static String gzip(String primStr) {
        if (primStr == null || primStr.length() == 0) {
            return primStr;
        }


        ByteArrayOutputStream out = new ByteArrayOutputStream();


        GZIPOutputStream gzip = null;
        try {
            gzip = new GZIPOutputStream(out);
            gzip.write(primStr.getBytes());
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (gzip != null) {
                try {
                    gzip.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return new sun.misc.BASE64Encoder().encode(out.toByteArray());
    }


    /**
     * <p>
     * Description:使用gzip进行解压缩
     * </p>
     *
     * @param compressedStr
     * @return
     */
    public static String gunzip(String compressedStr) {
        if (compressedStr == null) {
            return null;
        }


        ByteArrayOutputStream out = new ByteArrayOutputStream();
        ByteArrayInputStream in = null;
        GZIPInputStream ginzip = null;
        byte[] compressed = null;
        String decompressed = null;
        try {
            compressed = new
sun.misc.BASE64Decoder().decodeBuffer(compressedStr);
            in = new ByteArrayInputStream(compressed);
            ginzip = new GZIPInputStream(in);


            byte[] buffer = new byte[1024];
```

```
            int offset = -1;
            while ((offset = ginzip.read(buffer)) != -1) {
                out.write(buffer, 0, offset);
            }
            decompressed = out.toString();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (ginzip != null) {
                try {
                    ginzip.close();
                } catch (IOException e) {
                }
            }
            if (in != null) {
                try {
                    in.close();
                } catch (IOException e) {
                }
            }
            if (out != null) {
                try {
                    out.close();
                } catch (IOException e) {
                }
            }
        }


        return decompressed;
    }


    /**
     * 使用zip进行压缩
     *
     * @param str 压缩前的文本
     * @return 返回压缩后的文本
     */
    public static final String zip(String str) {
        if (str == null)
            return null;
        byte[] compressed;
        ByteArrayOutputStream out = null;
        ZipOutputStream zout = null;
        String compressedStr = null;
        try {
            out = new ByteArrayOutputStream();
            zout = new ZipOutputStream(out);
            zout.putNextEntry(new ZipEntry("0"));
            zout.write(str.getBytes());
            zout.closeEntry();
            compressed = out.toByteArray();
            compressedStr = new
sun.misc.BASE64Encoder().encodeBuffer(compressed);
        } catch (IOException e) {
            compressed = null;
        } finally {
```

```
            if (zout != null) {
                try {
                    zout.close();
                } catch (IOException e) {
                }
            }
            if (out != null) {
                try {
                    out.close();
                } catch (IOException e) {
                }
            }
        }
        return compressedStr;
    }


    /**
     * 使用zip进行解压缩
     *
     * @param compressedStr 压缩后的文本
     * @return 解压后的字符串
     */
    public static final String unzip(String compressedStr) {
        if (compressedStr == null) {
            return null;
        }


        ByteArrayOutputStream out = null;
        ByteArrayInputStream in = null;
        ZipInputStream zin = null;
        String decompressed = null;
        try {
            byte[] compressed = new
sun.misc.BASE64Decoder().decodeBuffer(compressedStr);
            out = new ByteArrayOutputStream();
            in = new ByteArrayInputStream(compressed);
            zin = new ZipInputStream(in);
            zin.getNextEntry();
            byte[] buffer = new byte[1024];
            int offset = -1;
            while ((offset = zin.read(buffer)) != -1) {
                out.write(buffer, 0, offset);
            }
            decompressed = out.toString();
        } catch (IOException e) {
            decompressed = null;
        } finally {
            if (zin != null) {
                try {
                    zin.close();
                } catch (IOException e) {
                }
            }
            if (in != null) {
                try {
                    in.close();
```

```
                } catch (IOException e) {
                }
            }
            if (out != null) {
                try {
                    out.close();
                } catch (IOException e) {
                }
            }
        }
        return decompressed;
    }
}
```

3.4 Html解析工具类 HtmlParser 类

com.heima.crawler.utils.HtmlParser

```
/**
 * Html解析工具
 * 将html转换为特定格式
 */
@Log4j2
public class HtmlParser {

    /**
     * 需要处理的Html 标签
     */
    private final CrawlerEnum.HtmlType[] specialHtmlTypeArray =
            new CrawlerEnum.HtmlType[]{
                    CrawlerEnum.HtmlType.A_TAG,
                    CrawlerEnum.HtmlType.CODE_TAG,
                    CrawlerEnum.HtmlType.H1_TAG,
                    CrawlerEnum.HtmlType.H2_TAG,
                    CrawlerEnum.HtmlType.H3_TAG,
                    CrawlerEnum.HtmlType.H4_TAG,
                    CrawlerEnum.HtmlType.H5_TAG};

    /**
     * 默认css 设置
     */
    private Map<String, HtmlStyle> defaultStyleMap = null;

    /**
     * css 表达式
     */
    private String cssExpression = null;


    public static HtmlParser getHtmlParser(String cssExpression, Map<String,
HtmlStyle> defaultStyleMap) {
        return new HtmlParser(cssExpression, defaultStyleMap);
    }
```

```
    public HtmlParser(String cssExpression, Map<String, HtmlStyle>
defaultStyleMap) {
        this.cssExpression = cssExpression;
        this.defaultStyleMap = defaultStyleMap;
    }

    /**
     * 解析html 内容
     *
     * @param content
     * @return
     */
    public List<HtmlLabel> parseHtml(String content) {
        long currentTime = System.currentTimeMillis();
        log.info("开始解析文章内容");
        Document document = Jsoup.parse(content);
        Elements elements = document.select(cssExpression);
        List<HtmlLabel> htmlLabelList = parseElements(elements);
        log.info("解析文章内容完成，耗时：{}", System.currentTimeMillis() -
currentTime);
        return htmlLabelList;
    }


    /**
     * 解析Html内容并转换为json数据
     *
     * @param content
     * @return
     */
    public String parserHtmlToJson(String content) {
        List<HtmlLabel> htmlLabelList = parseHtml(content);
        return JSON.toJSONString(htmlLabelList);
    }

    /**
     * 解析Html dom树
     *
     * @param elements
     * @return
     */
    private List<HtmlLabel> parseElements(Elements elements) {
        List<HtmlLabel> htmlLabelList = new ArrayList<HtmlLabel>();
        elements.forEach(new Consumer<Element>() {
            @Override
            public void accept(Element element) {
                List<HtmlLabel> labelList = parserElement(element, new
ParserCallBack() {
                    @Override
                    public void callBack(Elements elements) {
                        parseElements(elements);
                    }
                });
                htmlLabelList.addAll(labelList);
            }
        });
        return htmlLabelList;
    }
```

```
    /**
     * 解析Html 元素
     *
     * @param element
     * @param callBack
     * @return
     */
    private List<HtmlLabel> parserElement(Element element, ParserCallBack
callBack) {
        List<HtmlLabel> htmlLabelList = new ArrayList<HtmlLabel>();
        //校验元素是否需要处理
        if (isNeedHandel(element)) {
            HtmlLabel htmlLabel = parseNodeByByElement(element);
            htmlLabelList.add(htmlLabel);
        } else {
            //获取元素所有的子节点
            List<Node> childNodes = element.childNodes();
            //解析节点列表
            List<HtmlLabel> list = parserNodeList(childNodes, callBack);
            htmlLabelList.addAll(list);
        }
        return htmlLabelList;
    }

    /**
     * 解析 Html 节点列表
     *
     * @param nodeList
     * @param callBack
     * @return
     */
    private List<HtmlLabel> parserNodeList(List<Node> nodeList, ParserCallBack
callBack) {
        List<HtmlLabel> htmlLabelList = new ArrayList<HtmlLabel>();
        if (null != nodeList && !nodeList.isEmpty()) {
            List<Element> elementList = new ArrayList<Element>();
            for (Node node : nodeList) {
                //检查节点是否需要处理
                if (isNeedHandel(node)) {
                    //解析node节点
                    HtmlLabel htmlLabel = parseNode(node);
                    if (null != htmlLabel) {
                        htmlLabelList.add(htmlLabel);
                    }
                } else {
                    //如果不需要处理并且是element元素就加入list中交给回调方法进行递归处理
                    if (node instanceof Element) {
                        elementList.add((Element) node);
                    }
                }
            }
            if (!elementList.isEmpty()) {
                //调用回调方法进行递归处理
                callBack.callBack(new Elements(elementList));
            }
        }
        return htmlLabelList;
```

```
    }

    /**
     * 解析Html 节点
     *
     * @param node
     * @return
     */
    private HtmlLabel parseNode(Node node) {
        HtmlLabel htmlLabel = null;
        if (null != node) {
            if (node instanceof TextNode) {
                htmlLabel = parseNodeByTextNode((TextNode) node);
            } else if (node instanceof Element) {
                htmlLabel = parseNodeByByElement((Element) node);
            }
        }
        return htmlLabel;
    }

    /**
     * 获取文本类型的数据节点
     *
     * @param textNode
     * @return
     */
    private HtmlLabel parseNodeByTextNode(TextNode textNode) {
        HtmlLabel htmlLabel = null;
        if (null != textNode) {
            String text = textNode.getWholeText();
            if (StringUtils.isNotBlank(text)) {
                htmlLabel = new HtmlLabel();
                htmlLabel.setValue(textNode.getWholeText());
                htmlLabel.setType("text");
            }
        }
        return htmlLabel;
    }

    /**
     * 解析Element 元素
     *
     * @param element
     * @return
     */
    private HtmlLabel parseNodeByByElement(Element element) {
        HtmlLabel htmlLabel = null;
        if (null != element) {
            String tagName = element.tagName();
            if (CrawlerEnum.HtmlType.A_TAG.getLabelName().equals(tagName)) {
                // explanLabel = getExplanLabelByaLink(element);
            } else if
(CrawlerEnum.HtmlType.IMG_TAG.getLabelName().equals(tagName)) {
                htmlLabel = parseNodeByImage(element);
            } else if
(CrawlerEnum.HtmlType.CODE_TAG.getLabelName().equals(tagName)) {
                htmlLabel = parseNodeByCode(element);
            } else {
```

```
                htmlLabel = parseNodeByOther(element);
            }
        }
        return htmlLabel;
    }


    /**
     * 获取A 标签的链接
     *
     * @param element
     * @return
     */
    private HtmlLabel parseNodeByByaLink(Element element) {
        HtmlLabel htmlLabel = null;
        if (null != element) {
            String link = element.attr("href");
            String text = element.ownText();
            htmlLabel = new HtmlLabel();
            htmlLabel.setValue(text);
            // explanLabel.setLink(link);
            htmlLabel.setType(CrawlerEnum.HtmlType.A_TAG.getDataType());
        }
        return htmlLabel;
    }

    /**
     * 获取图片的信息
     *
     * @param element
     * @return
     */
    private HtmlLabel parseNodeByImage(Element element) {
        HtmlLabel htmlLabel = null;
        if (null != element) {
            String src = element.attr("src");
            src = imageUrlHandel(src);
            String width = element.attr("width");
            String height = element.attr("height");
            htmlLabel = new HtmlLabel();
            HtmlStyle htmlStyle = new HtmlStyle();
            htmlStyle.addStyle("width", width + "px");
            htmlStyle.addStyle("height", height + "px");
            htmlLabel.setValue(src);
            htmlLabel.setStyle(htmlStyle.getCssStyle());
            htmlLabel.setType(CrawlerEnum.HtmlType.IMG_TAG.getDataType());
        }
        return htmlLabel;
    }

    /**
     * 处理图片URL带参数问题
     *
     * @param src
     * @return
     */
    private String imageUrlHandel(String src) {
        if (StringUtils.isNotEmpty(src) && src.contains("?")) {
```

```
            src = src.substring(0, src.indexOf("?"));
        }
        return src;
    }

    /**
     * 获取代码的数据
     *
     * @param element
     * @return
     */
    private HtmlLabel parseNodeByCode(Element element) {
        HtmlLabel htmlLabel = null;
        if (null != element) {
            String text = element.ownText();
            htmlLabel = new HtmlLabel();
            htmlLabel.setValue(text);
            htmlLabel.setType(CrawlerEnum.HtmlType.CODE_TAG.getDataType());
        }
        return htmlLabel;
    }

    /**
     * 其他数据处理
     *
     * @param element
     * @return
     */
    private HtmlLabel parseNodeByOther(Element element) {
        HtmlLabel htmlLabel = null;
        if (null != element) {
            HtmlStyle htmlStyle = defaultStyleMap.get(element.tagName());
            String text = element.ownText();
            htmlLabel = new HtmlLabel();
            htmlLabel.setValue(text);
            htmlLabel.setType("text");
            if (null != htmlStyle) {
                htmlLabel.setStyle(htmlStyle.getCssStyle());
            }

        }
        return htmlLabel;
    }

    /**
     * 检查节点否需要处理
     *
     * @return
     */
    private boolean isNeedHandel(Node node) {
        boolean flag = false;
        //没有子节点
        if (node.childNodes().isEmpty()) {
            flag = true;
        } else {
            if (null != node && node instanceof Element) {
                Element element = (Element) node;
                flag = isNeedHandel(element);
```

```
            }
        }
        return flag;
    }

    /**
     * 校验是否需要进行处理
     *
     * @param element
     * @return
     */
    private boolean isNeedHandel(Element element) {
        boolean flag = false;
        if (null != element) {
            String tagName = element.tagName();
            for (CrawlerEnum.HtmlType htmlType : specialHtmlTypeArray) {
                if
(htmlType.getLabelName().toLowerCase().equals(tagName.toLowerCase())) {
                    flag = true;
                    break;
                }
            }
        }
        return flag;
    }


    /**
     * 内部接口
     */
    private interface ParserCallBack {
        void callBack(Elements elements);
    }

    public Map<String, HtmlStyle> getDefaultStyleMap() {
        return defaultStyleMap;
    }

    public void setDefaultStyleMap(Map<String, HtmlStyle> defaultStyleMap) {
        this.defaultStyleMap = defaultStyleMap;
    }

    public String getCssExpression() {
        return cssExpression;
    }

    public void setCssExpression(String cssExpression) {
        this.cssExpression = cssExpression;
    }
}
```

3.5 反射工具类 ReflectUtils类

com.heima.utils.common.ReflectUtils

```
public class ReflectUtils {


    /**
     * 转换为Map
     *
     * @param bean
     * @return
     */
    public static Map<String, Object> beanToMap(Object bean) {
        PropertyDescriptor[] propertyDescriptorArray =
getPropertyDescriptorArray(bean);
        Map<String, Object> parameterMap = new HashMap<String, Object>();
        for (PropertyDescriptor propertyDescriptor : propertyDescriptorArray) {
            Object value = getPropertyDescriptorValue(bean, propertyDescriptor);
            parameterMap.put(propertyDescriptor.getName(), value);
        }
        return parameterMap;
    }

    /**
     * 通过反射设置属性
     *
     * @param bean
     * @param key
     * @param value
     */
    public static void setPropertie(Object bean, String key, Object value) {
        if (null != bean && StringUtils.isNotEmpty(key)) {
            PropertyDescriptor[] descriptor = getPropertyDescriptorArray(bean);
            PropertyDescriptor propertyDescriptor =
getPropertyDescriptor(descriptor, key);
            setPropertyDescriptorValue(bean, propertyDescriptor, value);
        }
    }

    /**
     * 通过反射设置属性
     *
     * @param bean
     * @param key
     * @param value
     * @param skipExist 是否跳过已存在的属性
     */
    public static void setPropertie(Object bean, String key, Object value,
boolean skipExist) {
        if (null != bean && StringUtils.isNotEmpty(key)) {
            if (skipExist) {
                Object propValue = getPropertie(bean, key);
                if (null == propValue) {
                    setPropertie(bean, key, value);
                }
            } else {
                setPropertie(bean, key, value);
            }
        }
```

```
    }


    /**
     * 通过反射将map的key value 映射到实体类中
     *
     * @param bean
     * @param skipExist 是否跳过已存在的属性
     */
    public static void setPropertie(Object bean, Map<String, Object>
parameterMap, boolean skipExist) {
        if (null != bean && null != parameterMap && !parameterMap.isEmpty()) {
            for (Map.Entry<String, Object> entry : parameterMap.entrySet()) {
                setPropertie(bean, entry.getKey(), entry.getValue());
            }
        }

    }


    /**
     * 获取属性的值
     *
     * @param bean
     * @param key
     * @return
     */
    public static Object getPropertie(Object bean, String key) {
        Object value = null;
        if (null != bean && StringUtils.isNotEmpty(key)) {
            PropertyDescriptor[] descriptor = getPropertyDescriptorArray(bean);
            PropertyDescriptor propertyDescriptor =
getPropertyDescriptor(descriptor, key);
            value = getPropertyDescriptorValue(bean, propertyDescriptor);
        }
        return value;
    }


    public static Object getPropertyDescriptorValue(Object bean,
PropertyDescriptor propertyDescriptor) {
        Object value = null;
        if (null != propertyDescriptor) {
            Method readMethod = propertyDescriptor.getReadMethod();
            value = invok(readMethod, bean,
propertyDescriptor.getPropertyType(), null);
        }
        return value;
    }


    public static void setPropertyDescriptorValue(Object bean,
PropertyDescriptor propertyDescriptor, Object value) {
        if (null != propertyDescriptor) {
            Method writeMethod = propertyDescriptor.getWriteMethod();
            invok(writeMethod, bean, propertyDescriptor.getPropertyType(),
value);
        }
```

```
    }

    /**
     * 获取 PropertyDescriptor 属性
     *
     * @param propertyDescriptorArray
     * @param key
     * @return
     */
    public static PropertyDescriptor getPropertyDescriptor(PropertyDescriptor[]
propertyDescriptorArray, String key) {
        PropertyDescriptor propertyDescriptor = null;
        for (PropertyDescriptor descriptor : propertyDescriptorArray) {
            String fieldName = descriptor.getName();
            if (fieldName.equals(key)) {
                propertyDescriptor = descriptor;
                break;
            }
        }
        return propertyDescriptor;
    }


    /**
     * 获取 PropertyDescriptor 属性
     *
     * @param bean
     * @param key
     * @return
     */
    public static PropertyDescriptor getPropertyDescriptor(Object bean, String
key) {
        PropertyDescriptor[] propertyDescriptorArray =
getPropertyDescriptorArray(bean);
        return getPropertyDescriptor(propertyDescriptorArray, key);
    }


    /**
     * invok 调用方法
     *
     * @param methodName
     * @param bean
     * @param targetType
     * @param value
     * @return
     */
    public static Object invok(String methodName, Object bean, Class<?>
targetType, Object value) {
        Object resultValue = null;
        if (StringUtils.isNotEmpty(methodName) && null != bean) {
            Method method = getMethod(bean.getClass(), methodName);
            if (null != method) {
                resultValue = invok(method, bean, targetType, value);
            }
        }
        return resultValue;
    }
```

```
    /**
     * 调用 invok 方法
     *
     * @param method
     * @param bean
     * @param value
     */
    public static Object invok(Method method, Object bean, Class<?> targetType,
Object value) {
        //  System.out.println("method:" + method.getName() + "   bean:" +
bean.getClass().getName() + "     " + value);
        Object resultValue = null;
        if (null != method && null != bean) {
            try {
                int count = method.getParameterCount();
                if (count >= 1) {
                    if (null != value) {
                        value = ConvertUtils.convert(value, targetType);
                    }
                    resultValue = method.invoke(bean, value);
                } else {
                    resultValue = method.invoke(bean);
                }

            } catch (IllegalAccessException e) {
                e.printStackTrace();
            } catch (InvocationTargetException e) {
                e.printStackTrace();
            }
        }
        return resultValue;
    }

    /**
     * 获取内省的属性
     *
     * @param bean
     * @return
     */
    public static PropertyDescriptor[] getPropertyDescriptorArray(Object bean) {
        BeanInfo beanInfo = null;
        PropertyDescriptor[] propertyDescriptors = null;
        try {
            beanInfo = Introspector.getBeanInfo(bean.getClass());
        } catch (IntrospectionException e) {
            e.printStackTrace();
        }
        if (null != beanInfo) {
            propertyDescriptors = beanInfo.getPropertyDescriptors();
        }
        return propertyDescriptors;
    }

    /**
     * 获取method 方法
     *
     * @param clazz
```

```
     * @param methodName
     * @return
     */
    private static Method getMethod(Class clazz, String methodName) {
        Method method = null;
        if (null != clazz) {
            try {
                method = clazz.getDeclaredMethod(methodName);
            } catch (NoSuchMethodException e) {
                e.printStackTrace();
            }
        }
        return method;
    }

    private static Object getBean(Class clazz) {
        Object bean = null;
        if (null != clazz) {
            try {
                bean = clazz.newInstance();
            } catch (InstantiationException e) {
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                e.printStackTrace();
            }
        }
        return bean;
    }

    /**
     * 同步 bean 中的数据
     *
     * @param oldBean
     * @param newBean
     * @param <T>
     */
    public static <T> void syncBeanData(T oldBean, T newBean) {
        PropertyDescriptor[] descriptorArray =
getPropertyDescriptorArray(newBean);
        for (PropertyDescriptor propertyDescriptor : descriptorArray) {
            Object newValue = getPropertyDescriptorValue(newBean,
propertyDescriptor);
            Object oldValue = getPropertyDescriptorValue(oldBean,
propertyDescriptor);
            if (null == newValue && oldValue != null) {
                setPropertyDescriptorValue(newBean, propertyDescriptor,
oldValue);
            }
        }
    }


    /**
     * 通过反射获取class字节码文件
     *
     * @param className
     * @return
     */
```

```
    public static Class getClassForName(String className) {
        Class clazz = null;
        if (StringUtils.isNotEmpty(className)) {
            try {
                clazz = Class.forName(className);
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        }
        return clazz;
    }


    /**
     * 通过反射获取对象
     *
     * @param className
     * @return
     */
    public static Object getClassForBean(String className) {
        Object bean = null;
        Class clazz = getClassForName(className);
        if (null != clazz) {
            try {
                bean = clazz.newInstance();
            } catch (InstantiationException e) {
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                e.printStackTrace();
            }
        }
        return bean;
    }

    /**
     * 获取属性字段的注解属性
     *
     * @param bean
     * @param propertyDescriptor
     * @return
     */
    public static Annotation[] getFieldAnnotations(Object bean,
PropertyDescriptor propertyDescriptor) {
        List<Field> fieldList =
Arrays.asList(bean.getClass().getDeclaredFields()).stream().filter(f ->
f.getName().equals(propertyDescriptor.getName())).collect(Collectors.toList());
        if (null != fieldList && fieldList.size() > 0) {
            return fieldList.get(0).getDeclaredAnnotations();
        }
        return null;
    }

    /**
     * 获取属性字段的注解属性
     *
     * @param bean
     * @param key
     * @return
```

```
     */
    public static Annotation[] getFieldAnnotations(Object bean, String key) {
        PropertyDescriptor propertyDescriptor = getPropertyDescriptor(bean,
key);
        return getFieldAnnotations(bean, propertyDescriptor);
    }


}
```

3.6 其他工具类

```
com.heima.crawler.utils.DateUtils;
com.heima.common.common.util.HMStringUtils;
com.heima.common.kafka.KafkaSender;
com.heima.common.kafka.messages.SubmitArticleAuthMessage
```

3.7 AbstractProcessFlow

com.heima.crawler.process.AbstractProcessFlow

```
  /**
     * 获取原始的请求的JSON数据
     *
     * @param url
     * @param parameterMap
     * @return
     */
    public String getOriginalRequestJsonData(String url, Map<String, String>
parameterMap) {
        //获取代理
        CrawlerProxy proxy = crawlerProxyProvider.getRandomProxy();

        //获取Cookie列表
        List<CrawlerCookie> cookieList = cookieHelper.getCookieEntity(url,
proxy);
        //通过HttpClient方式来获取数据
        String jsonData = getHttpClientRequestData(url, parameterMap,
cookieList, proxy);
        //如果不是JSON 说明数据抓取失败则通过SeleniumUtils的方式来获取数据
        if (!isJson(jsonData)) {
            CrawlerHtml crawlerHtml = getSeleniumRequestData(url, parameterMap,
proxy);
            jsonData = seleniumClient.getJsonData(crawlerHtml);
        }
        return jsonData;
    }
```

```
/**
     * 验证 字符串是否是json格式
     *
     * @param jsonData
     * @return
     */
    public boolean isJson(String jsonData) {
        boolean isJson = false;
        try {
            isJson = JsonValidator.getJsonValidator().validate(jsonData);
        } catch (ExecutionException e) {
            e.printStackTrace();
        }
        return isJson;
    }
```

3.8 CrawlerHtmlParsePipeline类

com.heima.crawler.process.parse.impl.CrawlerHtmlParsePipeline

> 解析后的数据存储，因为数据量较大，所有使用线程池来进行操作控制并发线程数

```
/**
 * 对 AbstractHtmlParsePipeline 抽象类的实现，
 * 对应数据要转换为何种格式
 * 具体对象要怎么存储，
 * 数据库的存储方式
 * <p>
 * 对评论的抓取以及存储
 */
@Component
@Log4j2
public class CrawlerHtmlParsePipeline extends
AbstractHtmlParsePipeline<CrawlerParseItem> {

    private static final ResourceBundle resourceBundle =
ResourceBundle.getBundle("crawler");
    private static final String csdn_comment_url =
resourceBundle.getString("csdn.comment.url");


    /**
     * 下次更新时间
     */
    private static final String[] next_update_hour_array =
resourceBundle.getString("crawler.nextupdatehours").split(",");


    @Autowired
    private KafkaSender kafkaSender;

    @Autowired
    private CrawlerNewsService crawlerNewsService;
```

```
    @Autowired
    private CrawlerNewsCommentService clNewsCommentService;

    @Autowired
    private CrawlerNewsAdditionalService crawlerNewsAdditionalService;


    @Autowired
    private AdLabelService adLabelService;


    @Override
    public void preParameterHandel(Map<String, Object> parameter) {
        String readCount = HMStringUtils.toString(parameter.get("readCount"));
        if (StringUtils.isNotEmpty(readCount)) {
            readCount = readCount.split(" ")[1];
            if (StringUtils.isNotEmpty(readCount)) {
                parameter.put("readCount", readCount);
            }
        }
    }

    /**
     * Html 数据处理的入口
     *
     * @param parseItem
     */
    @Override
    public void handelHtmlData(final CrawlerParseItem parseItem) {
        long currentTime = System.currentTimeMillis();
        log.info("将数据加入线程池进行执行，url:{},handelType:{}",
parseItem.getUrl(), parseItem.getHandelType());
        CrawlerThreadPool.submit(() -> {
            //正向抓取
            if
(CrawlerEnum.HandelType.FORWARD.name().equals(parseItem.getHandelType())) {
                log.info("开始处理消息,url:{},handelType:{}", parseItem.getUrl(),
parseItem.getHandelType());
                //  kafkaSender.sendCrawlerParseItemMessage(parseItem);
                addParseItemMessage(parseItem);
                //逆向抓取
            } else if
(CrawlerEnum.HandelType.REVERSE.name().equals(parseItem.getHandelType())) {
                //更新附加数据
                updateAdditional(parseItem);
            }
            log.info("处理文章数据完成，url:{},handelType:{}，耗时：{}",
parseItem.getUrl(), parseItem.getHandelType(), System.currentTimeMillis() -
currentTime);
        });

    }

    /**
     * 添加解析后的消息
     *
     * @throws ExecutionException
     * @throws InterruptedException
```

```
     */
    public void addParseItemMessage(CrawlerParseItem parseItem) {
        long currentTime = System.currentTimeMillis();
        String url = null;
        String handelType = null;
        if (null != parseItem) {
            url = parseItem.getUrl();
            handelType = parseItem.getHandelType();
            log.info("开始添加数据,url:{},handelType:{}", url,
parseItem.getHandelType());
            //添加文章数据
            ClNews clNews = addClNewsData(parseItem);
            if (null != clNews) {
                //添加附加数据
                addAdditional(parseItem, clNews);
                //添加回复数据 只有评论数>0 才能进行添加
                if (null != parseItem && null != parseItem.getCommentCount() &&
parseItem.getCommentCount() > 0) {
                    addCommentData(parseItem, clNews);
                }
                sendSubmitArticleAutoMessage(clNews.getId());
            }
        }
        log.info("添加数据完成,url:{},handelType:{},耗时:{}", url, handelType,
System.currentTimeMillis() - currentTime);
    }

    /**
     * 发起自动主动审核
     *
     * @param clNewId
     */
    public void sendSubmitArticleAutoMessage(Integer clNewId) {
        log.info("开始发送自动审核消息,id:{}", clNewId);
        SubmitArticleAuto submitArticleAuto = new SubmitArticleAuto();
        submitArticleAuto.setArticleId(clNewId);
        submitArticleAuto.setType(SubmitArticleAuto.ArticleType.CRAWLER);
        SubmitArticleAuthMessage submitArticleAuthMessage = new
SubmitArticleAuthMessage(submitArticleAuto);
        kafkaSender.sendSubmitArticleAuthMessage(submitArticleAuthMessage);
        log.info("发送自动审核消息完成,id:{}", clNewId);
    }

    /**
     * 文章内容的处理
     *
     * @param parseItem
     * @return
     */
    private ClNews addClNewsData(CrawlerParseItem parseItem) {
        log.info("开始添加文章内容");
        ClNews clNews = null;
        if (null != parseItem) {
            HtmlParser htmlParser =
HtmlParser.getHtmlParser(getParseExpression(), getDefHtmlStyleMap());
            //将Html内容转换为HtmlLabel 对象类别
            List<HtmlLabel> htmlLabelList =
htmlParser.parseHtml(parseItem.getContent());
```

```
            //获取文章类型
            int type = getType(htmlLabelList);
            parseItem.setDocType(type);
            String josnStr = JSON.toJSONString(htmlLabelList);
            parseItem.setCompressContent(ZipUtils.gzip(josnStr));
            ClNewsAdditional clNewsAdditional =
crawlerNewsAdditionalService.getAdditionalByUrl(parseItem.getUrl());
            if (null == clNewsAdditional) {
                clNews = toClNews(parseItem);
                long currentTime = System.currentTimeMillis();
                log.info("开始插入新的文章");
                crawlerNewsService.saveNews(clNews);
                log.info("插入新的文章完成，耗时：{}", System.currentTimeMillis() -
currentTime);
            } else {
                log.info("文章URL已存在不重复添加，URL：{}",
clNewsAdditional.getUrl());
            }
        }
        log.info("添加文章内容完成");
        return clNews;
    }


    /**
     * 处理文章附加信息
     *
     * @param parseItem
     * @param clNews
     */
    public void addAdditional(CrawlerParseItem parseItem, ClNews clNews) {
        long currentTime = System.currentTimeMillis();
        log.info("开始处理文章附加数据");
        if (null != parseItem && null != clNews) {
            ClNewsAdditional clNewsAdditional = toClNewsAdditional(parseItem,
clNews);
            crawlerNewsAdditionalService.saveAdditional(clNewsAdditional);
        }
        log.info("文章附加数据处理完成,耗时：{}", System.currentTimeMillis() -
currentTime);
    }

    /**
     * 逆向数据更新
     *
     * @param parseItem
     */
    public void updateAdditional(CrawlerParseItem parseItem) {
        long currentTime = System.currentTimeMillis();
        log.info("开始更新文章附加数据");
        if (null != parseItem) {
            ClNewsAdditional clNewsAdditional =
crawlerNewsAdditionalService.getAdditionalByUrl(parseItem.getUrl());
            if (null != clNewsAdditional) {
                clNewsAdditional.setNewsId(null);
                clNewsAdditional.setUrl(null);
                //阅读量设置
                clNewsAdditional.setReadCount(parseItem.getReadCount());
```

```
                //评论数设置
                clNewsAdditional.setComment(parseItem.getCommentCount());
                //点赞数设置
                clNewsAdditional.setLikes(parseItem.getLikes());
                //更新数据设置
                clNewsAdditional.setUpdatedTime(new Date());
                clNewsAdditional.setUpdateNum(clNewsAdditional.getUpdateNum() +
1);
                int nextUpdateHours =
getNextUpdateHours(clNewsAdditional.getUpdateNum());
                clNewsAdditional.setNextUpdateTime(DateUtils.addHours(new
Date(), nextUpdateHours));
                crawlerNewsAdditionalService.updateAdditional(clNewsAdditional);
            }
        }
        log.info("更新文章附加数据完成，耗时：{}", System.currentTimeMillis() -
currentTime);
    }

    /**
     * 处理评论数据
     *
     * @param parseItem
     */
    public void addCommentData(CrawlerParseItem parseItem, ClNews clNews) {
        long currentTime = System.currentTimeMillis();
        log.info("开始获取文章评论数据");
        List<ClNewsComment> commentList = getCommentData(parseItem);
        if (null != commentList && !commentList.isEmpty()) {
            for (ClNewsComment comment : commentList) {
                comment.setNewsId(clNews.getId());
                clNewsCommentService.saveClNewsComment(comment);
            }
        }
        log.info("获取文章评论数据完成，耗时：{}", System.currentTimeMillis() -
currentTime);
    }

    /**
     * 转换为数据库对象
     *
     * @param parseItem
     * @return
     */
    private ClNews toClNews(CrawlerParseItem parseItem) {
        ClNews clNews = new ClNews();
        clNews.setName(parseItem.getAuthor());
        clNews.setLabels(parseItem.getLabels());
        clNews.setContent(parseItem.getCompressContent());
        clNews.setLabelIds(adLabelService.getLableIds(parseItem.getLabels()));
        Integer channelId =
adLabelService.getAdChannelByLabelIds(clNews.getLabelIds());
        clNews.setChannelId(channelId);
        clNews.setTitle(parseItem.getTitle());
        clNews.setType(parseItem.getDocType());
        clNews.setStatus((byte) 1);
        clNews.setCreatedTime(new Date());
        String releaseDate = parseItem.getReleaseDate();
```

```
        if (StringUtils.isNotEmpty(releaseDate)) {
            clNews.setOriginalTime(DateUtils.stringToDate(releaseDate,
DateUtils.DATE_TIME_FORMAT_CHINESE));
        }
        return clNews;
    }


    /**
     * 转换为文章附加信息
     *
     * @param parseItem
     * @param clNews
     * @return
     */
    private ClNewsAdditional toClNewsAdditional(CrawlerParseItem parseItem,
ClNews clNews) {
        ClNewsAdditional clNewsAdditional = null;
        if (null != parseItem) {
            clNewsAdditional = new ClNewsAdditional();
            //设置文章ID
            clNewsAdditional.setNewsId(clNews.getId());
            //设置阅读数
            clNewsAdditional.setReadCount(parseItem.getReadCount());
            //设置回复数
            clNewsAdditional.setComment(parseItem.getCommentCount());
            //设置点赞数
            clNewsAdditional.setLikes(parseItem.getLikes());
            //设置URL
            clNewsAdditional.setUrl(parseItem.getUrl());
            //设置更新时间
            clNewsAdditional.setUpdatedTime(new Date());
            //设置创建时间
            clNewsAdditional.setCreatedTime(new Date());
            //设置更新次数
            clNewsAdditional.setUpdateNum(0);
            //设置下次更新时间
            int nextUpdateHour =
getNextUpdateHours(clNewsAdditional.getUpdateNum());
            /**
             * 设置下次更新时间
             */
            clNewsAdditional.setNextUpdateTime(DateUtils.addHours(new Date(),
nextUpdateHour));
        }
        return clNewsAdditional;
    }

    /**
     * 获取下次更新时间
     *
     * @param count
     * @return
     */
    private int getNextUpdateHours(Integer count) {
        if (null != next_update_hour_array && next_update_hour_array.length >
count) {
            return Integer.parseInt(next_update_hour_array[count]);
```

```
        } else {
            return 2 << count;
        }
    }

    /**
     * 获取图文类型
     * <p>
     * 0：无图片
     * 1：单图
     * 2：多图
     *
     * @param htmlLabelList
     * @return
     */
    public int getType(List<HtmlLabel> htmlLabelList) {
        int type = 0;
        int num = 0;
        if (null != htmlLabelList && !htmlLabelList.isEmpty()) {
            for (HtmlLabel htmlLabel : htmlLabelList) {
                if
(CrawlerEnum.HtmlType.IMG_TAG.getDataType().equals(htmlLabel.getType())) {
                    num++;
                }
            }
        }
        if (num == 0) {
            type = 0;
        } else if (num == 1) {
            type = 1;
        } else {
            type = 2;
        }
        return type;
    }


    /**
     * 获取评论列表
     *
     * @param parseItem
     * @return
     */
    private List<ClNewsComment> getCommentData(ParseItem parseItem) {
        //构建评论的URl
        String buildCommentUrl = buildCommentUrl(parseItem);
        //调用父类方法进行HttpClient请求进行获取数据
        String jsonData = getOriginalRequestJsonData(buildCommentUrl, null);
        //解析获取的JSON数据
        List<ClNewsComment> commentList = analysisCommentJsonData(jsonData);
        return commentList;
    }

    /**
     * 解析评论数据
     *
     * @param jsonData
     * @return
```

```
     */

    public List<ClNewsComment> analysisCommentJsonData(String jsonData) {
        if (StringUtils.isEmpty(jsonData)) {
            return null;
        }
        List<ClNewsComment> commentList = new ArrayList<ClNewsComment>();
        JSONObject jsonObject = JSON.parseObject(jsonData);
        Map<String, Object> map = jsonObject.getObject("data", Map.class);
        JSONArray jsonArray = (JSONArray) map.get("list");
        if (null != jsonArray) {
            List<Map> dataInfoList = jsonArray.toJavaList(Map.class);
            for (Map<String, Object> dataInfo : dataInfoList) {
                JSONObject infoObject = (JSONObject) dataInfo.get("info");
                Map<String, Object> infoMap =
infoObject.toJavaObject(Map.class);
                ClNewsComment comment = new ClNewsComment();

 comment.setContent(HMStringUtils.toString(infoMap.get("Content")));

 comment.setUsername(HMStringUtils.toString(infoMap.get("UserName")));
                Date date =
DateUtils.stringToDate(HMStringUtils.toString(infoMap.get("PostTime")),
DateUtils.DATE_TIME_FORMAT);
                comment.setCommentDate(date);
                comment.setCreatedDate(new Date());
                commentList.add(comment);
            }
        }
        return commentList;
    }

    /**
     * 生成评论访问链接
     *
     * @param parseItem
     * @return
     */
    private String buildCommentUrl(ParseItem parseItem) {
        String buildCommentUrl = csdn_comment_url;
        Map<String, Object> map = ReflectUtils.beanToMap(parseItem);
        for (Map.Entry<String, Object> entry : map.entrySet()) {
            String key = entry.getKey();
            String buildKey = "${" + key + "}";
            Object value = entry.getValue();
            if (null != value) {
                String strValue = value.toString();
                buildCommentUrl = buildCommentUrl.replace(buildKey, strValue);
            }
        }
        return buildCommentUrl;
    }


    @Override
    public int getPriority() {
        return 1000;
    }
```

```
}
```

3.9 综合测试

启动测试，是否能存储数据到数据库中