

# 实时计算模块

学习目标:

- 1 掌握Flink数据流开发过程
- 2 掌握redis代码的编写
- 3 掌握实时指标的查询
- 4 了解热数据和冷数据的处理方式

## 1 实时数据处理(cold-flink)

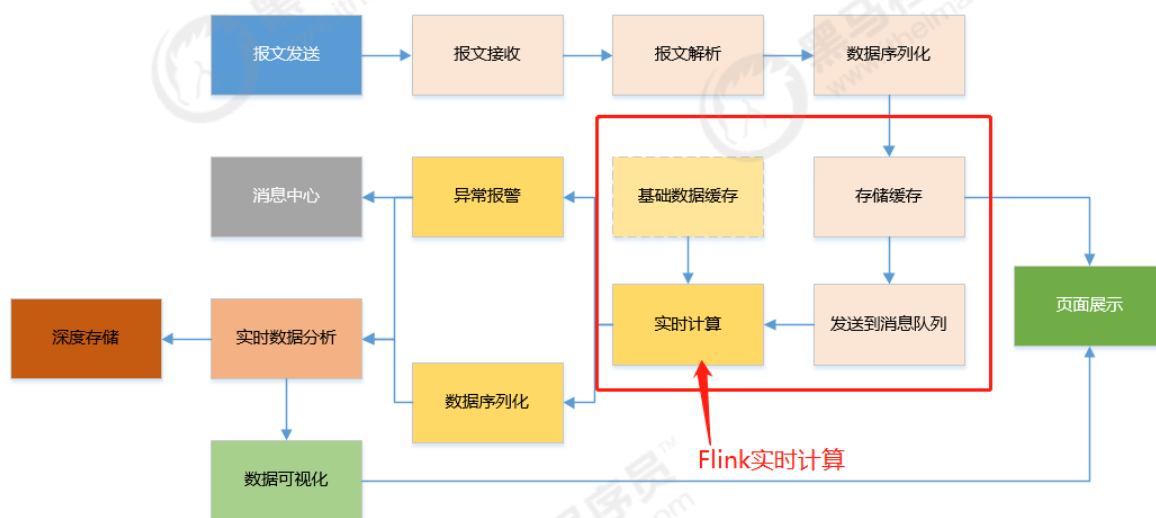
### 1.1 需求分析

在冷链监控项目中，我们需要将设备采集器将温度、湿度等数据和温湿度阈值进行比对，从中找到温湿度异常记录。

实现过程是：

- Netty服务器将接收的设备报文数据转换成json数据，提交到kafka队列中
- 我们的flink程序实时监听kafka中的消息，根据消息中的温湿度阈值和当前温湿度进行比对，进行异常预警。

业务流程如下：



此时实时计算的数据包括两部分：

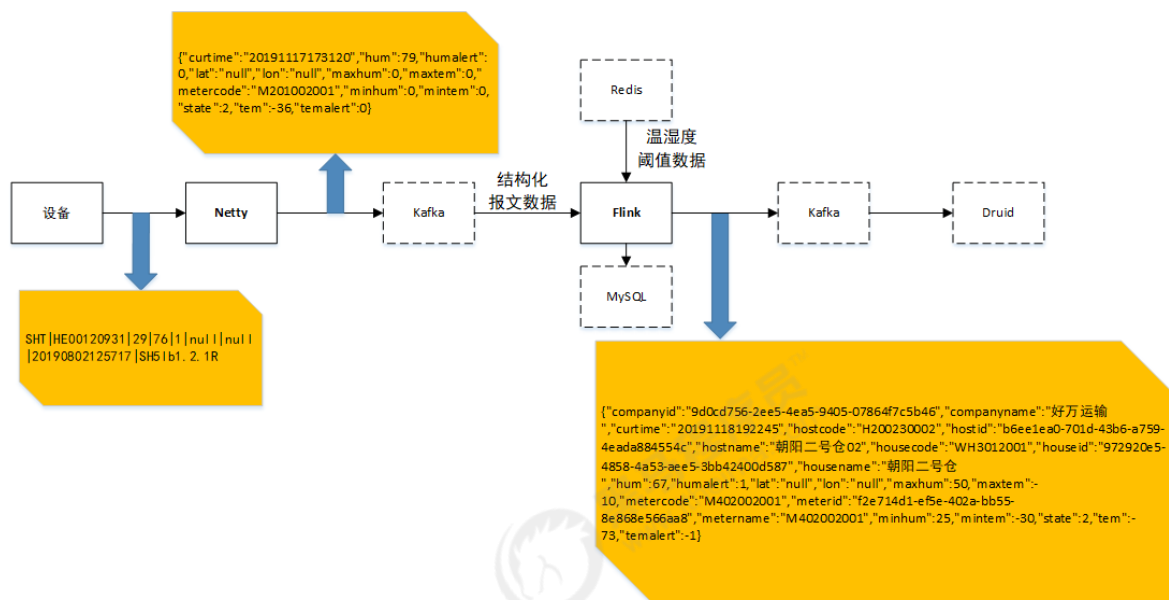
1. 缓存中仪表的数据，主要是温度阈值和湿度阈值
2. kafka消息队列中的设备提交的数据，主要是温度和湿度

根据这两部分数据，我们希望计算出某台设备是否温度高于阈值或者湿度高于阈值，同时也希望通过报表形式将过去一小时内每台设备报警数统计出来。

这个需求的业务规则比较简单，通过这个需求的开发，我们能够掌握如何使用flink开发实时流数据处理程序。

### 1.2 代码实现

#### 1.2.1 实现逻辑



1. 监听kafka队列中的数据
2. 从redis中获取仪表阈值基础数据
3. 将第1、2两步的数据合并，并判断温度、湿度是否超过阈值，同时更新预警字段
4. 将第3步的数据推送至kafka队列中（Apache Druid数据源）
5. 将第3步的数据保存到mysql数据库中，供应用系统查询

### 1.2.2 数据模型

名称	类型	长度	是否主键	默认值	备注
meterCode	varchar	50	1		设备编码
meterId	varchar	50			设备ID
meterName	varchar	100			设备名称
hostId	varchar	50			主机Id
hostCode	varchar	50			主机编码
hostName	varchar	100			主机名称
houseId	varchar	50			仓库Id
houseCode	varchar	50			仓库编码
houseName	varchar	100			仓库名称
companyId	varchar	50			公司Id
companyName	varchar	100			公司名称

名称	类型	长度	是否主键	默认值	备注
tem	int	5			温度
maxTem	int	5			温度上限
minTem	int	5			温度下限
hum	int	5			湿度
maxHum	int	5			湿度上限
minHum	int	5			湿度下限
temAlert	int	5		0	温度状况： 1：高温，0：正常，-1：低温
humAlert	int	5		0	湿度状况： 1：高湿，0：正常，-1：低湿
state	int	5			设备状态: 1-在用，0-停用，2-异常
lon	varchar	30			经度
lat	varchar	30			维度
curtime	varchar	50			提交时间
createTime	timestamp	0		CURRENT_TIMESTAMP	创建时间

### 1.2.3 核心代码

```

public class StreamingJob {
    public static void main(String[] args) throws Exception{
        // 获取环境信息
        final StreamExecutionEnvironment env =
            StreamExecutionEnvironment.getExecutionEnvironment();

        //设置容错
        env.enableCheckpointing(5000);

        //设置检查点模式

        env.getCheckpointConfig().setCheckpointingMode(CheckpointConfig.DEFAULT_MODE);

        //设置重启策略
    }
}

```

```

env.getConfig().setRestartStrategy(RestartStrategies.fixedDelayRestart(4,
1000));

Properties properties = new Properties();
properties.setProperty("bootstrap.servers", "172.17.0.143:9092");
properties.setProperty("group.id", "coldflink");
properties.setProperty("auto.offset.reset", "earliest");

//kafka队列consumer, 接收netty发送过来的数据
FlinkKafkaConsumer<String> consumer = new FlinkKafkaConsumer<String>
("device_msg_topic", new SimpleStringSchema(), properties);
consumer.setStartFromLatest();

//设置实时数据源: kafka队列
DataStream<MessageEntity> stream = env.addSource(consumer)
    .setParallelism(1)
    .map(string -> JSON.parseObject(string, MessageEntity.class));

//从redis中获取设备信息, 合并至消息中
DataStream<MessageEntity> outputStream = stream.map(new
MessageAggregate());

//存储设备指标消息
outputStream.addSink(new SinkToMySQL());

// 将消息发送到kafka队列
// Druid监听此队列, 将消息保存到druid中
FlinkKafkaProducer<String> myProducer = new FlinkKafkaProducer<String>
("all_device_message", new SimpleStringSchema(), properties);
outputStream.map(new MapFunction<MessageEntity, String>() {
    @Override
    public String map(MessageEntity value) throws Exception {
        return JSON.toJSONString(value);
    }
}).addSink(myProducer);
myProducer.close();

env.execute("冷链设备实时监控任务");
}

}

```

获取redis中存储的温湿度阈值：

```

public class MessageAggregate implements MapFunction<MessageEntity,
MessageEntity> {
    final JedisUtil jedisUtil= JedisUtil.getInstance();

    @Override
    public MessageEntity map(MessageEntity value) throws Exception {
        Jedis jedis = jedisUtil.getJedis();
        try{
            byte[] meterBy =
jedis.get(SafeEncoder.encode(value.getMetercode()));

```

```

        MeterEntity meter = (MeterEntity)
SerializeUtil.unserialize(meterBy);

        value.setMeterid(meter.getId());
        value.setMetername(meter.getMetername());
        value.setHostcode(meter.getHostcode());
        value.setHostid(meter.getHostid());
        value.setHostname(meter.getHostname());

        value.setHouseid(meter.getHouseid());
        value.setHousecode(meter.getHousecode());
        value.setHousename(meter.getHousename());
        value.setCompanyid(meter.getCompanyid());
        value.setCompanyname(meter.getCompanyname());
//        value.setState(Integer.valueOf(meter.getMeterstatus()));

        value.setMaxtem(Integer.valueOf(meter.getMaxtem()));
        value.setMintem(Integer.valueOf(meter.getMintem()));
        value.setMaxhum(Integer.valueOf(meter.getMaxhum()));
        value.setMinhum(Integer.valueOf(meter.getMinhum()));

        if(value.getTem() > value.getMaxhum()){
            value.setTemalert(1);    //高温
        }
        else if(value.getTem() < value.getMintem()){
            value.setTemalert(-1);    //低温
        }
        else{
            value.setTemalert(0);    //正常
        }

        if(value.getHum() > value.getMaxhum()){
            value.setHumalert(1);
        }
        else if(value.getHum() < value.getMinhum()){
            value.setHumalert(-1);
        }
        else{
            value.setHumalert(0);
        }

        return value;
    }
    catch(Exception e){
        jedisUtil.returnBrokenResource(jedis);

        e.printStackTrace();
    }
    finally {
        jedisUtil.returnJedis(jedis);
    }

    return value;
}
}

```

### 存储到mysql：

```

public class SinkToMySQL extends RichSinkFunction<MessageEntity> {
    private Connection connection = null;
    private PreparedStatement preparedStatement = null;
    private String userName = null;
    private String password = null;
    private String driverName = null;
    private String DBUrl = null;
    private StringBuffer sql = new StringBuffer();

    public SinkToMySQL() {
        userName = "cold";
        password = "cold123";
        driverName = "com.mysql.jdbc.Driver";
        DBUrl = "jdbc:mysql://172.17.0.143:3306/cold?
allowMultiQueries=true&useUnicode=true&characterEncoding=UTF-
8&useSSL=false&autoReconnect=true";
        sql.append( "replace into cz_message_realtime(" );

        sql.append( "meterCode, meterId, meterName, ");
        sql.append("hostId, hostCode, hostName, ");
        sql.append( "houseId, houseCode, houseName, ");
        sql.append( "companyId, companyName, ");
        sql.append( "tem, maxTem, minTem, ");
        sql.append( "hum, maxHum, minHum, ");
        sql.append( "temAlert, humAlert, state, ");
        sql.append( "lon, lat, curtime) ");
        sql.append( "values(");
        sql.append( "? , ? , ? , ");
        sql.append( "? , ? , ? , ");
        sql.append( "? , ? , ? , ");
        sql.append( "? , ? , ");
        sql.append( "? , ? , ? , ");
        sql.append( "? , ? , ? , ");
        sql.append( "? , ? , ? , ");
        sql.append( "? , ? , ? , ");
        sql.append( "? , ? , ?)");
    }

    @Override
    public void open(Configuration parameters) throws Exception {
        super.open(parameters);
        Class.forName(driverName);
        connection = DriverManager.getConnection(DBUrl, userName, password);
        preparedStatement = connection.prepareStatement(sql.toString());
        super.open(parameters);
    }

    @Override
    public void close() throws Exception {
        super.close();
        if(preparedStatement!=null){
            preparedStatement.close();
        }
        if(connection!=null){
            connection.close();
        }
    }
}

```

```

        super.close();
    }

    @Override
    public void invoke(MessageEntity value) {
        try{
            preparedStatement.setString(1, value.getMetercode());
            preparedStatement.setString(2, value.getMeterid());
            preparedStatement.setString(3, value.getMetername());
            preparedStatement.setString(4, value.getHostid());
            preparedStatement.setString(5, value.getHostcode());
            preparedStatement.setString(6, value.getHostname());
            preparedStatement.setString(7, value.getHouseid());
            preparedStatement.setString(8, value.getHousecode());
            preparedStatement.setString(9, value.getHousename());
            preparedStatement.setString(10, value.getCompanyid());
            preparedStatement.setString(11, value.getCompanyname());

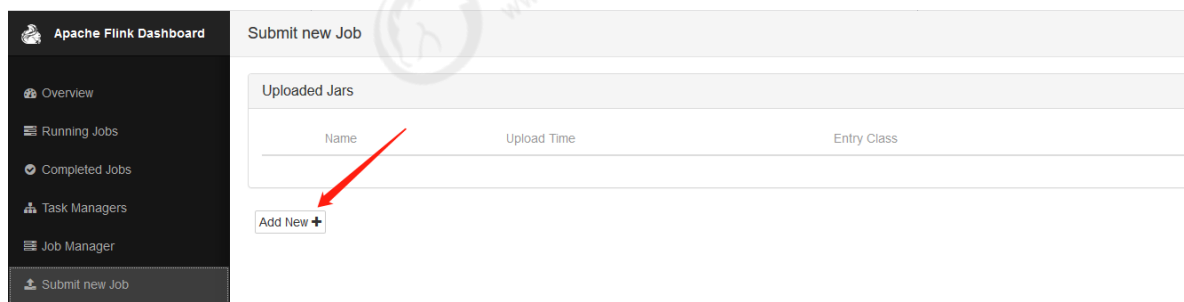
            preparedStatement.setInt(12, value.getTem());
            preparedStatement.setInt(13, value.getMaxtem());
            preparedStatement.setInt(14, value.getMintem());
            preparedStatement.setInt(15, value.getHum());
            preparedStatement.setInt(16, value.getMaxhum());
            preparedStatement.setInt(17, value.getMinhum());

            preparedStatement.setInt(18, value.getTemalert());
            preparedStatement.setInt(19, value.getHumalert());
            preparedStatement.setInt(20, value.getState());
            preparedStatement.setString(21, value.getLon());
            preparedStatement.setString(22, value.getLat());
            preparedStatement.setString(23, value.getCurtime());

            //        preparedStatement.toString();
            preparedStatement.executeUpdate();
        }
        catch (Exception e){
            e.printStackTrace();
        }
    }
}

```

## 1.2.4 提交程序



Apache Flink Dashboard

Submit new Job

Overview

Running Jobs

Completed Jobs

Task Managers

Job Manager

Submit new Job

Submit new Job

Uploaded Jars

Name	Upload Time	Entry Class
Add New +		

Apache Flink Dashboard

Overview

Running Jobs

Completed Jobs

Task Managers

Job Manager

Submit new Job

Submit new Job

Uploaded Jars

Name	Upload Time	Entry Class
<input checked="" type="checkbox"/> cold-flink-1.0.0.jar	2019-08-27, 8:15:15	com.itcast.cold.flink.StreamingJob

com.itcast.cold.flink.StreamingJob

Parallelism

Show Plan

Program Arguments

Submit

Savepoint Path

☐ Allow Non Restored State

Apache Flink Dashboard

Overview

Running Jobs

Completed Jobs

Task Managers

Job Manager

Submit new Job

Running Jobs

Start Time	End Time	Duration	Job Name	Job ID	Tasks	Status
2019-08-27, 15:45:19	2019-08-27, 15:45:32	12s	冷链设备实时监控任务	b38854dd9af5dcb80540b04c1460ac1e	<div><div>1</div><div>0</div><div>0</div><div>1</div><div>0</div><div>0</div><div>0</div><div>0</div></div>	<div>RUNNING</div>

## 2 实时数据查询(cold-monitor)

### 2.1 需求分析

#### 1. 仓库监控

仓库实时数据

28 秒后自动刷新

全部 35

合格 35

报警中 35

未监控 35

undefined

昌平三号仓

-100 °C

-40/-20

34 %

30/60

昌平三号仓

-1 °C

-5/10

31 %

40/45

朝阳一号仓

-42 °C

-5/10

9 %

40/45

昌平三号仓

-10 °C

0/15

86 %

30/50

朝阳二号仓

-51 °C

-30/-10

51 %

25/50

朝阳二号仓

-6 °C

15/25

0 %

25/50

朝阳二号仓

-69 °C

-20/-10

88 %

40/70

昌平二号仓

-27 °C

-10/46

45 %

36/75

朝阳二号仓

-55 °C

10/20

39 %

25/45

朝阳一号仓

-76 °C

-10/15

59 %

30/50

昌平一号仓

-60 °C

-20/20

83 %

10/40

昌平一号仓

-72 °C

-5/45

8 %

20/100

#### 2. 实时报警



实时报警

设备编号

请输入查询编号

仪表编号	仪表名称	管理主机号	仓库名称	企业名称	采集日期	温度	温度阈值	温度级别	湿度	
M402002002	M402002002	H200230002	昌平三号仓	好万运输	20191120203455	-35 °C	-40 / -20	正常	7 %	
M401001001	M401001001	H200130001	朝阳一号仓	好万运输	20191120203451	-63 °C	-5 / 10	低温报警	30 %	
M301001001	M301001001	H100330001	昌平三号仓	好万运输	20191120203450	-11 °C	0 / 15	低温报警	13 %	
M301001002	M301001002	H100330001	昌平三号仓	好万运输	20191120203450	-94 °C	-5 / 10	低温报警	18 %	
M402002001	M402002001	H200230002	朝阳二号仓	好万运输	20191120203445	-40 °C	-30 / -10	低温报警	63 %	
M402003001	M402003001	H209214531	朝阳二号仓	cese	20191120203442	-35 °C	-20 / -10	低温报警	75 %	
M402001001	M402001001	H200230001	朝阳二号仓	好万运输	20191120203441	-19 °C	10 / 20	低温报警	46 %	
M402001002	M402001002	H200230001	朝阳二号仓	好万运输	20191120203441	-26 °C	15 / 25	低温报警	76 %	
M201002001	M201002001	H100220002	昌平二号仓	cese	20191120203440	-97 °C	-10 / 46	低温报警	12 %	

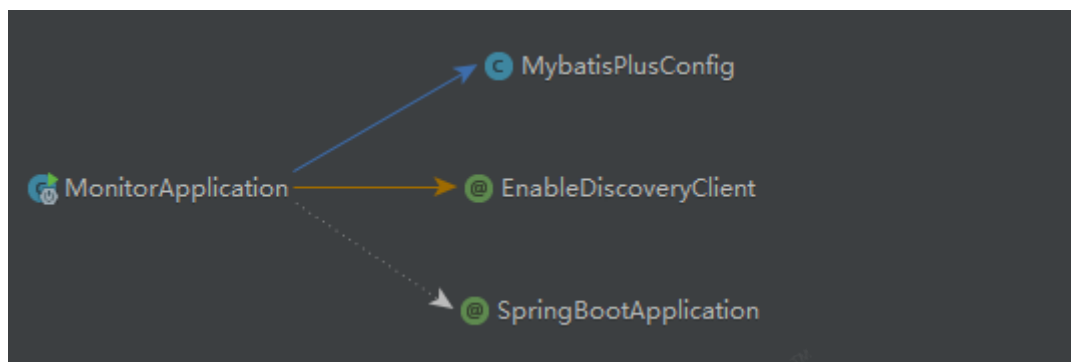
## 2.2 数据模型

名称	类型	长度	是否主键	默认值	备注
meterCode	varchar	50	1		设备编码
meterId	varchar	50			设备ID
meterName	varchar	100			设备名称
hostId	varchar	50			主机Id
hostCode	varchar	50			主机编码
hostName	varchar	100			主机名称
houseId	varchar	50			仓库Id

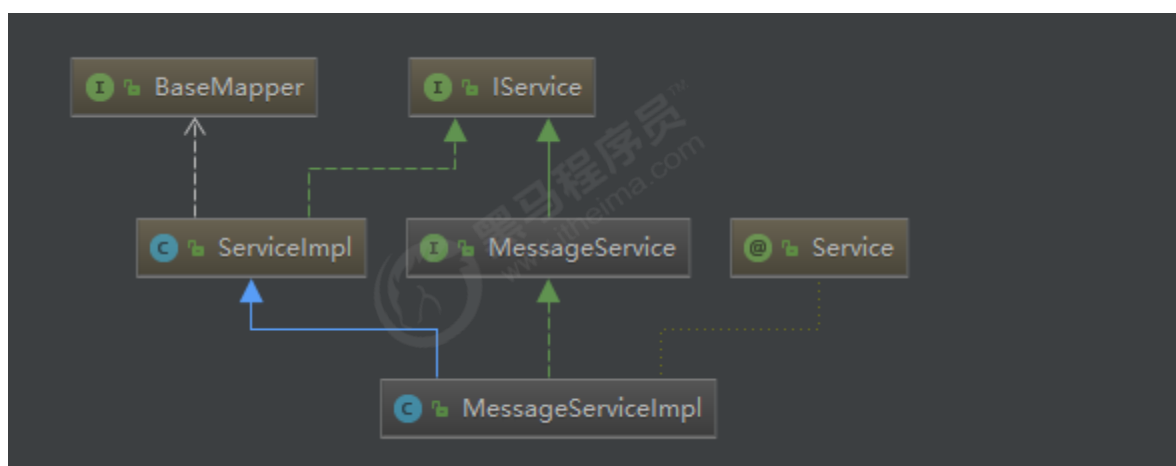
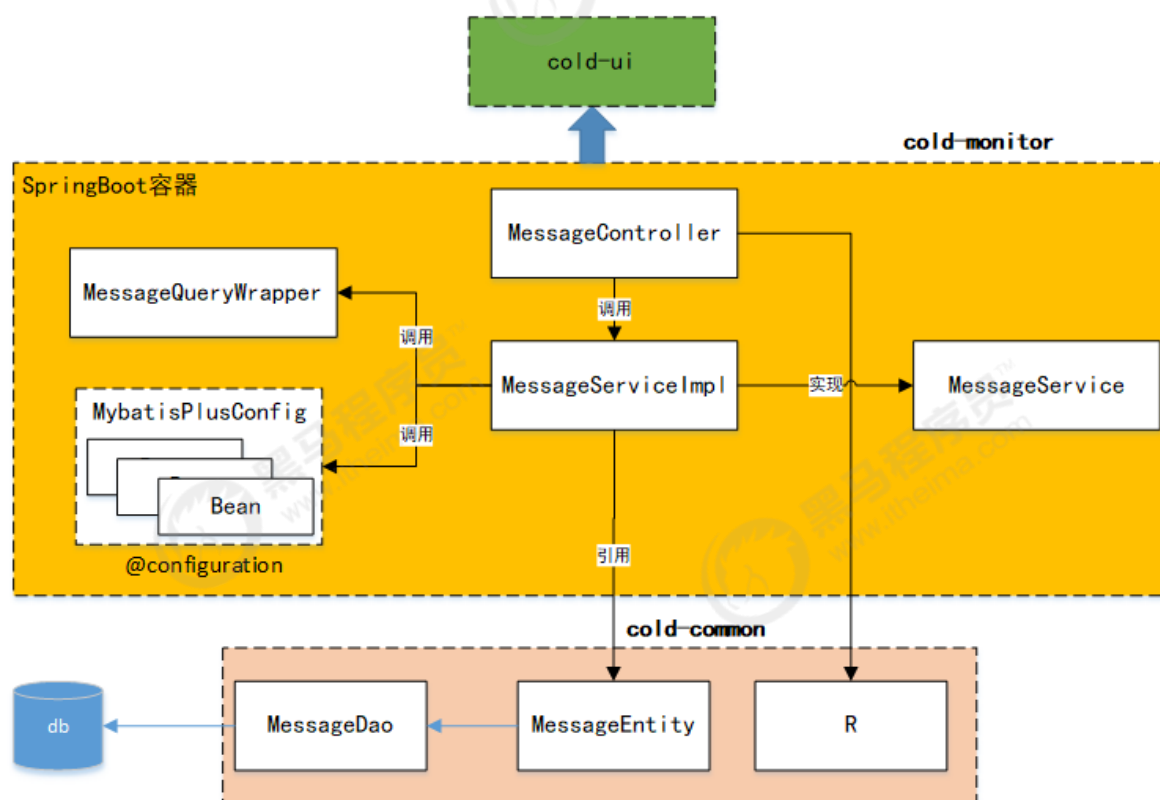
名称	类型	长度	是否主键	默认值	备注
houseCode	varchar	50			仓库编码
houseName	varchar	100			仓库名称
companyId	varchar	50			公司Id
companyName	varchar	100			公司名称
tem	int	5			温度
maxTem	int	5			温度上限
minTem	int	5			温度下限
hum	int	5			湿度
maxHum	int	5			湿度上限
minHum	int	5			湿度下限
temAlert	int	5		0	温度状况： 1：高温，0：正常，-1：低温
humAlert	int	5		0	湿度状况： 1：高湿，0：正常，-1：低湿
state	int	5			设备状态: 1-在用，0-停用，2-异常
lon	varchar	30			经度
lat	varchar	30			维度
curtime	varchar	50			提交时间
createTime	timestamp	0		CURRENT_TIMESTAMP	创建时间

## 2.3 代码实现

### 2.3.1 主程序结构



### 2.3.2 代码结构



### 2.3.3 数据模型

名称	类型	长度	是否主键	默认值	备注
meterCode	varchar	50	1		设备编码
meterId	varchar	50			设备ID
meterName	varchar	100			设备名称
hostId	varchar	50			主机Id
hostCode	varchar	50			主机编码
hostName	varchar	100			主机名称
houseId	varchar	50			仓库Id
houseCode	varchar	50			仓库编码
houseName	varchar	100			仓库名称
companyId	varchar	50			公司Id
companyName	varchar	100			公司名称
tem	int	5			温度
maxTem	int	5			温度上限
minTem	int	5			温度下限
hum	int	5			湿度
maxHum	int	5			湿度上限
minHum	int	5			湿度下限
temAlert	int	5		0	温度状况： 1：高温，0：正常，-1：低温
humAlert	int	5		0	湿度状况： 1：高湿，0：正常，-1：低湿
state	int	5			设备状态: 1-在用，0-停用，2-异常
lon	varchar	30			经度

名称	类型	长度	是否主键	默认值	备注
lat	varchar	30			维度
curtime	varchar	50			提交时间
createTime	timestamp	0		CURRENT_TIMESTAMP	创建时间

### 2.3.4 核心代码

**controller:**

```
/**
 * 仓库监控
 * @param params
 * @return
 */
@RequestMapping("/list")
public R deviceList(@RequestParam Map<String, Object> params){
    List<MessageEntity> list =
messageService.queryMessageRealtime(params).getRecords();

    Map<String, Object> map = new HashMap<String, Object>();
    map.put("items", list);
    return R.ok(map);
}

/**
 * 实时报警
 * @param params
 * @return
 */
@RequestMapping("/realtime")
public R realtime(@RequestParam Map<String, Object> params){
    IPage<MessageEntity> list = messageService.queryMessageRealtime(params);

    Map<String, Object> map = new HashMap<String, Object>();
    map.put("total", list.getTotal());
    map.put("page", list.getPages());
    map.put("items", list.getRecords());
    return R.ok(map);
}
```

**Service:**

```
@Service("messageService")
public class MessageServiceImpl extends ServiceImpl<MessageDao,
MessageEntity>implements MessageService {
    @Override
    public IPage<MessageEntity> queryMessageRealtime(Map<String, Object> params)
    {
        QueryWrapper querywrapper =
MessageQueryWrapper.getMessageCondition(params);
```

```

        int page = Integer.valueOf(params.get("page").toString());
        int pagesize = Integer.valueOf(params.get("pagesize").toString());

        return this.page(
            new Page<MessageEntity>(page, pagesize),
            queryWrapper
        );
    }
}

```

#### 查询条件：

```

/**
 * 组装消息查询条件
 * @param params
 * @return
 */
public static QueryWrapper getMessageCondition(Map<String, Object> params) {
    QueryWrapper<MessageEntity> queryWrapper = new
    QueryWrapper<MessageEntity>();

    try{
        String companyId = params.get("companyId") == null? null:
        params.get("companyId").toString().trim();
        if(companyId != null && !companyId.equals("NaN") &&
        !companyId.equals("")){
            queryWrapper.eq("companyId", companyId);
        }

        String metercode = params.get("metercode") == null? null :
        params.get("metercode").toString();
        if(metercode != null && !metercode.isEmpty()){
            queryWrapper.eq("meterCode",
            params.get("metercode").toString().trim());
        }

        String stTime = params.get("stTime") == null ? null :
        params.get("stTime").toString();
        if(stTime != null && !stTime.isEmpty()){
            queryWrapper.ge("curtime", formateDateTime(stTime));
        }

        String endTime = params.get("endTime") == null? null:
        params.get("endTime").toString();
        //        System.out.println("endTime" + endTime);
        if(endTime != null && !endTime.isEmpty()){
            queryWrapper.le("curtime", formateDateTime(endTime));
        }

        queryWrapper.orderByDesc("curtime");
    }
    catch(Exception e){
        e.printStackTrace();
    }

    return queryWrapper;
}

```

```
}
```

#### 分页：

```
@Configuration
public class MybatisPlusConfig {

    /**
     * 分页插件
     */
    @Bean
    public PaginationInterceptor paginationInterceptor() {

        return new PaginationInterceptor();
    }
}
```

#### Dao：

```
@Mapper
public interface MessageDao extends BaseMapper<MessageEntity> {
}
```

#### Entity：

```
@Data
@TableName("cz_message_realtime")
public class MessageEntity implements Serializable {
    private static final long serialVersionUID = 1L;

    /**
     * 设备编码
     */
    private String metercode;

    /**
     * 设备主键
     */
    private String meterid;

    /**
     * 设备名称
     */
    private String metername;

    ...
}
```

## 3 总结

- Flink项目开发过程
- Redis查询

- 实时指标的查询

