Javalin

Javalin

1

Javalin

https://javalin.io/documentation#getting-started

1 Javalin　　Maven

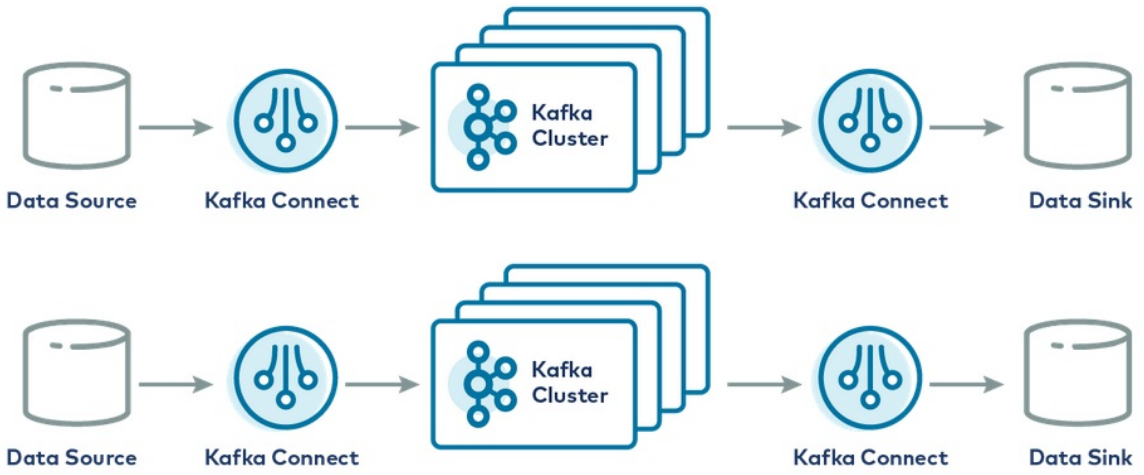
2 guice

Kafka Connect

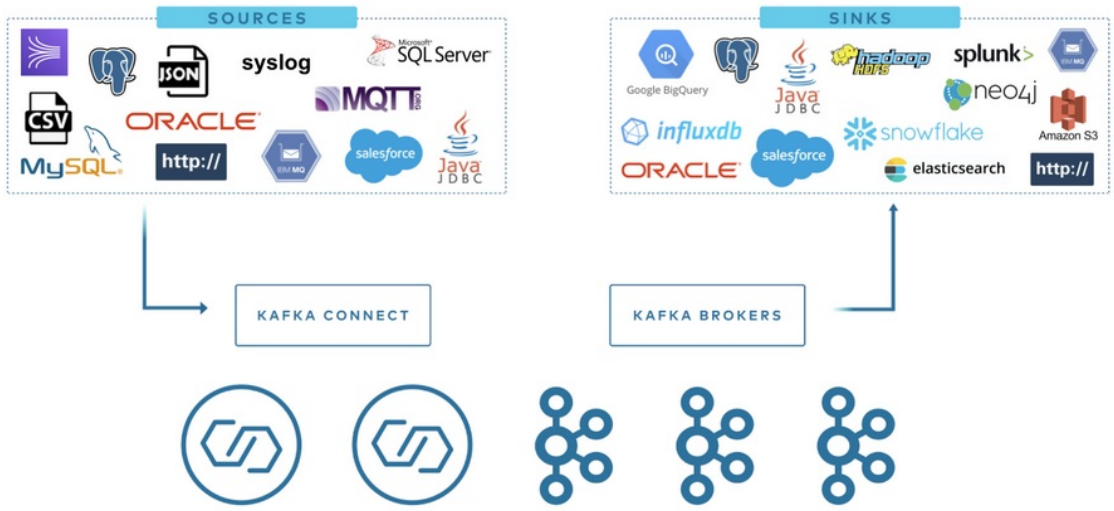
Kafka Connect

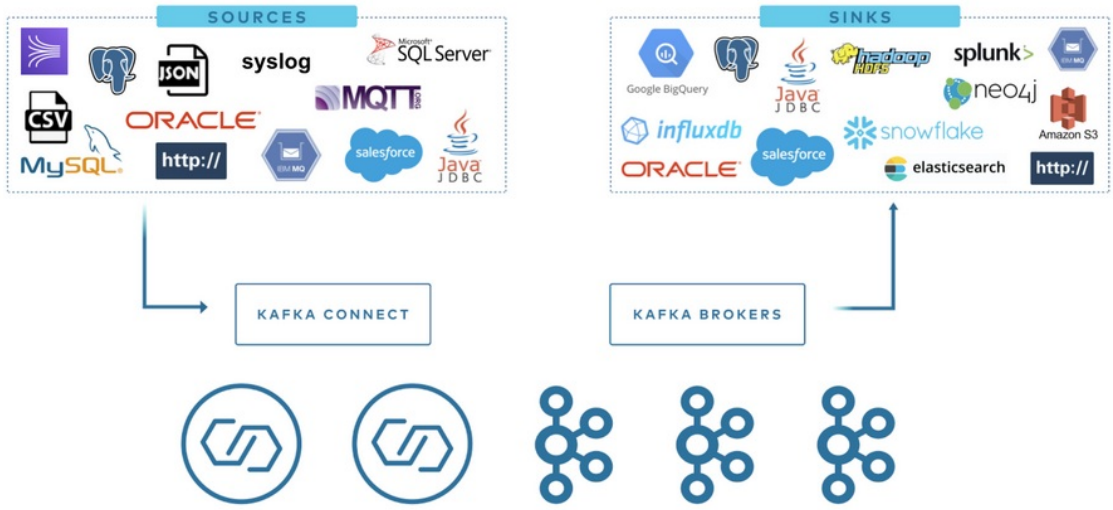
Kafka Connect

> Kafka Connect Apache Kafka Kafka Kafka Connect
> Kafka, Kafka Connect Kafka Source Connector, Sink Connector
> Source Connector Sink Connector Kafka BrokerSource Connector
> Kafka ConnectSink Connector Kafka Connect



Source Connector Kafka ConnectSink Connector Kafka Connect





> Kafka Connector Kafka Connect Schema
> Connector Kafka

Kafka data pipeline

- Kafka Databend Mysql Kafka
- Elasticsearch Kafka Kafka Elasticsearch Kafka

Kafka Connect

- Source Connect Kafka
- Sink Connect Kafka

> Kafka Confluent Hub Connector Elasticsearch Service Sink Connector, Amazon Sink Connector, HDFS Sink Connector Kafka

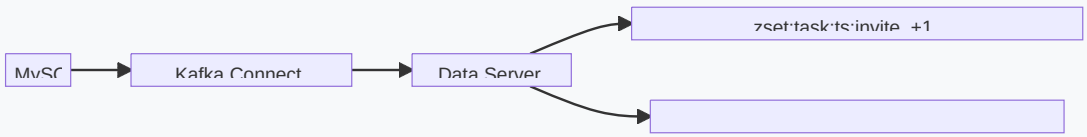
Kafka Connect ROI

Kafka Connect

> - 1 Kafka Connect MySQL topic
> - 2Data Server kafka-clients
>
> ```
> <!-- https://mvnrepository.com/artifact/org.apache.kafk
> ```

```
    a/kafka-clients -->
    <dependency>
        <groupId>org.apache.kafka</groupId>
        <artifactId>kafka-clients</artifactId>
        <version>3.8.0</version>
    </dependency>
```

3Data Server


zset:task:ts:invite_+1
MySQ
Kafka Connect
Data Server

- 4

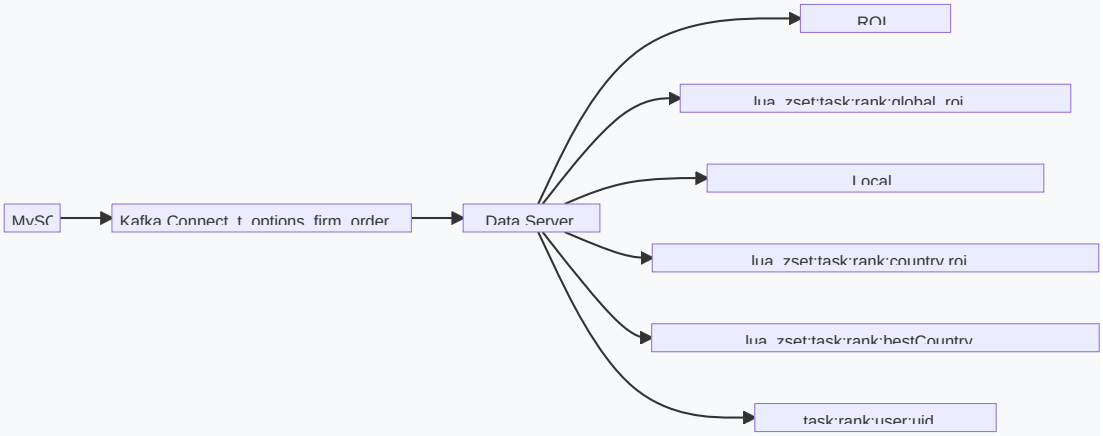
Kafka Connect ROI

ROI

- 1 Kafka Connect MySQL topic
- 2Data Server kafka-clients

```
<!-- https://mvnrepository.com/artifact/org.apache.kafk
a/kafka-clients -->
<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>3.8.0</version>
</dependency>
```

3Data Server ROI = profit_loss/(price*quantity)


MySC
Kafka Connect_t_options_firm_order
Data Server
ROI
lua_zset:task:rank:global_roi
Local
lua_zset:task:rank:country_roi
lua_zset:task:rank:bestCountry
task:rank:user:uid

> - 4

Data Server

Data Server

Redis

```
protected JedisPooled provideRedis() {
    var poolConfig = new GenericObjectPoolConfig<Connection>()
;
    //
    poolConfig.setMaxWaitMillis(config.getMaxWaitMillis());
    //
    poolConfig.setMaxTotal(config.getMaxTotal());
    //
    poolConfig.setMaxIdle(config.getMaxIdle());
    //
    poolConfig.setMinIdle(config.getMinIdle());
    // ( PoolableObjectFactory.validateObject() )
    poolConfig.setTestOnBorrow(config.isTestOnBorrow());
    // ( PoolableObjectFactory.validateObject() )
    poolConfig.setTestOnReturn(config.isTestOnReturn());
    //
    poolConfig.setTestWhileIdle(config.isTestWhileIdle());
    //
    poolConfig.setMinEvictableIdleTimeMillis(config.getMinEvic
tableIdleTimeMillis());
    //
    poolConfig.setTimeBetweenEvictionRunsMillis(config.getTime
BetweenEvictionRunsMillis());
    //
    poolConfig.setNumTestsPerEvictionRun(-1);

    var clientConfig = DefaultJedisClientConfig.builder()
        .user(config.getUser())
        .password(config.getPassword())
        .database(config.getDatabase())
        .build();
    var hostPort = new HostAndPort(config.getAddress(), config
.getPort());
    return new JedisPooled(poolConfig, hostPort, clientConfig)
;
}
```

- 1

- 2

Hikari

```
private DataSource createDatasource(JdbcConfig conf) {
    var config = new HikariConfig();
    config.setJdbcUrl(conf.getUrl());
    config.setUsername(conf.getUser());
    config.setPassword(conf.getPass());
    //config.setThreadFactory(Thread.ofVirtual().factory());
    config.setConnectionTestQuery(conf.getConnectionTestQuery(
));
    config.setMaxLifetime(conf.getMaxLifetime());
    config.setMaximumPoolSize(conf.getMaxPoolSize());
    config.setMinimumIdle(conf.getMinIdle());
    config.setIdleTimeout(conf.getIdleTimeout());
    return new HikariDataSource(config);
}
```

-
-

1

> “” inviteDao.aggInvites Const.INVITE_LEVELS

```
public void run() {
    var nowTs = DateUtil.current();
    var lastTs = redisDao.getLong(RedisKey.inviteTaskTs(), nul
l);

    var result = inviteDao.aggInvites(lastTs);
    var levels = Const.INVITE_LEVELS;
    for (var it : result.entrySet()) {
        var inviteCount = it.getValue().getCount();
        for (var n : levels) {
            if (inviteCount < n) continue;
            //n, ,
            achievementDao.add(it.getKey(), Const.ACH_INVITE,
n.toString(), inviteCount.toString());
            break;
        }
```

```
        }

        redisDao.setLong(RedisKey.inviteTaskTs(), nowTs);
    }
```

2ROI

> roiRank24h t_options_firm_ordert_user

```
SELECT * FROM
(
    SELECT
        {0}.id,
        {0}.user_id uid,
        {1}.user_name un,
        {1}.country ct,
        (profit_loss/(price*quantity)) roi,
        @rn := IF(@prev = user_id, @rn + 1, 1) AS rn,
        @prev := user_id
    FROM {0}
    JOIN (SELECT @prev := NULL, @rn := 0) AS vars
    LEFT JOIN {1} on {0}.user_id = {1}.id
    WHERE {2}
    ORDER BY {0}.user_id, roi desc
) AS T1
WHERE rn = 1
order by roi desc
```

> roiRank24h cacheRoiRank24hInRedis

-
- Redis
-

```
//
public void run() {
    var nowTs = DateUtil.current();
    var ranks = orderDao.roiRank24h(nowTs-Const.DAY_MS, config
.getBaseToken(), config.getQuoteToken());
    cacheDao.cacheRoiRank24hInRedis(ranks, 200);
}

//
public void cacheRoiRank24hInRedis(List<RankingVo> ranks, int
limit) {
```

```
    var countries = countryDao.all();
    var counter = new Counter();
    var countryMap = new HashMap<String, List<RankingVo>>();
    var bested = new HashSet<String>();
    var best = new ArrayList<RankingVo>();
    var global = new ArrayList<RankingVo>();
    for (var i = 0; i < ranks.size(); i++) {
        var it = ranks.get(i);
        it.setGr(i+1); //
        if(i < limit) global.add(it);

        var isWhite = countries.containsKey(it.getCt());
        var country = isWhite ? it.getCt() : "Local";

        var countryRank = counter.incr(country);
        it.setLr(countryRank);//
        if(!countryMap.containsKey(country)) {
            countryMap.put(country, new ArrayList<>());
        }

        var local = countryMap.get(country);
        if(local.size() <= limit) local.add(it);

        //best country
        if(isWhite && !bested.contains(country)) {
            bested.add(country);
            best.add(it);
        }

        worker.submit(() -> {
            redisDao.setJson(RedisKey.userRank(it.getUid()), i
t, config.getRankExpiresSec(), false);
        });
    }

    redisDao.setJson(RedisKey.globalRank(), global, config.get
RankExpiresSec(), true);
    redisDao.setJson(RedisKey.bestCountry(), best, config.getR
ankExpiresSec(), true);
    countryMap.forEach((name, rank) -> {
        redisDao.setJson(RedisKey.countryRank(name), rank, con
fig.getRankExpiresSec(), true);
    });
}
```

> zset
>
> - Kafka Connect MySQL Kafka MySQL >> Kafka >> Redis

- RocketMQ Connect MySQL RocketMQ MySQL >> Kafka >> Redis
- Canal MySQL binlog MySQL >> Canal >> Redis

redisDao.setJson() redis set json

- 50 >> >> >> top50, zset zset top50,

- zset

JavaEasyExcel

JavaEasyExcel

xcelEasyExcel

www.jb51.net/article...
github.com/alibaba/e...

```
const download = () => {
  axios({
    method: 'GET',
    url: config.http.baseUrl + '/templateDownload',
    responseType: 'blob',
  })
    .then(function (res) {
    const content = res.data
    const blob = new Blob([content], { type: "application/application/vnd
    const downloadElement = document.createElement("a");
    const href = window.URL.createObjectURL(blob);
    downloadElement.href = href;
    downloadElement.download = decodeURI(res.headers['filename']);
    document.body.appendChild(downloadElement);
    downloadElement.click();
    document.body.removeChild(downloadElement); //
    window.URL.revokeObjectURL(href); // blob
  })
}
```

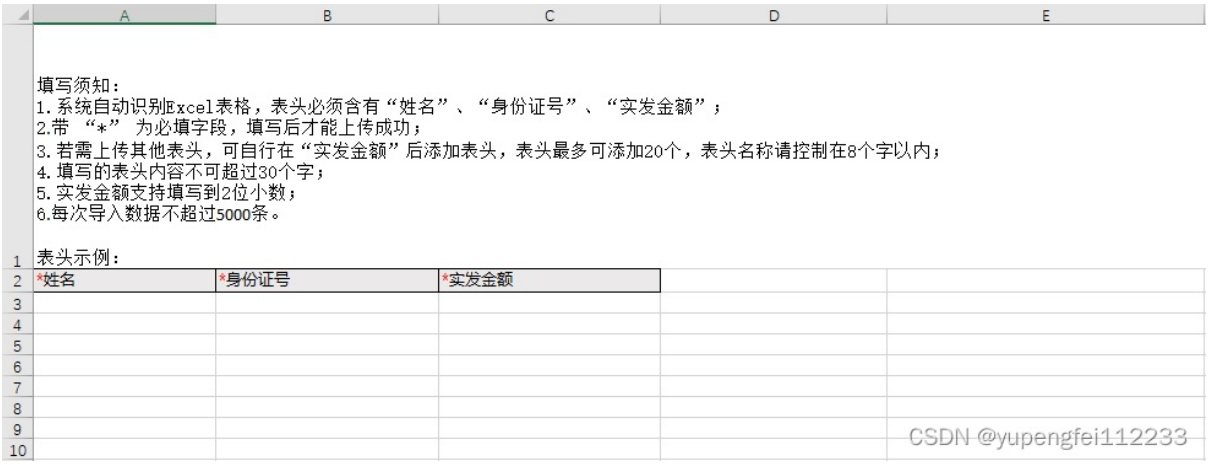
excelspringbootjar

```
@Override
public void templateDownload(HttpServletResponse response, HttpServletReq
    //
    String fileName = ".xlsx"      ;
    //
    String filePath = "/template/template.xlsx";
    TemplateDownloadUtil.download(response, request, fileName, filePath);
}
```

```
import lombok.extern.slf4j.Slf4j;
import org.springframework.core.io.ClassPathResource;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.URLEncoder;

/**
 *
 * @author
 * @date 2021/05/20 9:20
 */
@Slf4j
public class TemplateDownloadUtil {

    public static void download(HttpServletResponse response, HttpServletRequest
        try {
            response.setContentType("application/application/vnd.openxmlformats-offic
            response.setCharacterEncoding("utf-8");
            // URLEncoder.encode easyexcel
            response.setHeader("Content-Disposition", "attachment; filename="
            response.setHeader("filename", URLEncoder.encode(fileName, "UTF-8"
            response.setHeader("Access-Control-Expose-Headers", "filename,Content-Dis

            //
//            String filePath = getClass().getResource("/template/template.xlsx").get
//            FileInputStream input = new FileInputStream(filePath);

            //
            ClassPathResource resource = new ClassPathResource(filePath);
            InputStream input = resource.getInputStream();
            OutputStream out = response.getOutputStream();
            byte[] b = new byte[2048];
            int len;
            while ((len = input.read(b)) != -1) {
                out.write(b, 0, len);
            }
            // Excel“xxx.xlsx”""
//            response.setHeader("Content-Length", String.valueOf(input.getChannel().
            input.close();
        } catch (Exception e) {
            log.error(" :"      , e);
        }
    }
}
```

EasyExcel

EasyExcel



	A	B	C	D	E
	填写须知： 1.系统自动识别Excel表格，表头必须含有“姓名”、“身份证号”、“实发金额”； 2.带 “*” 为必填字段，填写后才能上传成功； 3.若需上传其他表头，可自行在“实发金额”后添加表头，表头最多可添加20个，表头名称请控制在8个字以内； 4.填写的表头内容不可超过30个字； 5.实发金额支持填写到2位小数； 6.每次导入数据不超过5000条。				
1	表头示例：				
2	*姓名	*身份证号	*实发金额		
3					
4					
5					
6					
7					
8					
9					
10					


```
import com.alibaba.excel.context.AnalysisContext;
import com.alibaba.excel.event.AnalysisEventListener;
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import lombok.Data;
import lombok.extern.slf4j.Slf4j;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

/**
 * excel
 *
 * @author yupf
 * @description Listener springexcelnew,spring
 */
@Slf4j
@Data
public class BatchReadListener extends AnalysisEventListener<Map<Integer,

    /**
     * 500list
     */
    private static final int BATCH_COUNT = 500;
    //Excel
    private List<Map<Integer, Map<Integer, String>>> list = new ArrayList
    //Excel
    private Map<Integer, String> headTitleMap = new HashMap<>();


    /**
     * DAOservice
```

```
     */
    private DbFileBatchService dbFileBatchService;
    private DbFileContentService dbFileContentService;
    private FileBatch fileBatch;
    private int total = 0;

    /**
     * spring,Listenerspring
     */
    public BatchReadListener(DbFileBatchService dbFileBatchService, DbFileContentServ
        this.dbFileBatchService = dbFileBatchService;
        this.dbFileContentService = dbFileContentService;
        this.fileBatch = fileBatch;
    }

    /**
     *
     *
     * @param data    one row value. Is is same as {@link AnalysisContext#readRowHold
     * @param context
     */
    @Override
    public void invoke(Map<Integer, String> data, AnalysisContext context
        log.info(":{}"       , JSON.toJSONString(data));
        total++;
        Map<Integer, Map<Integer, String>> map = new HashMap<>();
        map.put(context.readRowHolder().getRowIndex(), data);
        list.add(map);
        // BATCH_COUNTOOM
        if (list.size() >= BATCH_COUNT) {
            saveData();
            //  list
            list.clear();
        }
    }

    /**
     *
     *
     * @param context
     */
    @Override
    public void doAfterAllAnalysed(AnalysisContext context) {
        //
        saveData();
        log.info(""         );
    }

    /**
     *
     **/
    @Override
    public void invokeHeadMap(Map<Integer, String> headMap, AnalysisContext
        log.info("{}"     , JSONObject.toJSONString(headMap));
        headTitleMap = headMap;
    }

    /**
     *
     */
```

```
    private void saveData() {
        log.info("{}"            , list.size());
        FileContent fileContent = null;
        List<FileContent> fileContentList = list.stream().flatMap(
            integerMap -> integerMap.entrySet().stream().map(entrySet ->
                //entrySet.getKey()RowIndex,
                Integer rowIndex = entrySet.getKey();
                Map<Integer, String> value = entrySet.getValue();
                log.info(JSONObject.toJSONString(value));
                fileContent = new FileContent();
                fileContent.setBatchId(fileBatch.getId());
                fileContent.setBatchNo(fileBatch.getBatchNo());
                //
                fileContent.setName(value.get(0) != null ? value.get(0).trim
                fileContent.setCertNo(value.get(1) != null ? value.get(1)
                fileContent.setRealAmount(value.get(2) != null ? value.get
                //JSON
                fileContent.setFieldsValue(JSONObject.toJSONString(value)
                //rowIndex
                fileContent.setRowNum(rowIndex + 1);
                fileContent.setCreateTime(LocalDateTime.now());
                return xcSalaryFileContent;
            }
        )).collect(Collectors.toList());
        log.info(JSONObject.toJSONString(fileContentList));
        dbFileContentService.saveBatch(fileContentList);
        log.info(""        );
    }
}
```

```
BatchReadListener listener = new BatchReadListener(dbFileBatchService, dbFileContentS
try {
    //headRowNumber12
    EasyExcel.read(fileInputStream, listener).headRowNumber(2).sheet().doRead
} catch (Exception e) {
    log.info("EasyExcel{}"       , e);
    throw new CustomException(""            );
}
//
Map<Integer, String> headTitleMap = listener.getHeadTitleMap();
//
List<String> headList = headTitleMap.keySet().stream().map(key -> {
    String head = headTitleMap.get(key);
    log.info(head);
    return head == null ? "" : head.replace("*", "");
}).collect(Collectors.toList());
//
```

EasyExcel

```
private List<List<String>> getFileHeadList( FileBatch fileBatch) {
    String head = fileBatch.getFileHead();
    List<String> headList = Arrays.asList(head.split(","));
    List<List<String>> fileHead = headList.stream().map(item ->     concatHead
    fileHead.add(concatHead(Lists.newArrayList(""  )));
    return fileHead;
}

/**
 *
 * @param headContent
 * @return
 */
private List<String> concatHead(List<String> headContent) {
    String remake = "
            "1.Excel“”“”“”\n"                                    +
            "2. “*” \n"                  +
            "3.“”208\n"                                                 +
            "4.30\n"              +
            "5.2\n"             +
            "6.5000\n"           +
            "\n" +
            "\n"                         +
            "\n" +
            ""     ;
    headContent.add(0, remake);
    return headContent;
}
```

```
List<FileContent> fileContentList = dbFileContentService.list(
    Wrappers.<FileContent>lambdaQuery()
    .eq(FileContent::getBatchId, fileBatch.getId())
    .orderByAsc(FileContent::getRowNum)
);
List<List<Object>> contentList = fileContentList.stream().map(fileContent
    List<Object> rowList = new ArrayList<>();
    String fieldsValue = fileContent.getFieldsValue();
    JSONObject contentObj = JSONObject.parseObject(fieldsValue);
    for (int columnIndex = 0 , length = headList.size(); columnIndex < length
        Object content = contentObj.get(columnIndex);
        rowList.add(content == null ? "" : content);
    }
    rowList.add(fileContent.getCheckMessage());
    return rowList;
}).collect(Collectors.toList());
```

```
import com.alibaba.excel.metadata.data.DataFormatData;
import com.alibaba.excel.metadata.data.WriteCellData;
import com.alibaba.excel.write.handler.context.CellWriteHandlerContext;
import com.alibaba.excel.write.metadata.style.WriteCellStyle;
import com.alibaba.excel.write.metadata.style.WriteFont;
import com.alibaba.excel.write.style.HorizontalCellStyleStrategy;
import org.apache.poi.ss.usermodel.BorderStyle;
import org.apache.poi.ss.usermodel.HorizontalAlignment;
import org.apache.poi.ss.usermodel.IndexedColors;

import java.util.List;

/**
 *
 */
public class CellStyleStrategy extends HorizontalCellStyleStrategy {

    private final WriteCellStyle headWriteCellStyle;
    private final WriteCellStyle contentWriteCellStyle;

    /**
     *
     */
    private final List<Integer> columnIndexes;

    public CellStyleStrategy(List<Integer> columnIndexes,WriteCellStyle headWriteCell
        this.columnIndexes = columnIndexes;
        this.headWriteCellStyle = headWriteCellStyle;
        this.contentWriteCellStyle = contentWriteCellStyle;
    }

    //
    @Override
    protected void setHeadCellStyle( CellWriteHandlerContext context) {
        //
        WriteFont headWriteFont = new WriteFont();
        headWriteFont.setFontName(""  );
        //
        if (columnIndexes.get(0).equals(context.getRowIndex())) {
            headWriteCellStyle.setFillForegroundColor(IndexedColors.WHITE
            headWriteCellStyle.setHorizontalAlignment(HorizontalAlignment
            headWriteFont.setFontHeightInPoints((short) 12);
            headWriteFont.setBold(false);
            headWriteFont.setFontName(""  );
        }else{
            headWriteCellStyle.setFillForegroundColor(IndexedColors.GREY_25_PERCENT
            headWriteCellStyle.setHorizontalAlignment(HorizontalAlignment
            headWriteFont.setFontHeightInPoints((short) 11);
            headWriteFont.setBold(false);
            headWriteFont.setFontName(""    );
        }
        headWriteCellStyle.setWriteFont(headWriteFont);
        DataFormatData dataFormatData = new DataFormatData();
        dataFormatData.setIndex((short)49);
        headWriteCellStyle.setDataFormatData(dataFormatData);
        if (stopProcessing(context)) {
            return;
        }
        WriteCellData<?> cellData = context.getFirstCellData();
        WriteCellStyle.merge(headWriteCellStyle, cellData.getOrCreateStyle
```

```
        }

        //
        @Override
        protected void setContentCellStyle(CellWriteHandlerContext context) {
            WriteFont contentWriteFont = new WriteFont();
            contentWriteFont.setFontName(""  );
            contentWriteFont.setFontHeightInPoints((short) 11);
            //
            contentWriteCellStyle.setWriteFont(contentWriteFont);
            contentWriteCellStyle.setBorderLeft(BorderStyle.THIN);
            contentWriteCellStyle.setBorderTop(BorderStyle.THIN);
            contentWriteCellStyle.setBorderRight(BorderStyle.THIN);
            contentWriteCellStyle.setBorderBottom(BorderStyle.THIN);
            DataFormatData dataFormatData = new DataFormatData();
            dataFormatData.setIndex((short)49);
            contentWriteCellStyle.setDataFormatData(dataFormatData);
            contentWriteCellStyle.setHorizontalAlignment(HorizontalAlignment.CENTER
            WriteCellData<?> cellData = context.getFirstCellData();
            WriteCellStyle.merge(contentWriteCellStyle, cellData.getOrCreateStyle
        }
    }
```

```
    import com.alibaba.excel.write.style.row.AbstractRowHeightStyleStrategy;
    import org.apache.poi.ss.usermodel.Row;

    /**
     *
     */
    public class CellRowHeightStyleStrategy extends AbstractRowHeightStyleStrategy

        @Override
        protected void setHeadColumnHeight(Row row, int relativeRowIndex) {
            //17.7
            if(relativeRowIndex == 0){
                //excel1515*20=300
                row.setHeight((short) 3240);
            }
        }

        @Override
        protected void setContentColumnHeight(Row row, int relativeRowIndex)
        }
    }
```

EasyExcelLongestMatchColumnWidthStyleStrategy()

```
    EasyExcel.write(fileName, LongestMatchColumnWidthData.class)
        .registerWriteHandler(new LongestMatchColumnWidthStyleStrategy()).sheet
```

```
import com.alibaba.excel.enums.CellDataTypeEnum;
import com.alibaba.excel.metadata.Head;
import com.alibaba.excel.metadata.data.CellData;
import com.alibaba.excel.metadata.data.WriteCellData;
import com.alibaba.excel.write.metadata.holder.WriteSheetHolder;
import com.alibaba.excel.write.style.column.AbstractColumnWidthStyleStrategy
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.collections.CollectionUtils;
import org.apache.poi.ss.usermodel.Cell;

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

/**
 * @author yupf
 * @description
 * @date 2022/9/7 18:48
 */
@Slf4j
public class CellWidthStyleStrategy extends AbstractColumnWidthStyleStrategy
    private Map<Integer, Map<Integer, Integer>> CACHE = new HashMap<>();

    @Override
    protected void setColumnWidth(WriteSheetHolder writeSheetHolder, List
        Map<Integer, Integer> maxColumnWidthMap = CACHE.get(writeSheetHolder
        if (maxColumnWidthMap == null) {
            maxColumnWidthMap = new HashMap<>();
            CACHE.put(writeSheetHolder.getSheetNo(), maxColumnWidthMap);
        }
        if (isHead) {
            if(relativeRowIndex.intValue() == 1){
                Integer length = cell.getStringCellValue().getBytes().length
                Integer maxColumnWidth = maxColumnWidthMap.get(cell.getColumnIndex
                if (maxColumnWidth == null || length > maxColumnWidth) {
                    maxColumnWidthMap.put(cell.getColumnIndex(), length);
                    writeSheetHolder.getSheet().setColumnWidth(cell.getColumnIndex
                }
            }
        }else{
            Integer columnWidth = this.dataLength(cellDataList, cell, isHead
            if (columnWidth >= 0) {
                if (columnWidth > 255) {
                    columnWidth = 255;
                }
                Integer maxColumnWidth = maxColumnWidthMap.get(cell.getColumnIndex
                if (maxColumnWidth == null || columnWidth > maxColumnWidth
                    maxColumnWidthMap.put(cell.getColumnIndex(), columnWidth
                    writeSheetHolder.getSheet().setColumnWidth(cell.getColumnIndex
                }
            }
        }
    }
```

```
        private Integer dataLength(List<WriteCellData<?>> cellDataList, Cell cell
            if (isHead) {
                return cell.getStringCellValue().getBytes().length;
            } else {
                CellData cellData = cellDataList.get(0);
                CellDataTypeEnum type = cellData.getType();
                if (type == null) {
                    return -1;
                } else {
                    switch (type) {
                        case STRING:
                            return cellData.getStringValue().getBytes().length
                        case BOOLEAN:
                            return cellData.getBooleanValue().toString().getBytes
                        case NUMBER:
                            return cellData.getNumberValue().toString().getBytes
                        default:
                            return -1;
                    }
                }
            }
        }
    }
```

```
    EasyExcel.write(response.getOutputStream())
        .head(head)
        .registerWriteHandler(new CellRowHeightStyleStrategy())   //
        .registerWriteHandler(new CellStyleStrategy(Arrays.asList(0,1),new WriteCellStyle
        .registerWriteHandler(new CellWidthStyleStrategy())
        .sheet(sheetName)
        .doWrite(list);
```

EasyExcel

EasyExcelapi

JavaEasyExcel,EasyExcel
:

```
    JavaeasyExcelexcel
    JavaEasyExcel
    JavaEasyexcel
    JavaEasyExcel
    JavaEasyExcel
    Java easyExcel
    JavaEasyExcelExcel
    Java EasyExcelsheet
    Java EasyExcel
```

```
JavaExcel(EasyExcel)
```

EasyExcel