ExcelShift 使用文档

1. 简介

ExcelShift是一个基于配置的excel数据提取框架，该框架支持多种数据提取模式，包括单个对象、普通列表、行组列表以及垂直列表等。

2. 项目结构

```
ExcelShift4/
├── src/
|   └── main/
|       ├── java/
|       |   └── com/
|       |       └── excel/
|       |           └── shift/
|       |               ├── excel/          # 核心提取器实现
|       |               ├── model/          # 数据模型类
|       |               ├── result/         # 结果处理类
|       |               └── util/           # 工具类
|       └── resources/                      # 配置文件目录
└── pom.xml                                 # 项目依赖管理
```

3. 快速开始

3.1 添加依赖

在您的`pom.xml`中添加以下依赖：

```xml
<dependency>
    <groupId>com.excel.shift</groupId>
    <artifactId>ExcelShift</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

3.2 创建配置文件

首先创建一个JSON配置文件来定义数据映射规则。例如：

```
{
  "wellTestData": {
    "targetClass": "com.your.package.WellTestData",
    "description": "井测试数据提取配置",
    "order": 1,
    "resultType": "LIST",    // 代表返回类型为list类型
    "startRow": "5",
    "startColumn": "A",
    "endRow": null,
    "table": {
      "columns": {
        "depthMd": {
```

```
          "order": 1,
          "javaFieldName": "depthMd",
          "javaFieldType": "Double",
          "columnCell": "E",
          "description": "深度(MD) m"
        },
        "sandBody": {
          "order": 2,
          "javaFieldName": "sandBody",
          "javaFieldType": "String",
          "columnCell": "C",
          "description": "砂体",
          "isMergeType": true
        }
      }
    }
  }
}
```

3.3 创建数据模型类

创建与配置对应的Java实体类：

```java
@Data
public class WellTestData {
    private Double depthMd;   // 深度(MD)
    private String sandBody;  // 砂体
    // 其他字段...
}
```

3.4 使用ExcelExtractor提取数据

```java
// 指定Excel文件路径
String excelPath = "path/to/your/excel/file.xlsx";
// 配置文件路径
String configPath = "path/to/your/config.json";

// 需要提取的类列表
List<Class<?>> classList = new ArrayList<>();
classList.add(WellTestData.class);

// 创建提取器
ExcelExtractor extractor = new ExcelExtractor(excelPath, configPath, classList);

// 提取指定工作表的数据
SheetExtractionResult result = extractor.extractSheetByIndex(0); // 解析第一个excel的sheet

// 获取提取结果
if (result.isSuccess()) {
    List<WellTestData> dataList = result.getResultList(WellTestData.class);
  // 如果是single类型的类，那么使用getResult 来接受单个对象。
    // 处理提取的数据
```

```
    dataList.forEach(System.out::println);
}

// 或者提取excel所有工作表数据 适用于与当前配置一样的sheet否则只有部分成功。
ExtractionResult allResults = extractor.extractAllSheet();
```

4. 配置文件详解

4.1 提取器类型

ExcelShift支持四种提取器类型说明我们以这个日报举例说明：

XXXX-X-X井地质日报

CNOOC？？公司　　早报NO.9

井名：XXXX-X-X	作业时间　3月23日06:00~3月24日06:00	汇报日期：2019年3月24日
井　　深：1634.00m	海水深度：87.40m，　补心高：25.00m，　上层套管：339.72mm，　套管鞋深度：972.51m，　当前井眼：　311.15mm	
日 进 尺：639　　钻进时间：13.51h	泥浆类型：PDF-PLUS/KCL ，　比重(g/cm³)：1.12，　粘度(sec/qt)：41 ，　塑性粘度(mPa·s)：17	
钻　　速：47.3m/h　迟到深度：1579.00	失水(ml/30min)：5.6 ，　含油(%)：　0，　含水(%)：94 ，　含砂(%)：0.2，　Cl-(mg/l)：38000	

层段（m）		岩性描述	深度	全烃	气测显示 (ppm)						荧光	
顶	底		(m)	(%)	C_1	C_2	C_3	iC_4	nC_4	CO_2(%)	直照	滴照
990	1075	浅灰色中厚-厚层含砾粗砂岩、粗砂岩与中厚层灰色泥岩呈略等厚互层，局部见薄层灰白色灰质粉砂岩。	1295	2.8031	2.5463	0.0921	0.0114	0.0037	0.0074	0.03	暗黄色	乳白色
1075	1579	浅灰色薄-厚层含砾粗砂岩、粗砂岩、中砂岩、细砂岩与中厚-厚层灰色泥岩呈不等厚互层，局部见薄层灰白色灰质粉砂岩。	1545	3.5053	1.9404	0.1415	0.1086	0.0325	0.0556	0.03	暗黄色	乳白色

作业概况：

3月23日　06:00~06:30 循环；

06:30~07:30 上提至套管鞋内，地漏试验：井深995.00m，泥浆密度1.07g/cm³，泵压760psi，地层破裂，破裂当量钻井液密度1.62g/cm³，地层漏失压力590psi，漏失当量钻井液密度1.50g/cm³；

07:30~24:00 钻进311.15mm井眼自995.00m至1493.00m。

3月24日　00:00~06:00 钻311.15mm井眼自1493.00m至1634.00m。

井斜数据

类型	深度(m)	井斜(°)	方位(°)
多点	984.08	0.50	61.88
	1012.44	0.82	40.05
	1042.40	0.95	39.18
	1071.62	1.39	19.58
	1100.89	1.40	22.03
	1130.11	1.35	35.62
	1159.56	1.20	30.92
	1187.90	1.30	31.79
	1216.97	1.44	28.97
	1246.48	1.39	28.05
	1275.48	1.54	31.54
	1304.50	1.50	40.75
	1333.83	1.26	41.31
		1.32	47.66

作业计划：继续钻进311.15mm井眼，循环，投测多点，短起钻，回收多点，下钻至井底，继续钻进

主要检测项目	校验日期	工作状况	主要选择项目	校验日期	工作状况
RESERVAL	3月19日	正常	FLAIR	3月22日	正常
硫化氢检测	3月19日	正常	地化录井	3月20日	正常
二氧化碳检测	3月19日	正常	X衍射全岩分析录井	3月20日	正常
三维定量荧光	3月20日	正常			

？？？公司勘探部

地质总监：？？？

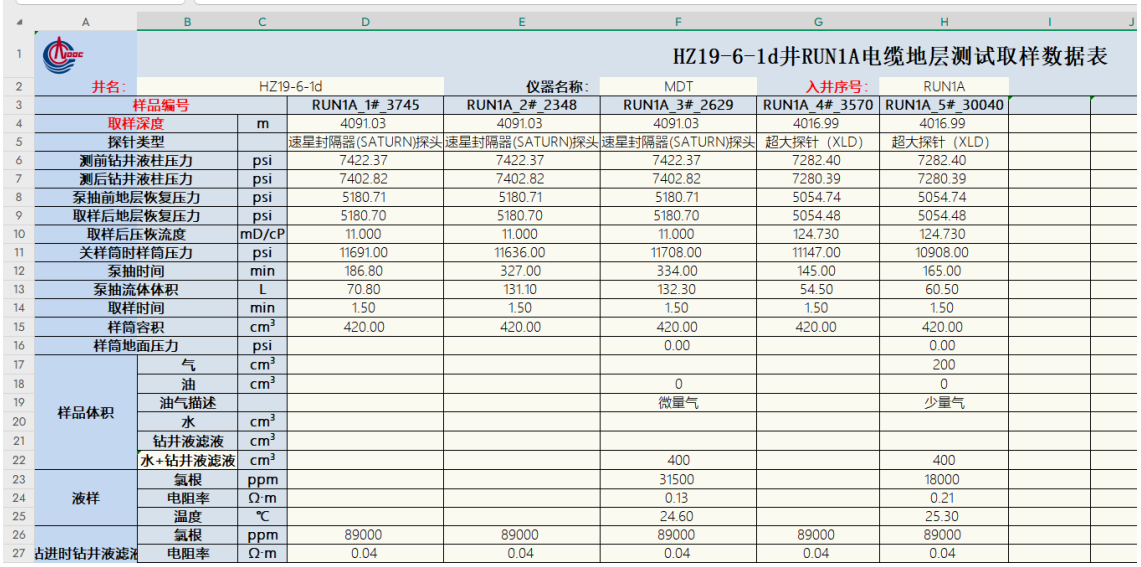
地质监督：？？？

- **SINGLE**：提取单个对象，如表头、表格概述等 例如标题、某某公司等信息
- **LIST**：提取普通表格数据，每行对应一个对象 例如上面图中的井斜数据、作业计划等等
- **GROUP_LIST**：提取行组数据，多行组成一个对象 如下　4、5、6 三行组成一个逻辑上的对象。

HZ19-6-1d井录井油气显示表

序号	井段md (m)		厚度 (m)	井段Tvd (m)		垂厚 (m)	层位	岩石定名		钻时 (min/m)	气测值(%)									荧光			岩样调照		含油产状	槽面显示
								颜色	岩性		全量	C1	C2	C3	iC4	nC4	iC5	nC5	CO2	面积	级别	颜色	反应	颜色		
1	2720.00	2726.00	6.00	2626.16	2631.83	5.67	珠江组	浅灰色	荧光泥质粉砂岩	2.79 ~ 5.01	0.1015 ~ 0.1311	0.0099 ~ 0.0200	0.0013 ~ 0.0020	0.0010 ~ 0.0022	0.0007 ~ 0.0017	0.0004 ~ 0.0008	0.0006 ~ 0.0011	0.0004 ~ 0.0007	0.01	10	D	暗黄色	慢速	乳白色	荧光	无
2	2739.00	2740.00	1.00	2644.07	2645.01	0.94	珠江组	浅灰色	荧光泥质粉砂岩	2.79 ~ 5.01	0.1071 ~ 0.1842	0.0115 ~ 0.0334	0.0010 ~ 0.0039	0.0011 ~ 0.0051	0.0009 ~ 0.0037	0.0005 ~ 0.0019	0.0006 ~ 0.0024	0.0006 ~ 0.0011	0.01	5	D	暗黄色	慢速	乳白色	荧光	无
3	2771.00	2775.00	4.00	2674.18	2677.94	3.76	珠江组	浅灰色	荧光泥质粉砂岩	2.48 ~	0.1210 ~	0.0143 ~	0.0015 ~	0.0016 ~	0.0013 ~	0.0007 ~	0.0009 ~	0.0006 ~	0.01	30	C	亮黄色	中速	乳白色	荧光	无

- **VERTICAL_LIST**：提取垂直表格数据，每列对应一个对象，每行是对象的不同属性 如下图



HZ19-6-1d井RUN1A电缆地层测试取样数据表

井名:		HZ19-6-1d		仪器名称:	MDT	入井序号:	RUN1A	
样品编号			RUN1A_1#_3745	RUN1A_2#_2348	RUN1A_3#_2629	RUN1A_4#_3570	RUN1A_5#_30040	
取样深度		m	4091.03	4091.03	4091.03	4016.99	4016.99	
探针类型			速星封隔器(SATURN)探头	速星封隔器(SATURN)探头	速星封隔器(SATURN)探头	超大探针（XLD）	超大探针（XLD）	
测前钻井液柱压力		psi	7422.37	7422.37	7422.37	7282.40	7282.40	
测后钻井液柱压力		psi	7402.82	7402.82	7402.82	7280.39	7280.39	
泵抽前地层恢复压力		psi	5180.71	5180.71	5180.71	5054.74	5054.74	
取样后地层恢复压力		psi	5180.70	5180.70	5180.70	5054.48	5054.48	
取样后压恢流度		mD/cP	11.000	11.000	11.000	124.730	124.730	
关样筒时样筒压力		psi	11691.00	11636.00	11708.00	11147.00	10908.00	
泵抽时间		min	186.80	327.00	334.00	145.00	165.00	
泵抽流体体积		L	70.80	131.10	132.30	54.50	60.50	
取样时间		min	1.50	1.50	1.50	1.50	1.50	
样筒容积		cm^3	420.00	420.00	420.00	420.00	420.00	
样筒地面压力		psi			0.00		0.00	
样品体积	气	cm^3					200	
	油	cm^3			0		0	
	油气描述				微量气		少量气	
	水	cm^3						
	钻井液滤液	cm^3						
	水+钻井液滤液	cm^3			400		400	
液样	氯根	ppm			31500		18000	
	电阻率	Ω·m			0.13		0.21	
	温度	℃			24.60		25.30	
	氯根	ppm	89000	89000	89000	89000	89000	
占进时钻井液滤液	电阻率	Ω·m	0.04	0.04	0.04	0.04	0.04	

4.2 配置项说明

配置文件的完整示例结构如下，一个配置文件可以包含多个单个类的配置：

```
{
  "extractorId": {                    // 提取器唯一标识 用于动态表达式计算的引用
    "targetClass": "com.example.Data", // 目标Java类的全限定名，必填
    "description": "配置描述",          // 配置描述，用于说明该提取器的用途
    "order": 1,                       // 提取器执行顺序，必填，数字越小越先执行
    "resultType": "LIST",             // 结果类型，必填，可选值：SINGLE/LIST/GROUP_LIST/VERTICAL_LIST  一个类只能选则一种
     // 注意:startRow是以直接数据行作为设置的 例如数据从第一行开始那么startrow为1 而不是他的表头行。其余定位也是如此以数据行作为起始来设置位置信息。
    "startRow": "5",                  // 开始行号，从1开始计数 如果通过startFlag进行确定，则设置为null  否则必填 可以填动态表达式
    "startColumn": "A",               // 开始列名，使用Excel列名(A,B,C...)
    "endRow": null,                   // 结束行号，可选，null表示动态计算 如果通过endFlag进行确定，则设置为null 否则必填 可以填动态表达式
    "endColumn": null,                // 结束列名，可选，用于VERTICAL_LIST类型
    "isDynamic": true,                // 类里面是否包含动态表达式的计算 例如{operationRecord1.endRow + 1} 默认false
    "isDynamicRows": true,            // 表格结束行数不确定并且下方 包含其他的结构需要结合endFlag，startflag来使用来确定endRow或者startRow ，startFlag和endFlag 其中之一不为null则该值必须为true
    "groupRowCount": 3,               // 每组包含的行数，仅用于GROUP_LIST类型
    "startFlag": {                    // 开始标志配置，用于动态范围识别起始行 结合isDynamicRows使用
      "text": "作业概况",              // 标志文本
      "columnCell": "B"               // 标志所在列
    },
    "endFlag": {                      // 结束标志配置，用于动态范围识别结束行 结合isDynamicRows使用
      "text": "作业计划",              // 标志文本
      "columnCell": "B"               // 标志所在列
    },
    "fields": {                       // SINGLE类型的字段映射
      "fieldName": {    // 对应的实体类属性名最好和下面javaFieldName 保持一致
        "order": 1,                   // 字段顺序，必填
        "javaFieldName": "fieldName", // Java类中的字段名，必填
```

```
        "javaFieldType": "String",     // Java字段类型，必填
        "excelCell": "B4",             // Excel单元格坐标，必填 可以包含动态表达式例如"C${operationRecord1.endRow + 1}",
        "description": "字段描述",     // 字段描述
        "extractPattern": "井名：(.*)", // 正则表达式提取模式
        "defaultValue": ""             // 默认值
      }
      ...   // 下面还可以包含其余类似的filed字段
    },
    "table": {                         // LIST/GROUP_LIST/VERTICAL_LIST类型的字段映射
      "columns": {
        "fieldName": { // 对应的实体类属性名最好和下面javaFieldName 保持一致
          "order": 1,                  // 字段顺序，必填
          "javaFieldName": "fieldName", // Java类中的字段名，必填
          "javaFieldType": "String",   // Java字段类型，必填
          "columnCell": "A",           // Excel列名，用于LIST和GROUP_LIST类型
          "rowCell": "3",              // Excel行号，用于VERTICAL_LIST类型    用于确定垂直的excel的字段的行
          "groupRowIndex": 1,          // 组内行索引，仅用于GROUP_LIST类型 表示在组内的第几行取数据
          "description": "字段描述",   // 字段描述
          "unit": "m",                 // 单位
          "extractPattern": "正则表达式", // 正则表达式提取模式
          "defaultValue": "默认值",    // 默认值
          "isMergeType": true          // 是否为合并单元格    如果是合并单元格并且当前数据null，就会默认向上查询到第一个值作为当前的值
        }
      }
    }
  }
 "extractor2": {    // 第二个类结构配置与上面类似，下面还可以定义其余类的配置
    ...
    }
}
```

5. 高级功能

5.1 合并单元格处理

通过设置`isMergeType: true`，ExcelShift可自动处理合并单元格，从上方单元格提取数据。

5.2 正则表达式提取

使用`extractPattern`配置项可从单元格文本中提取指定模式的数据：

```
"extractPattern": "井\\s*深：(\\d+\\.?\\d*)m"
```

5.3 动态范围识别

可通过设置开始和结束标志自动识别数据范围：

```
"startFlag": {
  "text": "作业概况",
  "columnCell": "B"
},
"endFlag": {
  "text": "作业计划",
  "columnCell": "B"
}
```

5.4外部创建配置对象然后传入

除了通过配置文件路径创建提取器外，还可以预先创建和修改配置对象：

```
// 创建配置对象
ExcelMappingConfig config = new ExcelMappingConfig(configPath);
// 可以对配置进行修改
// ...

// 使用配置对象创建提取器
ExcelExtractor extractor = new ExcelExtractor(excelPath, config, classList);
```

6. 支持的数据类型

ExcelShift支持以下Java数据类型的自动转换：

- `String`：字符串
- `Integer/int`：整数
- `Long/long`：长整数
- `Double/double`：双精度浮点数
- `Float/float`：单精度浮点数
- `Boolean/boolean`：布尔值

7. 常见使用场景

7.1 提取表头数据或单个对象(SINGLE)

```
{
  "reportHeader": {
    "targetClass": "com.example.ReportHeader",
    "resultType": "SINGLE",
    "fields": {
      "wellName": {
        "javaFieldName": "wellName",
        "javaFieldType": "String",
        "excelCell": "B4",
        "extractPattern": "井名：(.*)"
      }
    }
  }
}
```

7.2 提取表格数据(LIST)

```
{
  "gasTestData": {
    "targetClass": "com.example.GasTestData",
    "resultType": "LIST",
    "startRow": "10",
    "table": {
      "columns": {
        "depth": {
          "javaFieldName": "depth",
          "javaFieldType": "Double",
          "columnCell": "A"
        },
        "totalGas": {
          "javaFieldName": "totalGas",
          "javaFieldType": "Double",
          "columnCell": "B"
        }
      }
    }
  }
}
```

7.3 提取行组数据(GROUP_LIST)

```
{
  "wellLoggingData": {
    "targetClass": "com.example.WellLoggingData",
    "resultType": "GROUP_LIST",
    "startRow": "5",
    "groupRowCount": 3,
    "table": {
      "columns": {
        "depthStart": {
          "javaFieldName": "depthStart",
          "javaFieldType": "Double",
          "columnCell": "C",
          "groupRowIndex": 1
        },
        "c1Min": {
          "javaFieldName": "c1Min",
          "javaFieldType": "Double",
          "columnCell": "H",
          "groupRowIndex": 2
        }
      }
    }
  }
}
```

7.4 提取垂直表格数据(VERTICAL_LIST)

```
{
  "sampleData": {
    "targetClass": "com.example.SampleData",
    "resultType": "VERTICAL_LIST",
    "startRow": "3",
    "startColumn": "D",
    "table": {
      "columns": {
        "sampleCode": {
          "javaFieldName": "sampleCode",
          "javaFieldType": "String",
          "rowCell": "3"
        },
        "sampleDepth": {
          "javaFieldName": "sampleDepth",
          "javaFieldType": "Double",
          "rowCell": "4"
        }
      }
    }
  }
}
```

8. 实际应用示例

8.1 提取钻井报告数据

```
public class DrillReportExample {
    public static void main(String[] args) {
        String excelPath = "path/to/drill_report.xls";
        String configPath = "src/main/resources/originalConfig.json";

        List<Class<?>> classList = new ArrayList<>();
        classList.add(DrillReport.class);
        classList.add(OperationRecord.class);
        classList.add(SelectedItem.class);
        classList.add(InspectionItem.class);
        classList.add(WellDeviationData.class);
        classList.add(LayerDescription.class);
        classList.add(GasTestData.class);

        ExcelExtractor extractor = new ExcelExtractor(excelPath, configPath,
classList);
        SheetExtractionResult result = extractor.extractSheetByIndex(9);

        if (result.isSuccess()) {
            // 获取钻井报告基本信息
            DrillReport report = result.getResult(DrillReport.class);

            // 获取各种列表数据
            List<OperationRecord> records =
result.getResultList(OperationRecord.class);
```

```
            List<LayerDescription> layerDescriptions =
result.getResultList(LayerDescription.class);
            List<GasTestData> gasData =
result.getResultList(GasTestData.class);
            List<InspectionItem> inspectionItems =
result.getResultList(InspectionItem.class);
            List<SelectedItem> selectedItems =
result.getResultList(SelectedItem.class);
            List<WellDeviationData> deviationData =
result.getResultList(WellDeviationData.class);

            // 处理提取的数据
            System.out.println("钻井报告基本信息：");
            System.out.println(report);

            System.out.println("\n作业记录：");
            records.forEach(System.out::println);

            // ... 处理其他数据
        }
    }
}
```

8.2 提取电缆地层测试数据

```
public class CableLayerSampleDataExample {
    public static void main(String[] args) {
        String excelPath = "path/to/cable_layer_test.xlsx";
        String configPath = "src/main/resources/verticalConfig.json";

        List<Class<?>> classList = new ArrayList<>();
        classList.add(CableLayerSampleData.class);
        classList.add(SampleVolume.class);

        ExcelExtractor extractor = new ExcelExtractor(excelPath, configPath,
classList);
        SheetExtractionResult result = extractor.extractSheetByIndex(1);

        if (result.isSuccess()) {
            // 获取样品数据
            List<CableLayerSampleData> sampleData =
result.getResultList(CableLayerSampleData.class);
            List<SampleVolume> volumeData =
result.getResultList(SampleVolume.class);

            System.out.println("样品数据：");
            sampleData.forEach(System.out::println);

            System.out.println("\n体积数据：");
            volumeData.forEach(System.out::println);
        }
    }
}
```

9. 常见问题

9.1 如何处理合并单元格?

对于合并单元格，有两种处理方式：

1. 使用`isMergeType: true`配置，自动从上方单元格获取数据
2. 使用`extractPattern`配置，从合并单元格文本中提取所需数据

9.2 如何动态识别数据范围?

通过配置`startFlag`和`endFlag`，可以动态识别数据范围：

```
{
  "isDynamic": true,
  "isDynamicRows": true,
  "startFlag": {
    "text": "作业概况",
    "columnCell": "B"
  },
  "endFlag": {
    "text": "作业计划",
    "columnCell": "B"
  }
}
```

9.3 如何处理垂直表格数据?

对于垂直表格，使用`VERTICAL_LIST`类型，并使用`rowCell`指定行号：

```
{
  "resultType": "VERTICAL_LIST",
  "startColumn": "D",
  "table": {
    "columns": {
      "fieldName": {
        "javaFieldName": "fieldName",
        "javaFieldType": "String",
        "rowCell": "3"
      }
    }
  }
}
```

10. 常用函数列表

10.1 ExcelExtractor 构造函数

```
// 方式1：使用配置文件路径创建
public ExcelExtractor(String excelPath, String configPath, List<Class<?>>
classList)

// 方式2：使用配置对象创建
```

```
public ExcelExtractor(String excelPath, ExcelMappingConfig config,
List<Class<?>> classList)
```

参数说明：

- `excelPath`：Excel文件路径
- `configPath`：配置文件路径
- `config`：ExcelMappingConfig配置对象
- `classList`：需要提取的类列表

10.2 数据提取方法

```
// 提取所有工作表数据  适用于一个excel文件里面的所有sheet结构相同的情况，否则只有部分会成功
public ExtractionResult extractAllSheet()

// 提取指定索引的工作表数据
public SheetExtractionResult extractSheetByIndex(int sheetIndex)

// 提取指定名称的工作表数据
public SheetExtractionResult extractSheetByName(String sheetName)
```

10.3 结果获取方法

```
// 获取单个对象结果
public <T> T getResult(Class<T> clazz)

// 获取对象列表结果
public <T> List<T> getResultList(Class<T> clazz)

// 获取提取是否成功
public boolean isSuccess()
```

10.4 使用示例

```
// 1. 基本使用
ExcelExtractor extractor = new ExcelExtractor(excelPath, configPath,
classList);
SheetExtractionResult result = extractor.extractSheetByIndex(0);
if (result.isSuccess()) {
    List<WellTestData> dataList = result.getResultList(WellTestData.class);
}

// 2. 提取多个工作表
ExtractionResult allResults = extractor.extractAllSheet();
for (SheetExtractionResult sheetResult : allResults.getSheetResults()) {
    if (sheetResult.isSuccess()) {
        // 处理每个工作表的数据
    }
}
```

10.5 配置对象操作

```java
// 创建配置对象
ExcelMappingConfig config = new ExcelMappingConfig(configPath);

// 获取所有提取器配置
List<ExtractorConfig> extractors = config.getAllExtractors();

// 获取指定ID的提取器配置
ExtractorConfig extractor = config.getExtractor("extractorId");

// 获取配置描述
String description = extractor.getDescription();

// 获取目标类
String targetClass = extractor.getTargetClass();

// 获取结果类型
String resultType = extractor.getResultType();
```

10.6 注意事项

1. 数据提取顺序：
    - 按照配置文件中`order`字段指定的顺序执行
    - 数字越小越先执行
    - 建议按照数据依赖关系设置顺序
2. 错误处理：
    - 始终检查`isSuccess()`返回值