# EasyExcel 自定义单元格合并导出

## 一、概述

### 1.1 前言

遇到一个关于导出 Excel 问题，需要根据数据对一些单元格进行合并。

使用的是阿里的 EasyExcel 框架，已经完成数据的正常导出（无合并单元格导出），但是在合并单元格时，遇到了一些问题，之前也没有做相关的操作，所以没有相关积累。

首先，先是想到去官网找相关的资料，看是否可以解决问题，可惜官网只有比较简单的描述，并无法解决问题。最后，通过从网上找一些资料参考完成了对于合并单元格导出。

做完后，感觉 EasyExcel 使用起来还是比较灵活方便的，通过查找上网资料也可以了解到 EasyExcel 还是比较不错的框架，也是基于其他框架进行的封装，下面就记录一下合并单元格的相关代码，并简单介绍下 EasyExcel 相关信息。

> 对于使用 EasyExcel 导出正常 Excel 文件不着重介绍，这里着重介绍一下如何将单元格进行合并。

### 1.2 EasyExcel

将数据导出为 Excel 文件，使用的是阿里的 EasyExcel 框架，该框架是基于 Apache POI 框架，其重写 POI 对07版 Excel 解析，导出性能更高，且不会出现内存溢出的情况；还对03版依赖 POI 的 SAX 模式，在上层做了模型转换的封装，让使用者更加简单方便。

## 1.3 官方网站

官方网站：[easyexcel.opensource.alibaba.com/](http://cxyroad.com/”https://easyexcel.opensource.alibaba.com/”)

Github 地址：[github.com/alibaba/eas...](http://cxyroad.com/”https://github.com/alibaba/easyexcel”)

Gitee 地址：[gitee.com/easyexcel/e...](http://cxyroad.com/”https://gitee.com/easyexcel/easyexcel”)

## 1.4 优点

* 快速：快速的读取 Excel 中的数据。
* 简洁：映射 Excel 和实体类，让代码变的更加简洁。
* 大文件：在读写大文件的时候使用磁盘做缓存，更加的节约内存。

## 1.5 EasyExcel 性能

16M内存23秒读取75M(46W行25列)的Excel（3.2.1+版本）

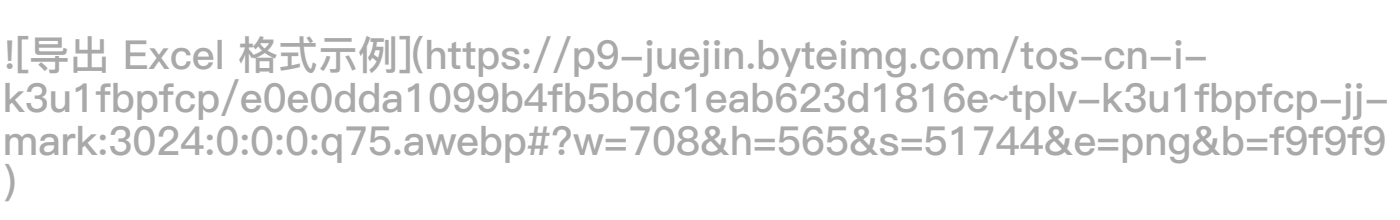
当然还有[极速模式](http://cxyroad.com/”https://easyexcel.opensource.alibaba.com/qa/read#%E5%BC%80%E5%90%AF%E6%80%A5%E9%80%9F%E6%A8%A1%E5%BC%8F”)能更快，但是内存占用会在100M多一点

![EasyExcel 性能](https://p1-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/941d99c3ecf3468a8edab26a3e04769e~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1434&h=310&s=61079&e=png&b=3c4042)

# 二、需求

## 2.1 需求描述

将数据库查询出的数据集合，导出为 Excel 文件，导出的 Excel 文件格式如下。




1. 进行跨行合并的列只有第A列（地市），第B列（区县），第F列（地市总人数），其他列不进行跨行合并；
2. 对于第F列（地市总人数）跨行合并的跨行数，与第A列跨行合并的跨行数保持一致；
3. 第F列（地市总人数）数值为该地市（第A列）的第E列（人数）数值总和；
4. 每个地市（第A列）的首行，为该地市的状态，需要跨列合并第B列（区县）和第C列（学校名称）。

## 2.2 实现步骤

1. 使用 EasyExcel 导出一个不进行单元格合并的 Excel 文件（数据源不需要查询数据库，手动初始化一个集合即可）；
2. 需要注意，上述导出文件需要将合并单元格赋值为相同数据，例如：

* A2 到 A11，10个单元格都需要赋值为“济南市”；
* B2 到 C2，2个单元格都需要赋值为“济南市”；
* B3 到 B4，2个单元格都需要赋值为“市直属”；

3. 需要实现、注册 CellWriteHandler，来完成单元格合并，具体操作步骤下面代码给出；
4. 添加单元格样式，所有单元格居中展示数据。

# 三、需求实现

## 3.1 数据准备

### 3.1.1 数据实体类

由于 EasyExcel 将 Excel 与实体类进行了映射，所以我们导出时，需要提前创建一个对应实体类。

```
package com.example.easy.excel.sample.model;

import com.alibaba.excel.annotation.ExcelIgnore;
import com.alibaba.excel.annotation.ExcelProperty;
import com.alibaba.excel.annotation.write.style.ColumnWidth;
import lombok.Data;

/**
 * @Date: 2024-06-08 17:43
 * @Description TODO 导出数据类
 */
@Data
public class CityExportDTO {

    @ExcelIgnore
    private String cityCode;
    @ExcelProperty("地市")
    @ColumnWidth(20)
    private String cityName;
    @ExcelIgnore
    private String countyCode;
    @ExcelProperty("区县")
    @ColumnWidth(20)
    private String countyName;
    @ExcelProperty("学校名称")
    @ColumnWidth(30)
    private String schoolName;
    @ExcelProperty("状态")
    @ColumnWidth(20)
    private String status;
    @ExcelProperty("人数")
    @ColumnWidth(20)
    private String num;
    @ExcelProperty("地市总人数")
    @ColumnWidth(30)
    private String totalNum;

    public CityExportDTO(String cityCode, String cityName, String
countyCode, String countyName, String schoolName, String status,
```

```
String num, String totalNum) {
        this.cityCode = cityCode;
        this.cityName = cityName;
        this.countyCode = countyCode;
        this.countyName = countyName;
        this.schoolName = schoolName;
        this.status = status;
        this.num = num;
        this.totalNum = totalNum;
    }
}
```

注解说明：

* @ExcelIgnore()：表示当前字段不输出到 Excel 文件中；
* @ExcelProperty()：表示输出到 Excel 文件中表头的名称；
* @ColumnWidth()：表示输出到 Excel 文件中列的列宽；

其他注解说明：

* @DatetimeFormat()：日期转换，参考 java.text.SimpleDateFormat 书写；
* @NumberFormat()：数字转换，参考 java.text.DecimalFormat 书写；
* @ExcelIgnoreUnannotated：表示导出时，字段上没有 `@ExcelProperty()` 注解的字段忽略，该注解在类上标识与 `@ExcelIgnore()` 作用基本一致；

### 3.1.2 初始化数据集合

该文章主要介绍记录导出 Excel 表格的合并单元格，所以导出的数据源手动初始化一个 List 集合即可，不需要从数据库获取。

调用下面方法，可以获取 [2.1 需求描述](http://cxyroad.com/ "2.1-需求描述") 图片示例相同数据。

```
public List<CityExportDTO> initDataList() {
    List<CityExportDTO> list = Arrays.asList(
            new CityExportDTO("370100", "济南市", "370100", "济南市", "济南市", "启用", "0", "184"),
```

```
            new CityExportDTO("370100", "济南市", "370101", "市直属",
"学校1", "启用", "12", "184"),
            new CityExportDTO("370100", "济南市", "370101", "市直属",
"学校2", "启用", "22", "184"),
            new CityExportDTO("370100", "济南市", "370102", "历下区",
"学校3", "启用", "15", "184"),
            new CityExportDTO("370100", "济南市", "370102", "历下区",
"学校4", "未启用", "26", "184"),
            new CityExportDTO("370100", "济南市", "370102", "历下区",
"学校5", "未接入", "31", "184"),
            new CityExportDTO("370100", "济南市", "370103", "市中区",
"学校6", "未知", "18", "184"),
            new CityExportDTO("370100", "济南市", "370104", "槐荫区",
"学校7", "启用", "24", "184"),
            new CityExportDTO("370100", "济南市", "370104", "槐荫区",
"学校8", "启用", "15", "184"),
            new CityExportDTO("370100", "济南市", "370104", "槐荫区",
"学校9", "未确认", "11", "184"),
            new CityExportDTO("370200", "青岛市", "370200", "青岛市",
"青岛市", "启用", "0", "268"),
            new CityExportDTO("370200", "青岛市", "370201", "市直属",
"学校10", "未接入", "22", "268"),
            new CityExportDTO("370200", "青岛市", "370201", "市直属",
"学校11", "启用", "35", "268"),
            new CityExportDTO("370200", "青岛市", "370201", "市南区",
"学校12", "启用", "32", "268"),
            new CityExportDTO("370200", "青岛市", "370201", "市南区",
"学校13", "未接入", "25", "268"),
            new CityExportDTO("370200", "青岛市", "370201", "市北区",
"学校14", "未接入", "27", "268"),
            new CityExportDTO("370200", "青岛市", "370201", "市北区",
"学校15", "启用", "16", "268"),
            new CityExportDTO("370200", "青岛市", "370201", "市北区",
"学校16", "未确认", "12", "268"),
            new CityExportDTO("370200", "青岛市", "370201", "黄岛区",
"学校17", "未知", "10", "268"),
            new CityExportDTO("370200", "青岛市", "370201", "黄岛区",
"学校18", "启用", "24", "268"),
            new CityExportDTO("370200", "青岛市", "370201", "黄岛区",
"学校19", "未启用", "33", "268"),
            new CityExportDTO("370200", "青岛市", "370201", "黄岛区",
"学校20", "未接入", "32", "268")
    );
    return list;
}

```

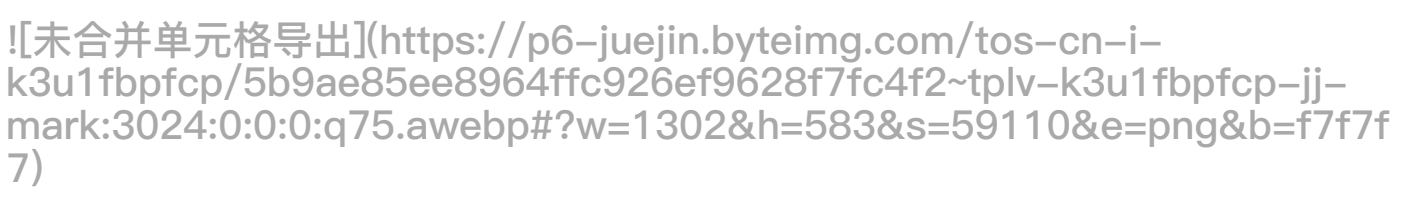
## 3.2 Excel 导出

实现正常未进行单元格合并的 Excel 表格导出。

```
package com.example.easy.excel.sample.controller;

import com.alibaba.excel.EasyExcel;
import com.example.easy.excel.sample.model.CityExportDTO;
import jakarta.servlet.http.HttpServletResponse;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;

/**
 * @Date: 2024-06-08 17:17
 * @Description TODO EasyExcel
 */
@RestController
@RequestMapping("/easyexcel")
public class EasyExcelController {

    @GetMapping("/export")
    public void export(HttpServletResponse response) throws IOException
{
        //获取数据集合
        List<CityExportDTO> cityExportList = initDataList();
        //定义文件名
        String fileName = new String("导出文件
".getBytes(StandardCharsets.UTF_8), StandardCharsets.ISO_8859_1);
        //设置响应参数
        response.setContentType("application/vnd.ms-excel;charset=utf-
8");
        response.setHeader("Content-Disposition",
"attachment;filename=" + fileName + ".xlsx");
        //使用EasyExcel导出文件
        EasyExcel.write(response.getOutputStream(), CityExportDTO.class)
                .sheet(1, "导出文件")
```

```
                .doWrite(cityExportList);
    }

}
```

导出效果如下图片所示：




### 3.3 实现单元格跨行合并

处理合并时，需要注意几个点：

* 正常情况下，是不知道需要跨多少行进行合并的，合并多少行由数据决定，所以跨行合并，行数不固定；
* 只有 【A, B, F】三列需要进行合并。像是 【D，E】 两列上下两个单元格值相同也不需要合并；
* 第F列（地市总人数）是需要与第A列保持一致。也就是说，假如两个地市（济南和青岛）的第F列数值相同（例如值都是184），也不能进行合并；

实现代码如下：

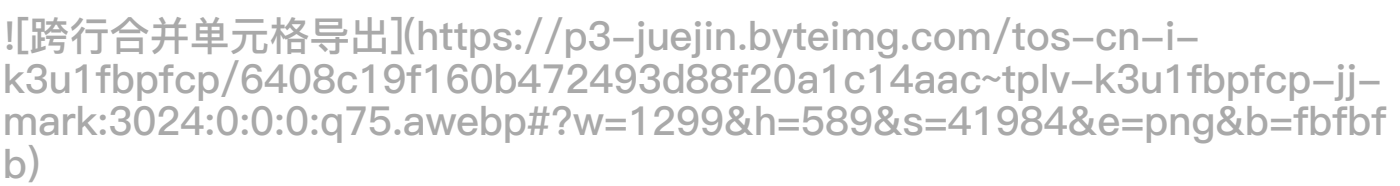
```
//使用EasyExcel导出文件
EasyExcel.write(response.getOutputStream(), CityExportDTO.class)
    .sheet(1, ”导出文件”)
    //注册 CellWriterHandler，实现单元格合并
    .registerWriteHandler(new CellWriteHandler() {

        /**
         * 在 Cell 写入后处理
         *
         * @param writeSheetHolder
         * @param writeTableHolder
         * @param cellDataList
```

```
         * @param cell                  当前 Cell
         * @param head
         * @param relativeRowIndex   表格内容行索引，从除表头的第一行开始，索引为0
         * @param isHead               是否是表头，true表头，false非表头
         */
        @Override
        public void afterCellDispose(WriteSheetHolder writeSheetHolder, WriteTableHolder writeTableHolder, List<WriteCellData<?>> cellDataList, Cell cell, Head head, Integer relativeRowIndex, Boolean isHead) {
            //判断当前为表头，不执行操作
            if (isHead) {
                log.info("\r\n当前为表头, 不执行操作");
                return;
            }

            /*
             * 判断当前 Cell 所在列索引不是0，1，5，不执行操作
             * 说明：
             *   1、列索引从0开始
             *   2、不添加当前代码判断，整个表都会进行跨行合并
             */
            if (cell.getColumnIndex() != 0 && cell.getColumnIndex() != 1 && cell.getColumnIndex() != 5) {
                log.info("\r\n当前不是 [0, 1, 5] 其中一列, 不执行操作");
                return;
            }

            //当前 Sheet
            Sheet sheet = cell.getSheet();
            //当前 Cell 所在行索引
            int rowIndexCurr = cell.getRowIndex();
            //当前 Cell 所在行的上一行索引
            int rowIndexPrev = rowIndexCurr - 1;
            //当前 Cell 所在行的 Row 对象
            Row rowCurr = cell.getRow();
            //当前 Cell 所在行的上一行 Row 对象
            Row rowPrev = sheet.getRow(rowIndexPrev);
            //当前单元格的上一行同列单元格
            Cell cellPrev = rowPrev.getCell(cell.getColumnIndex());

            //当前单元格的值
            Object cellValueCurr = cell.getCellType() == CellType.STRING ? cell.getStringCellValue() : cell.getNumericCellValue();
            //上面单元格的值
            Object cellValuePrev = cellPrev.getCellType() == CellType.STRING ? cellPrev.getStringCellValue() : cellPrev.getNumericCellValue();
```

```
            log.info("\r\n当前单元格值: " + cellValueCurr + ", 上面单元格的值:
" + cellValuePrev + ", 两个单元格值不相等, 不执行操作");

            //判断当前单元格与上面单元格是否相等，不相等不执行操作
            if (!cellValueCurr.equals(cellValuePrev)) {
                log.info("\r\n两个单元格值不相等, 不执行操作");
                return;
            }

            log.info("\r\n两个单元格值相等, 开始进行合并操作");

            /*
             *需要注意，合并时，以第一列（地市）为准
             * 当第一列上下两个单元格不一样时，说明不是一个地市数据，其他
列单元格值相等也不能进行合并
             * 由于可以通过数据知道第一列为字符型数据，可以直接获取
             */
            if
(!rowPrev.getCell(0).getStringCellValue().equals(rowCurr.getCell(0).getStri
ngCellValue())) {
                log.info("\r\n开始 " + rowCurr.getCell(0).getStringCellValue()
+ "地市数据合并操作");
                return;
            }

            //从 Sheet 中，获取所有合并区域
            List<CellRangeAddress> mergedRegions =
sheet.getMergedRegions();
            //是否合并过
            boolean merged = false;
            //遍历合并区域集合
            for (int i = 0; i < mergedRegions.size(); i++) {
                CellRangeAddress cellAddresses = mergedRegions.get(i);
                //判断 cellAddress 的范围是否是从 rowIndexPrev 到
cell.getColumnIndex()
                if (cellAddresses.isInRange(rowIndexPrev,
cell.getColumnIndex())) {
                    //从集合中移除
                    sheet.removeMergedRegion(i);
                    //设置范围最后一行，为当前行
                    cellAddresses.setLastRow(rowIndexCurr);
                    //重新添加到 Sheet 中
                    sheet.addMergedRegion(cellAddresses);
                    //已完成合并
                    merged = true;
                    break;
                }
            }
```

```
            //merged=false，表示当前单元格为第一次合并
            if (!merged) {
                CellRangeAddress cellAddresses = new
CellRangeAddress(rowIndexPrev, rowIndexCurr, cell.getColumnIndex(),
cell.getColumnIndex());
                sheet.addMergedRegion(cellAddresses);
            }
            log.info("\r\n合并单元格完成");
        }
    })
    .doWrite(cityExportList);
```

导出效果如下图片所示：




### 3.4 实现单元格跨列合并

我们分析数据可以看到，地市的第一行进行跨列合并是有规律可循的，只需要在第C列单元格进行判断，是否与左侧单元格（同行的B列单元格）值相同即可，相同则进行合并操作。

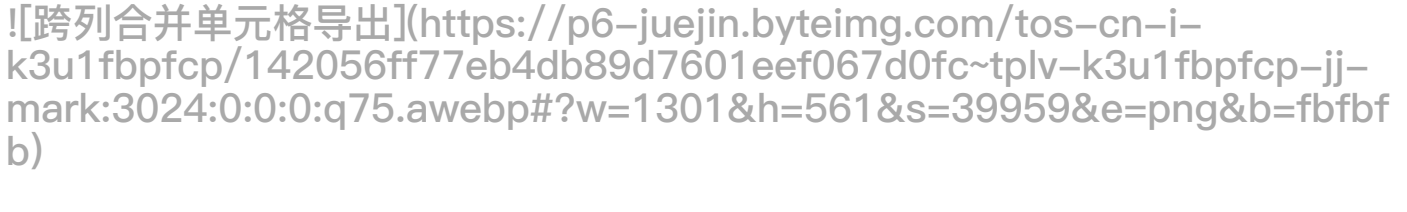
实现代码如下：

```
.registerWriteHandler(new CellWriteHandler() {

    /**
     * 在 Cell 写入后处理
     *
     * @param writeSheetHolder
     * @param writeTableHolder
     * @param cellDataList
     * @param cell 当前 Cell
     * @param head
```

```
     * @param relativeRowIndex 表格内容行索引，从除表头的第一行开始，索引为0
     * @param isHead 是否是表头，true表头，false非表头
     */
    @Override
    public void afterCellDispose(WriteSheetHolder writeSheetHolder, WriteTableHolder writeTableHolder, List<WriteCellData<?>> cellDataList, Cell cell, Head head, Integer relativeRowIndex, Boolean isHead) {
        //判断当前为表头，不执行操作
        if (isHead) {
            log.info("\r\n当前为表头, 不执行操作");
            return;
        }

        /*
         * 判断当前 Cell 所在列索引不是0，1，5，不执行操作
         * 说明：
         * 1、列索引从0开始
         * 2、不添加当前代码判断，整个表都会进行跨行合并
         */
        if (cell.getColumnIndex() != 0 && cell.getColumnIndex() != 1 && cell.getColumnIndex() != 2 && cell.getColumnIndex() != 5) {
            log.info("\r\n当前不是 [0, 1, 5] 其中一列, 不执行操作");
            return;
        }

        //当前 Sheet
        Sheet sheet = cell.getSheet();
        //当前 Cell 所在行索引
        int rowIndexCurr = cell.getRowIndex();
        //当前 Cell 所在行的上一行索引
        int rowIndexPrev = rowIndexCurr - 1;
        //当前 Cell 所在行的 Row 对象
        Row rowCurr = cell.getRow();
        //当前 Cell 所在行的上一行 Row 对象
        Row rowPrev = sheet.getRow(rowIndexPrev);
        //当前单元格的上一行同列单元格
        Cell cellPrev = rowPrev.getCell(cell.getColumnIndex());

        if (cell.getColumnIndex() == 0 || cell.getColumnIndex() == 1 || cell.getColumnIndex() == 5) {
            log.info("\r\n当前处理第 [0, 1, 5] 列单元格跨行合并");
            //原有代码
        } else if (cell.getColumnIndex() == 2) {
            log.info("\r\n当前处理第 [2] 列单元格跨列合并");

            //当前行的第二列，索引为1
            Cell cell1 = rowCurr.getCell(1);
```

```
            //当前行的第三列，索引为2
            Cell cell2 = rowCurr.getCell(2);

            //判断只要两个单元格值相等，就可以直接进行合并
            if (cell1 != null && cell2 != null &&
cell1.getStringCellValue().equals(cell2.getStringCellValue())) {
                //创建一个当前行到当前行，索引为1的列到索引为2的列的合并
，并添加到 Sheet 中
                sheet.addMergedRegion(new
CellRangeAddress(rowIndexCurr, rowIndexCurr, cell1.getColumnIndex(),
cell2.getColumnIndex()));
            }
        } else {
            return;
        }

        log.info("\r\n合并单元格完成");
    }
})

```

导出效果如下图片所示：

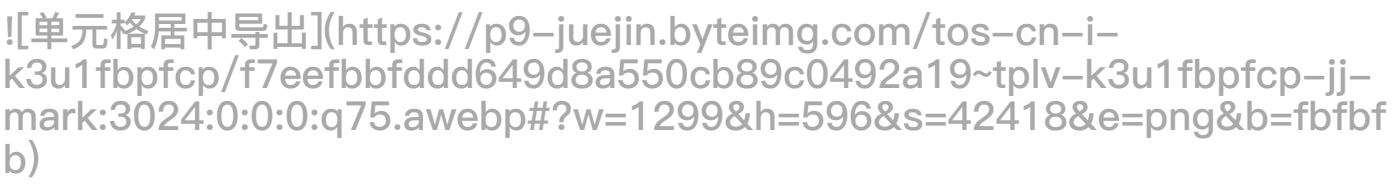



### 3.5 实现单元格居中样式

```

```
//创建 Cell 样式
WriteCellStyle writeCellStyle = new WriteCellStyle();
//设置垂直对齐方式
writeCellStyle.setVerticalAlignment(VerticalAlignment.CENTER);
//设置水平对齐方式
writeCellStyle.setHorizontalAlignment(HorizontalAlignment.CENTER);
//使用EasyExcel导出文件
EasyExcel.write(response.getOutputStream(), CityExportDTO.class)
        .sheet(1, "导出文件")
        //注册 CellWriterHandler，实现单元格合并
        .registerWriteHandler(new CellWriteHandler() {
```
```

```
            //省略
        })
        //注册样式策略
        .registerWriteHandler(new HorizontalCellStyleStrategy(null,
writeCellStyle))
        .doWrite(cityExportList);
```

导出效果如下图片所示：




至此，就实现了导出 Excel 的跨行和跨列合并，并且，还添加了单元格的样式。

四、完整代码
======

```
package com.example.easy.excel.sample.controller;

import com.alibaba.excel.EasyExcel;
import com.alibaba.excel.metadata.Head;
import com.alibaba.excel.metadata.data.WriteCellData;
import com.alibaba.excel.write.handler.CellWriteHandler;
import com.alibaba.excel.write.metadata.holder.WriteSheetHolder;
import com.alibaba.excel.write.metadata.holder.WriteTableHolder;
import com.alibaba.excel.write.metadata.style.WriteCellStyle;
import com.alibaba.excel.write.style.HorizontalCellStyleStrategy;
import com.example.easy.excel.sample.model.CityExportDTO;
import jakarta.servlet.http.HttpServletResponse;
import lombok.extern.slf4j.Slf4j;
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.ss.util.CellRangeAddress;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.io.IOException;
```

```
import java.io.Serializable;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;

/**
 * @Date: 2024-06-08 17:17
 * @Description TODO EasyExcel 读写测试
 */
@Slf4j
@RestController
@RequestMapping("/easyexcel")
public class EasyExcelController {

    @GetMapping("/export")
    public void export(HttpServletResponse response) throws IOException
{
        //获取数据集合
        List<CityExportDTO> cityExportList = initDataList();
        //定义文件名
        String fileName = new String("导出文件
".getBytes(StandardCharsets.UTF_8), StandardCharsets.ISO_8859_1);
        //设置响应参数
        response.setContentType("application/vnd.ms-excel;charset=utf-
8");
        response.setHeader("Content-Disposition",
"attachment;filename=" + fileName + ".xlsx");
        //设置 Cell 样式
        WriteCellStyle writeCellStyle = new WriteCellStyle();
        //设置垂直对齐方式
        writeCellStyle.setVerticalAlignment(VerticalAlignment.CENTER);
        //设置水平对齐方式
writeCellStyle.setHorizontalAlignment(HorizontalAlignment.CENTER);
  //使用EasyExcel导出文件
        EasyExcel.write(response.getOutputStream(), CityExportDTO.class)
                .sheet(1, "导出文件")
                //注册 CellWriterHandler，实现单元格合并
                .registerWriteHandler(new CellWriteHandler() {

                    /**
                     * 在 Cell 定入后处理
                     *
                     * @param writeSheetHolder
                     * @param writeTableHolder
                     * @param cellDataList
                     * @param cell 当前 Cell
                     * @param head
                     * @param relativeRowIndex 表格内容行索引，从除表头的第
```

```
一行开始，索引为0
                     * @param isHead 是否是表头，true表头，false非表头
                     */
                    @Override
                    public void afterCellDispose(WriteSheetHolder
writeSheetHolder, WriteTableHolder writeTableHolder,
List<WriteCellData<?>> cellDataList, Cell cell, Head head, Integer
relativeRowIndex, Boolean isHead) {
                        //判断当前为表头，不执行操作
                        if (isHead) {
                            log.info("\r\n当前为表头, 不执行操作");
                            return;
                        }

                        /*
                         * 判断当前 Cell 所在列索引不是0，1，5，不执行操作
                         *  说明：不添加当前代码判断，整个表都会进行跨行合并
                         *
                         * 添加列索引为2的，处理跨列合并
                         */
                        if (cell.getColumnIndex() != 0 && cell.getColumnIndex()
!= 1 && cell.getColumnIndex() != 2 && cell.getColumnIndex() != 5) {
                            log.info("\r\n当前不是 [0, 1, 5] 其中一列, 不执行操作");
                            return;
                        }

                        //当前 Sheet
                        Sheet sheet = cell.getSheet();
                        //当前 Cell 所在行索引
                        int rowIndexCurr = cell.getRowIndex();
                        //当前 Cell 所在行的上一行索引
                        int rowIndexPrev = rowIndexCurr - 1;
                        //当前 Cell 所在行的 Row 对象
                        Row rowCurr = cell.getRow();
                        //当前 Cell 所在行的上一行 Row 对象
                        Row rowPrev = sheet.getRow(rowIndexPrev);
                        //当前单元格的上一行同列单元格
                        Cell cellPrev = rowPrev.getCell(cell.getColumnIndex());

                        if (cell.getColumnIndex() == 0 || cell.getColumnIndex() ==
1 || cell.getColumnIndex() == 5) {
                            log.info("\r\n当前处理第 [0, 1, 5] 列单元格跨行合并");

                            //当前单元格的值
                            Object cellValueCurr = cell.getCellType() ==
CellType.STRING ? cell.getStringCellValue() : cell.getNumericCellValue();
                            //上面单元格的值
                            Object cellValuePrev = cellPrev.getCellType() ==
```

```
CellType.STRING ? cellPrev.getStringCellValue() :
cellPrev.getNumericCellValue();
                        log.info("\r\n当前单元格值: " + cellValueCurr + ", 上面
单元格的值: " + cellValuePrev + ", 两个单元格值不相等, 不执行操作");

                        //判断当前单元格与上面单元格是否相等，不相等不执行
操作
                        if (!cellValueCurr.equals(cellValuePrev)) {
                            log.info("\r\n两个单元格值不相等, 不执行操作");
                            return;
                        }

                        log.info("\r\n两个单元格值相等, 开始进行合并操作");

                        /*
                         *需要注意，合并时，以第一列（地市）为准
                         * 当第一列上下两个单元格不一样时，说明不是一个地市
数据，其他列单元格值相等也不能进行合并
                         * 由于可以通过数据知道第一列为字符型数据，可以直接
获取
                         */
                        if
(!rowPrev.getCell(0).getStringCellValue().equals(rowCurr.getCell(0).getStri
ngCellValue())) {
                            log.info("\r\n开始 " +
rowCurr.getCell(0).getStringCellValue() + "地市数据合并操作");
                            return;
                        }

                        //从 Sheet 中，获取所有合并区域
                        List<CellRangeAddress> mergedRegions =
sheet.getMergedRegions();
                        //是否合并过
                        boolean merged = false;
                        //遍历合并区域集合
                        for (int i = 0; i < mergedRegions.size(); i++) {
                            CellRangeAddress cellAddresses =
mergedRegions.get(i);
                            //判断 cellAddress 的范围是否是从 rowIndexPrev 到
cell.getColumnIndex()
                            if (cellAddresses.isInRange(rowIndexPrev,
cell.getColumnIndex())) {
                                //从集合中移除
                                sheet.removeMergedRegion(i);
                                //设置范围最后一行，为当前行
                                cellAddresses.setLastRow(rowIndexCurr);
                                //重新添加到 Sheet 中
                                sheet.addMergedRegion(cellAddresses);
```

```
                            //已完成合并
                            merged = true;
                            break;
                        }
                    }

                    //merged=false，表示当前单元格为第一次合并
                    if (!merged) {
                        CellRangeAddress cellAddresses = new
CellRangeAddress(rowIndexPrev, rowIndexCurr, cell.getColumnIndex(),
cell.getColumnIndex());
                        sheet.addMergedRegion(cellAddresses);
                    }
                } else if (cell.getColumnIndex() == 2) {
                    log.info("\r\n当前处理第 [2] 列单元格跨列合并");

                    //当前行的第二列，索引为1
                    Cell cell1 = rowCurr.getCell(1);
                    //当前行的第三列，索引为2
                    Cell cell2 = rowCurr.getCell(2);

                    //判断只要两个单元格值相等，就可以直接进行合并
                    if (cell1 != null && cell2 != null &&
cell1.getStringCellValue().equals(cell2.getStringCellValue())) {
                        //创建一个当前行到当前行，索引为1的列到索引为2的
列的合并，并添加到 Sheet 中
                        sheet.addMergedRegion(new
CellRangeAddress(rowIndexCurr, rowIndexCurr, cell1.getColumnIndex(),
cell2.getColumnIndex()));
                    }
                } else {
                    return;
                }

                log.info("\r\n合并单元格完成");
            }
        })
        .registerWriteHandler(new HorizontalCellStyleStrategy(null,
writeCellStyle))
        .doWrite(cityExportList);
    }

  //初始化数据集合非重点略，参考上面章节代码...
}

```

原文链接: https://juejin.cn/post/7380579033547112485