链滴

使用 opencsv 读写 csv 文件（结合 mysql）

作者：JellyfishMIX

原文链接：https://ld246.com/article/1604940051652

来源网站：链滴

前言

最近遇到了项目需求，需要从 mysql 中导出数据为 csv 文件，再从 csv 文件中读取数据保存到 mysq 。经过检索分析，决定使用 opencsv 实现需求。

本需求可以分为四部分：

1. mysql 数据转换为 java 对象。
2. java 对象转换为 csv 文件。
3. csv 文件转换为 java 对象。
4. java 对象映射保存到 mysql 中。

其中1, 4两步是我们熟悉的增删改查，不必多说。需要解决的是2, 3两步，下面给出2, 3两步的示例代 。

依赖

```
<!-- https://mvnrepository.com/artifact/com.opencsv/opencsv -->
<dependency>
    <groupId>com.opencsv</groupId>
    <artifactId>opencsv</artifactId>
    <version>4.6</version>
</dependency>

<!--lombok-->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.12</version>
</dependency>
```

代码

实体类 Person（使用了lombok依赖）

三个 lombok 注解必须加，如果未使用 lombok，请在此实体类加 setter & getter，全参构造方法 无参构造方法。

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Person {
    private Integer id;
    private String name;
    private Integer age;
}
```

writeCsv

```
public void writeCsv(List<Person> dataList, String finalPath) {
    try {
        Writer writer = new FileWriter(finalPath);

        // 设置显示的顺序
        String[] columnMapping = {"id", "name", "age"};
        ColumnPositionMappingStrategy<Person> mapper =
            new ColumnPositionMappingStrategy<>();
        mapper.setType(Person.class);
        mapper.setColumnMapping(columnMapping);

        // 写表头
        CSVWriter csvWriter = new CSVWriter(writer, CSVWriter.DEFAULT_SEPARATOR, CSVWrit
r.NO_QUOTE_CHARACTER, '\\', "\n");
        String[] header = { "编号", "姓名", "年龄"};
        csvWriter.writeNext(header);

        StatefulBeanToCsv beanToCsv = new StatefulBeanToCsvBuilder(writer)
            .withMappingStrategy(mapper)
            .withQuotechar(CSVWriter.NO_QUOTE_CHARACTER)
            .withSeparator(CSVWriter.DEFAULT_SEPARATOR)
            .withEscapechar('\\')
            .build();
        beanToCsv.write(dataList);
        csvWriter.close();
        writer.close();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (CsvDataTypeMismatchException e) {
        e.printStackTrace();
    } catch (CsvRequiredFieldEmptyException e) {
        e.printStackTrace();
    }
    System.out.println(finalPath + "数据导出成功");
}
```

readCsv

```
public void readCsv(String finalPath) {
    try {
        Reader reader = new InputStreamReader(new FileInputStream(finalPath), StandardChars
ts.UTF_8);
        CSVReader csvReader = new CSVReaderBuilder(reader).build();

        // 列名的映射
        HeaderColumnNameTranslateMappingStrategy<Person> strategy =
            new HeaderColumnNameTranslateMappingStrategy<>();
        strategy.setType(Person.class);
        Map<String, String> columnMapping = new HashMap<>();
        columnMapping.put("编号", "id");
        columnMapping.put("姓名", "name");
        columnMapping.put("年龄", "age");
        strategy.setColumnMapping(columnMapping);
```

```
        CsvToBean<Person> csvToBean = new CsvToBeanBuilder(csvReader)
            .withSeparator(CSVWriter.DEFAULT_SEPARATOR)
            .withQuoteChar(CSVWriter.NO_QUOTE_CHARACTER)
            .withMappingStrategy(strategy)
            .build();

        List<Person> list = csvToBean.parse();

        for (Person p : list) {
            System.out.println(p.toString());
        }
        csvReader.close();
        reader.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

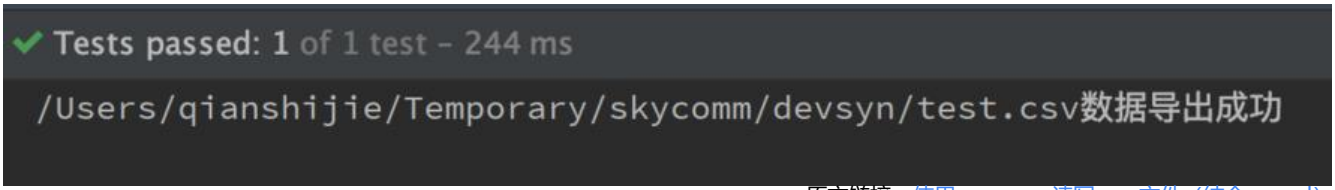
测试类的方法（maven项目结构中的测试类）

finalPath 是绝对路径+文件名，请确保欲生成的文件所在目录已创建好。

```
@Test
void writeCsv() {
    // test data
    Person person0 = new Person(11, "钱多多", 18);
    Person person1 = new Person(22, "王多鱼", 19);
    Person person2 = new Person(33, "喜羊羊", 20);
    List<Person> personList = new ArrayList<>();
    personList.add(person0);
    personList.add(person1);
    personList.add(person2);

    String finalPath = "/Users/qianshijie/Temporary/skycomm/devsyn/test.csv";

    personService.writeCsv(personList, finalPath);
}

@Test
void readCsv() {
    String finalPath = "/Users/qianshijie/Temporary/skycomm/devsyn/test.csv";

    personService.readCsv(finalPath);
}
```

运行结果

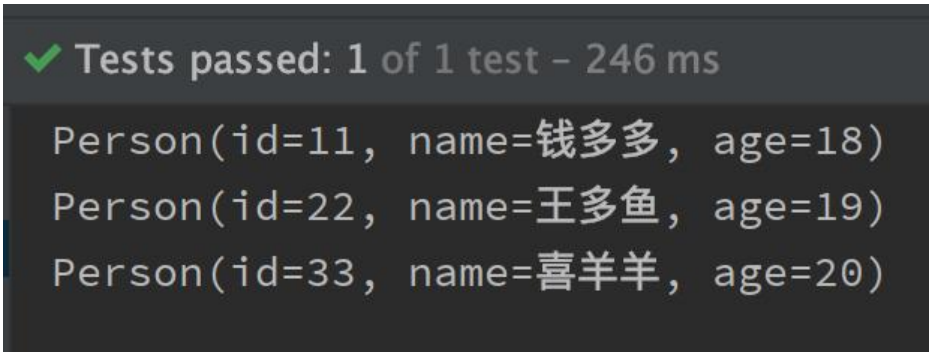
根据 java 对象生成 csv 文件成功：



生成的 csv 文件（可使用 vim 查看，使用 excel 打开 csv 文件会自动转换为表格）：

```
编号,姓名,年龄
11,钱多多,18
22,王多鱼,19
33,喜羊羊,20
~
~
```

读取 csv 文件转换为 java对象 成功：



One More Thing

上述方法中，csv 文件的列名均已被替换（已不是实体类的属性名）。如果想让生成的 csv 文件列名实体类属性名保持一致。请使用如下代码（只列举有变化的代码，未列出则代码无变动，变动已将原码注释，以供对比）：

writeCsv

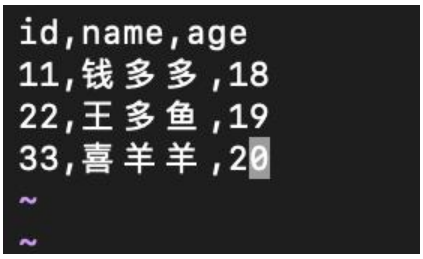
```
public void writeCsv(List<Person> dataList, String finalPath) {
    try {
        Writer writer = new FileWriter(finalPath);

        // 设置显示的顺序
        String[] columnMapping = {"id", "name", "age"};
        ColumnPositionMappingStrategy<Person> mapper =
            new ColumnPositionMappingStrategy<>();
        mapper.setType(Person.class);
        mapper.setColumnMapping(columnMapping);

        // 写表头
        CSVWriter csvWriter = new CSVWriter(writer, CSVWriter.DEFAULT_SEPARATOR, CSVWrit
r.NO_QUOTE_CHARACTER, '\\', "\n");
        // String[] header = { "编号", "姓名", "年龄"};
        String[] header = { "id", "name", "age"};
        csvWriter.writeNext(header);

        StatefulBeanToCsv beanToCsv = new StatefulBeanToCsvBuilder(writer)
            .withMappingStrategy(mapper)
            .withQuotechar(CSVWriter.NO_QUOTE_CHARACTER)
            .withSeparator(CSVWriter.DEFAULT_SEPARATOR)
            .withEscapechar('\\')
```

```
            .build();
        beanToCsv.write(dataList);
        csvWriter.close();
        writer.close();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (CsvDataTypeMismatchException e) {
        e.printStackTrace();
    } catch (CsvRequiredFieldEmptyException e) {
        e.printStackTrace();
    }
    System.out.println(finalPath + "数据导出成功");
}
```

readCsv

```
public void readCsv(String finalPath) {
    try {
        Reader reader = new InputStreamReader(new FileInputStream(finalPath), StandardChars
ts.UTF_8);
        CSVReader csvReader = new CSVReaderBuilder(reader).build();

        // 列名的映射
        // HeaderColumnNameTranslateMappingStrategy<Person> strategy =
        //         new HeaderColumnNameTranslateMappingStrategy<>();
        // strategy.setType(Person.class);
        // Map<String, String> columnMapping = new HashMap<>();
        // columnMapping.put("编号", "id");
        // columnMapping.put("姓名", "name");
        // columnMapping.put("年龄", "age");
        // strategy.setColumnMapping(columnMapping);

        CsvToBean<Person> csvToBean = new CsvToBeanBuilder<Person>(csvReader)
            .withSeparator(CSVWriter.DEFAULT_SEPARATOR)
            .withQuoteChar(CSVWriter.NO_QUOTE_CHARACTER)
            // .withMappingStrategy(strategy)
            .withType(Person.class)
            .build();

        List<Person> list = csvToBean.parse();

        for (Person p : list) {
            System.out.println(p.toString());
        }
        csvReader.close();
        reader.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

生成的 csv 文件



csv 文件读写均可正常运行。

引用/参考

openCSV读写CSV文件 - peterwanghao - CSDN