02_EasyExcel自定义ReadListener?

时间：2023年2月12日23:03:53

注意：

- 如果解析的数据为null，请检查*@Accessors(chain=false)*，chain=true会使得easyExcel不能正常工作。
    - 3.1.3

一、示例

- 虽然可以使用模板模式操作细节进行封装（限制空白行数），但是封装后，在面对负载导入逻辑时，只会增加使用复杂性。
    - 直接创建单独的PageReadListener即可。

```java
public class PageReadListener<T> implements ReadListener<T> {

    Defuault single handle the amount of data
    public static int BATCH_COUNT = 100;
    Temporary storage of data
    private List<T> cachedDataList = ListUtils.newArrayListWithExpectedSize(BATCH_COUNT);
    consumer
    private final Consumer<List<T>> consumer;

    Single handle the amount of data
    private final int batchCount;

    public PageReadListener(Consumer<List<T>> consumer) {
        this(consumer, BATCH_COUNT);
    }

    public PageReadListener(Consumer<List<T>> consumer, int batchCount) {
        this.consumer = consumer;
        this.batchCount = batchCount;
    }

    @Override
    public void invoke(T data, AnalysisContext context) {
        cachedDataList.add(data);
        if (cachedDataList.size() >= batchCount) {
            consumer.accept(cachedDataList);
            cachedDataList = ListUtils.newArrayListWithExpectedSize(batchCount);
        }
    }

    @Override
    public void doAfterAllAnalysed(AnalysisContext context) {
        if (CollectionUtils.isNotEmpty(cachedDataList)) {
            consumer.accept(cachedDataList);
        }
    }

}
```

```
    /**
     * 官方示例请参考：https://easyexcel.opensource.alibaba.com/docs/current/quickstart/read#excel%E7%A4%BA%E4%BE%8B
     * – headRowNumber默认为1
     *
     * @throws IOException
     */
    @Test
    public void demo03() throws IOException {
        InputStream inputStream = new ClassPathResource("files/用户清单.xlsx").getInputStream();
        EasyExcel.read(inputStream, UserExcelImportDTO.class, new UserExcelReadListener<UserExcelImportDTO>())
                .sheet()
                .headRowNumber(1)
                .doRead();
    }


    /* UserExcel读取监听器 */
    class UserExcelReadListener<T> implements ReadListener<T> {
        private final int BATCH_COUNT = 100;
        private final List<T> cachedDataList = new ArrayList<>(BATCH_COUNT);
        private int emptyRowCounter = 0;

        /**
         * 如果excel下方有大量空行，invoke默认会一直解析空行
         * – 17:50:34.187 [main] DEBUG com.alibaba.excel.read.processor.DefaultAnalysisEventProcessor – Empty row!
         */
        @Override
        public void invoke(T data, AnalysisContext context) {
            // 取消自动空行检测后对非空行进行操作
            if (!RowTypeEnum.EMPTY.equals(context.readRowHolder().getRowType())) {
                cachedDataList.add(data);
                if (cachedDataList.size() >= BATCH_COUNT) {
                    saveData();
                    cachedDataList.clear();
                }
            }
        }

        @Override
        public void doAfterAllAnalysed(AnalysisContext context) {
            saveData();
        }

        private void saveData() {
            if (cachedDataList.isEmpty()) return;
            log.info("{}条数据，开始存储数据库！", cachedDataList.size());
            log.info("存储数据库成功！");
            System.out.println("\uD83D\uDC49\uD83D\uDC49\uD83D\uDC49" + cachedDataList);
        }

        @Override
        public void invokeHead(Map<Integer, ReadCellData<?>> headMap, AnalysisContext context) {
            // 取消自动空行检测
            context.readWorkbookHolder().setIgnoreEmptyRow(false);
            ReadListener.super.invokeHead(headMap, context);
        }
```

```
    @Override
    public boolean hasNext(AnalysisContext context) {
        if (RowTypeEnum.EMPTY.equals(context.readRowHolder().getRowType())) {
            emptyRowCounter++;
        } else {
            emptyRowCounter = 0;
        }
        final int MAX_ALLOW_EMPTY_ROW = 10;
        if (emptyRowCounter > MAX_ALLOW_EMPTY_ROW) {
            // 停止前触发数据保存
            saveData();
            log.info("空行数目超过最大允许数目: {}, 解析中止", MAX_ALLOW_EMPTY_ROW);
            return false;
        }
        return true;
    }
}
```

二、读取数据，存储时处理重复数据的两种策略（优先采取方案一，首先要实现，然后才是优化）

> DB中需要唯一索引来标记字段的唯一性
>
> - 每批写入的数据，要根据唯一索引进行去重

① 重复时，覆盖原有数据?

- 读取一条，写入一条，先更新，如果更新失败则插入
  - ```
    boolean save = false;
    save = userService.saveOrUpdate(user1, Wrappers.<User>query().lambda().eq(User::getEmail, user1.getEmail()
    ```
  - ```
    JDBC Connection [com.mysql.cj.jdbc.ConnectionImpl@3dbb7bb] will not be managed by Spring
    ==>  Preparing: UPDATE User SET email=? WHERE (email = ?)
    ==> Parameters: 1(String), 1(String)
    <==    Updates: 1
    Closing non transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@d60cbad]
    ================================================================
    ```
    ```
    User user1 = new User( id: "9999999999999999999", createTime: null
    user1.setEmail("1");
    save = userService.saveOrUpdate(user1, Wrappers.<User>query().lambda().eq(User::getEmail, user1.getEmail()
    ```
- 分批写入?
  - 方案一
    - 读取唯一索引标记的所有列；
    - 获取【待更新数据】+【待添加数据】；
    - 开启事务，插入/更新，事务提交。
  - 方案二（必须确保数据库存在唯一索引）
    - 手动编写Mapper ID要手动生成填充，*ON DUPLICATE KEY UPDATE* 后除了ID的字段都要显式赋值
      - INSERT INTO user VALUES("4", "A4", "A4-nick-test") *ON DUPLICATE KEY UPDATE* nick_name=VALUES(nick_name),user_name=VALUES(user_name);

② 重复时，保留原有数据?

- 读取一条，写入一条，捕获重复异常（唯一索引重复），不抛出

```
boolean save = false;
try {
    save = userService.save(user1);
} catch (Exception e) {
    log.error(e.getMessage());
}
```

- 分批写入?
  - 方案一
    - 读取唯一索引标记的所有列；
    - 在【读取数据】中移除已存在记录；
    - 开启事务，写入数据，事务提交。
  - 方案二（必须确保数据库存在唯一索引）
    - 手动编写Mapper ID要手动生成并填充
      - *INSERT IGNORE INTO* user(user_id,email) VALUES(”111”, ”1”),(”222”,”2”),(”333”,”3”),(”111”, ”111”),(”222”,”222”),(”333”,”333”);