01_SpringBoot导入解析Excel?

时间：2023年2月12日20:05:39

测试用Excel：

- abc.xlsx

一、核心内容

1. 将每列的属性和POJO对应
2. 单条解析，存储解析后的数据（分批操作，例如一次写入1000条数据）

1. Excel与POJO

	A	B	C	D
1	姓名	编号	年龄	
2	AAA	0001	12	
3	BBB	0002	14	
4	CCC	0003	15	
5				

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class UserExcelImportDTO {
    0 个用法
    private String name;
    0 个用法
    private String code;
    0 个用法
    private Integer age;
}
```

默认会按照顺序与Excel中属性对应

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class UserExcelImportDTO {

    0 个用法
    @ExcelProperty(index = 0)
    private String name;
    0 个用法
    @ExcelProperty(index = 1)
    private String code;

    0 个用法
    @ExcelProperty(index = 2)
    private Integer age;

    0 个用法
    @ExcelProperty(index = 3)
    private LocalDateTime createTime;

    0 个用法
    @ExcelIgnore
    private String description;
}
```

约定：导入时，使用index将Excel列映射到POJO属性上

2. 接口

```
@Slf4j
@RestController
@RequestMapping(”/excels”)
public class ExcelController {

    /**
     * 解析Excel
     * 1. Tips：检测相关的工作请在前端完成
     * 2. 默认值
     *    2.1 PageReadListener.BATCH_COUNT=100                    -> 默认每组100个数据
     *    2.2 EasyExcel.read(a,b,c).sheet().headRowNumber(1).doRead()  -> 默认读取第一个sheet，header为第一行
     *
     * @param file MultipartFile
     * @return ”ok”/”error”
     */
    @PostMapping(”/upload”)
    public String excelUpload(@RequestPart(”file”) MultipartFile file) {
        try (InputStream inputStream = file.getInputStream()) {
            // 默认BATCH_COUNT=100
            PageReadListener.BATCH_COUNT = 5;
            EasyExcel.read(inputStream, UserExcelImportDTO.class, new PageReadListener<UserExcelImportDTO>
(dataList -> {
```

```
                for (UserExcelImportDTO demoData : dataList) {
                    log.info("读取到一条数据{}", JSONObject.toJSONString(demoData));
                }
            })).sheet().headRowNumber(1).doRead();
        } catch (IOException e) {
            log.error(e.getMessage());
            return "error";
        }

        return "ok";
    }
}
```

二、指定Excel列和POJO中属性对应

- index关联
- ~~名称关联~~

1. 位置关联（index） ***读取的时候使用位置关联即可***

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class UserExcelImportDTO {

    @ExcelProperty(index = 1)
    private String code;

    @ExcelProperty(index = 0)
    private String name;

    @ExcelProperty(index = 2)
    private Integer age;

    @ExcelIgnore
    private String description;
}
```

2. 时间怎么处理

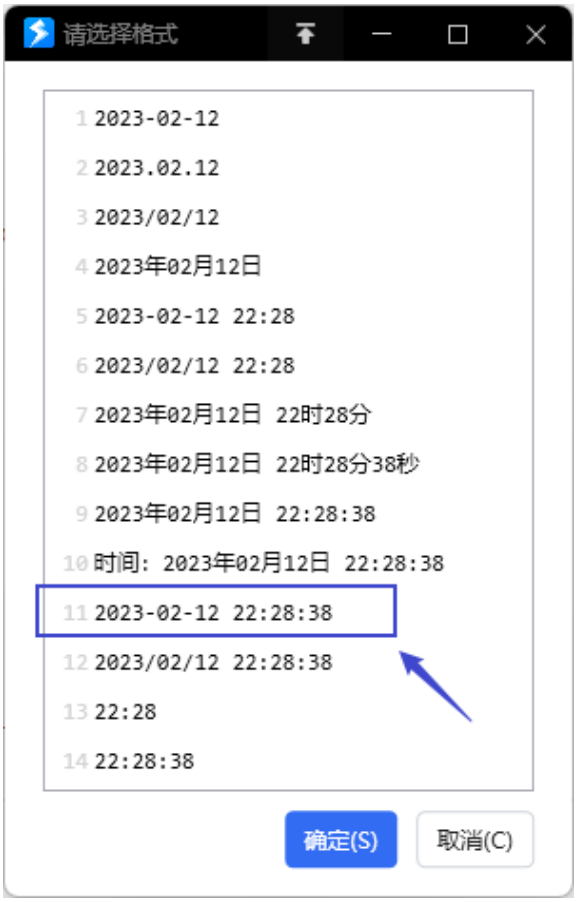
- ***导入时，请在Excel中创建时间类型数据***（*不要使用文本类型，如果使用文本类型，请符合标准格式*）

	A	B	C	D	E	F
1	姓名	编号	年龄	创建时间		
2	AAA	0001	12	2023-02-12 22:54:15		
3	BBB	0002	14	2023/2/13 22:24		
4	CCC	0003	15	2023/2/14 22:24		
5	DDD	0004	22	2023/2/12 22:24		
6	EEE	0005	24	2023/2/13 22:24		
7	FFF	0006	25	2023/2/14 22:24		
8						

文本类型

excel标准时间

- 标准格式即可：yyyy-MM-dd HH:mm:ss



- @DateTimeFormat("yyyy-MM-dd HH:mm:ss")，在导入时，会用于解析文本类型的时间。

```
// com.alibaba.excel.annotation.format.DateTimeFormat;
0 个用法
@ExcelProperty(value = "创建时间", index = 3)
@DateTimeFormat("yyyy-MM-dd HH:mm:ss")
private LocalDateTime createTime;
```

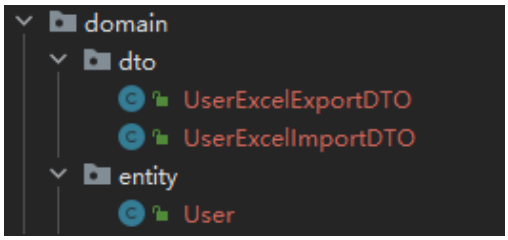
3. boolean类型支持什么数据

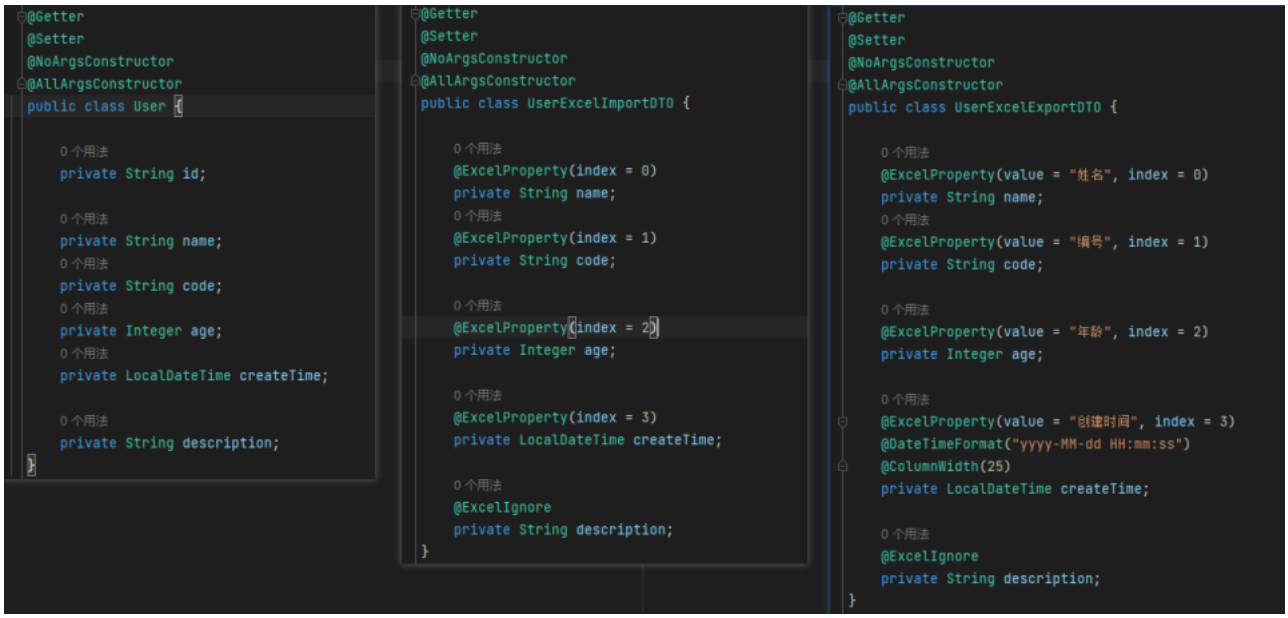
> 这里的boolean是原生类型
>
> - 被解析为true

	A	B	C	D	E	F	G
1	姓名	编号	年龄	创建时间	tag		
2	AAA	0001	12	2023-02-12 22:54:15	0		
3	BBB	0002	14	2023/2/13 22:24	1		
4	CCC	0003	15	2023/2/14 22:24	TRUE		
5	DDD	0004	22	2023/2/12 22:24	FALSE		
6	EEE	0005	24	2023/2/13 22:24	True		
7	FFF	0006	25	2023/2/14 22:24	fAlSe		
8	GGG	0007	25	2023/2/14 22:24			
9	HHH	0008	25	2023/2/14 22:24	-		
10							

三、为什么要使用DTO?

DataTransformObject

- 

- 

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class User {

    0 个用法
    private String id;

    0 个用法
    private String name;
    0 个用法
    private String code;
    0 个用法
    private Integer age;
    0 个用法
    private LocalDateTime createTime;

    0 个用法
    private String description;
}
```

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class UserExcelImportDTO {

    0 个用法
    @ExcelProperty(index = 0)
    private String name;
    0 个用法
    @ExcelProperty(index = 1)
    private String code;

    0 个用法
    @ExcelProperty(index = 2)
    private Integer age;

    0 个用法
    @ExcelProperty(index = 3)
    private LocalDateTime createTime;

    0 个用法
    @ExcelIgnore
    private String description;
}
```

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class UserExcelExportDTO {

    0 个用法
    @ExcelProperty(value = "姓名", index = 0)
    private String name;
    0 个用法
    @ExcelProperty(value = "编号", index = 1)
    private String code;

    0 个用法
    @ExcelProperty(value = "年龄", index = 2)
    private Integer age;

    0 个用法
    @ExcelProperty(value = "创建时间", index = 3)
    @DateTimeFormat("yyyy-MM-dd HH:mm:ss")
    @ColumnWidth(25)
    private LocalDateTime createTime;

    0 个用法
    @ExcelIgnore
    private String description;
}
```

1. 使用注解控制更方便；

2. 业务逻辑中更直观，不需要反复在User上动来动去，User实际上只是纯粹的数据库中表对应的实体。