

# BooksBATCH

Ever Hinojosa Aguirre

# Process

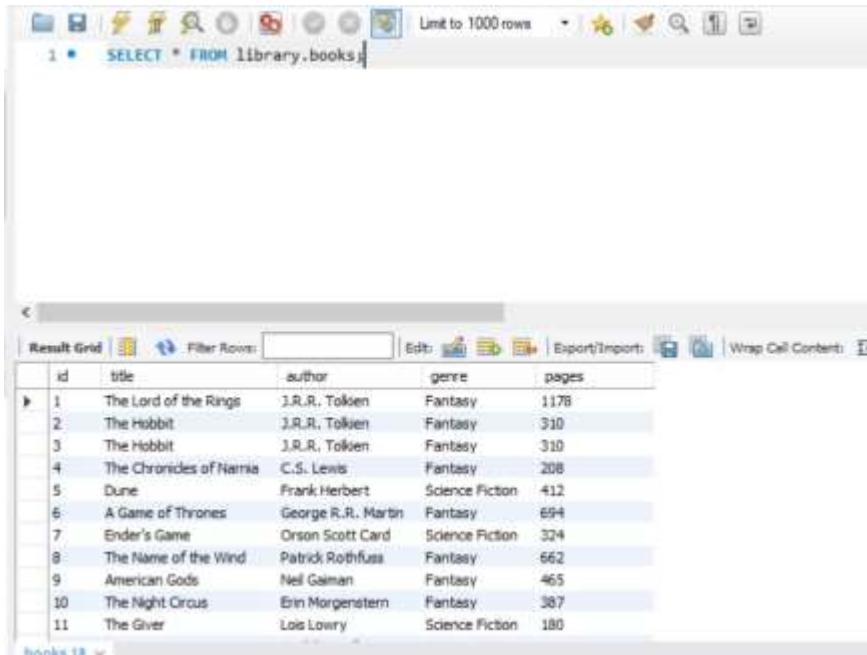
- ▶ Taking on the theme of books, the batch was adjusted to take in a list of 500 books( some of them are duplicates due to the csv being AI generated), and as i process them i take the ones with the genre Sciene finction and Fantasy

```
pa [academyMty24 main]ademia.BooksBatch.config;
2
3 import org.springframework.batch.core.Job;
25
26
27 @Configuration
28 @EnableBatchProcessing
29 @AllArgsConstructor
30 public class SpringBatchConfig {
31
32     private final JobBuilderFactory jobBuilderFactory;
33     private final StepBuilderFactory stepBuilderFactory;
34     private final BookRepository bookRepository;
35
36     @Bean
37     public FlatFileItemReader<Book> reader() {
38         FlatFileItemReader<Book> itemReader = new FlatFileItemReader<>();
39         itemReader.setResource(new FileSystemResource("src/main/resources/books.csv")); // Update path and filename if
40         itemReader.setName("csvReader");
41         itemReader.setLinesToSkip(1);
42         itemReader.setLineMapper(lineMapper());
43         return itemReader;
44     }
45
46     private LineMapper<Book> lineMapper() {
47         DefaultLineMapper<Book> lineMapper = new DefaultLineMapper<>();
48
49         DelimitedLineTokenizer lineTokenizer = new DelimitedLineTokenizer();
50         lineTokenizer.setDelimiter(",");
51         lineTokenizer.setStrict(false);
52         lineTokenizer.setNames("id", "title", "author", "genre", "pages");
53
54         BeanWrapperFieldSetMapper<Book> fieldSetMapper = new BeanWrapperFieldSetMapper<>();
55         fieldSetMapper.setTargetType(Book.class);
56
```

```
booksbatch src/main/java com.academia.booksbatch.config BOOKPROCESSOR
1 package com.academia.BooksBatch.config;
2
3 import org.springframework.batch.item.ItemProcessor;
5
6 public class BookProcessor implements ItemProcessor<Book, Book> {
7
8     @Override
9     public Book process(Book book) throws Exception {
10         if ("Science Fiction".equals(book.getGenre()) || "Fantasy".equals(book.getGenre())) {
11             return book;
12         } else {
13             return null;
14         }
15     }
16 }
```

# Books

- ▶ The entity is adjusted to accomodate my table books with the lombok to have less code. I truncated my table before running my job



The screenshot shows a SQL query result in a database client. The query is `SELECT * FROM library.books;`. The result is displayed in a table with 5 columns: `id`, `title`, `author`, `genre`, and `pages`. The table contains 11 rows of data.

id	title	author	genre	pages
1	The Lord of the Rings	J.R.R. Tolkien	Fantasy	1178
2	The Hobbit	J.R.R. Tolkien	Fantasy	310
3	The Hobbit	J.R.R. Tolkien	Fantasy	310
4	The Chronicles of Narnia	C.S. Lewis	Fantasy	208
5	Dune	Frank Herbert	Science Fiction	412
6	A Game of Thrones	George R.R. Martin	Fantasy	694
7	Ender's Game	Orson Scott Card	Science Fiction	324
8	The Name of the Wind	Patrick Rothfuss	Fantasy	662
9	American Gods	Neil Gaiman	Fantasy	465
10	The Night Circus	Erin Morgenstern	Fantasy	387
11	The Giver	Lois Lowry	Science Fiction	180

```
BooksBatch ▸ src/main/java ▸ com.academia.BooksBatch.entity ▸ Book ▸
1 package com.academia.BooksBatch.entity;
2
3 import lombok.AllArgsConstructor;
4
13
14 @Entity
15 @Table(name = "BOOKS")
16 @Data
17 @AllArgsConstructor
18 @NoArgsConstructor
19 public class Book {
20
21     @Id
22     @GeneratedValue(strategy = GenerationType.IDENTITY)
23     @Column(name = "ID")
24     private Integer id;
25
26     @Column(name = "TITLE", nullable = false)
27     private String title;
28
29     @Column(name = "AUTHOR", nullable = false)
30     private String author;
31
32     @Column(name = "GENRE")
33     private String genre;
34
35     @Column(name = "PAGES")
36     private Integer pages;
37 }
```

# Postman

- ▶ I took on the same configuration as the original project so to run the job it needs to be a POST method.

<http://localhost:9191/jobs/importBooks>

```
1 package com.academia.booksbatch.controller;
2
3 import org.springframework.batch.core.Job;
4
5 @RestController
6 @RequestMapping("/jobs")
7 public class JobController {
8
9     @Autowired
10    private JobLauncher jobLauncher;
11
12    @Autowired
13    private Job job;
14
15    @PostMapping("/importBooks")
16    public void importCsvToDBJob() {
17        JobParameters jobParameters = new JobParametersBuilder()
18            .addLong("startAt", System.currentTimeMillis()).toJobParameters();
19
20        try {
21            jobLauncher.run(job, jobParameters);
22        } catch (JobExecutionAlreadyRunningException | JobRestartException | JobInstanceAlreadyCompleteException | JobPa
23            e.printStackTrace();
24        }
25    }
26 }
```