# Padrões de desenvolvimento para aplicações Spring Batch

Giuliana Silva Bezerra

# Giuliana S. Bezerra

Arquiteta de Software
Instrutora Online

Email: giu.drawer@gmail.com

LinkedIn: giulianabezerra

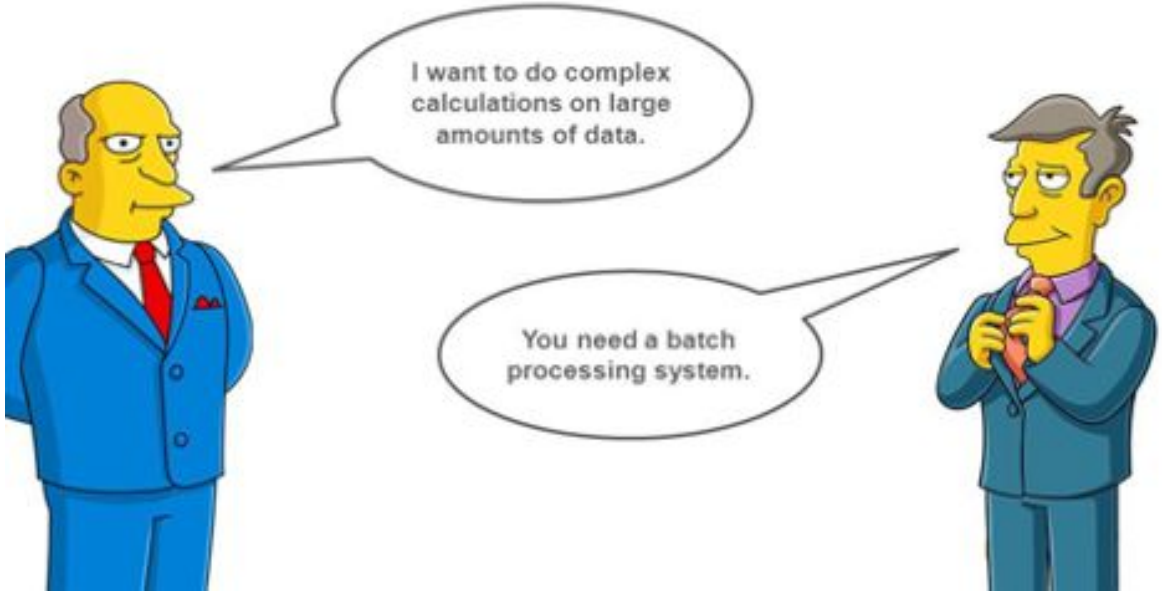
Medium: @giuliana-bezerra

Github: giuliana-bezerra

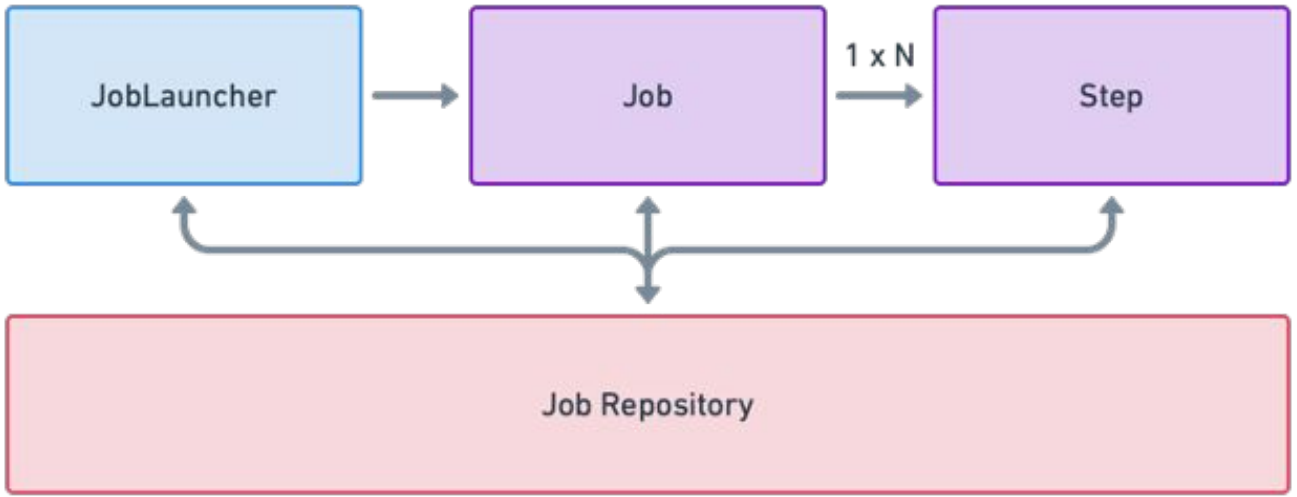
# Visão Geral

1. O que é Spring Batch?
2. Arquitetura do Spring Batch
3. Padrões e boas práticas

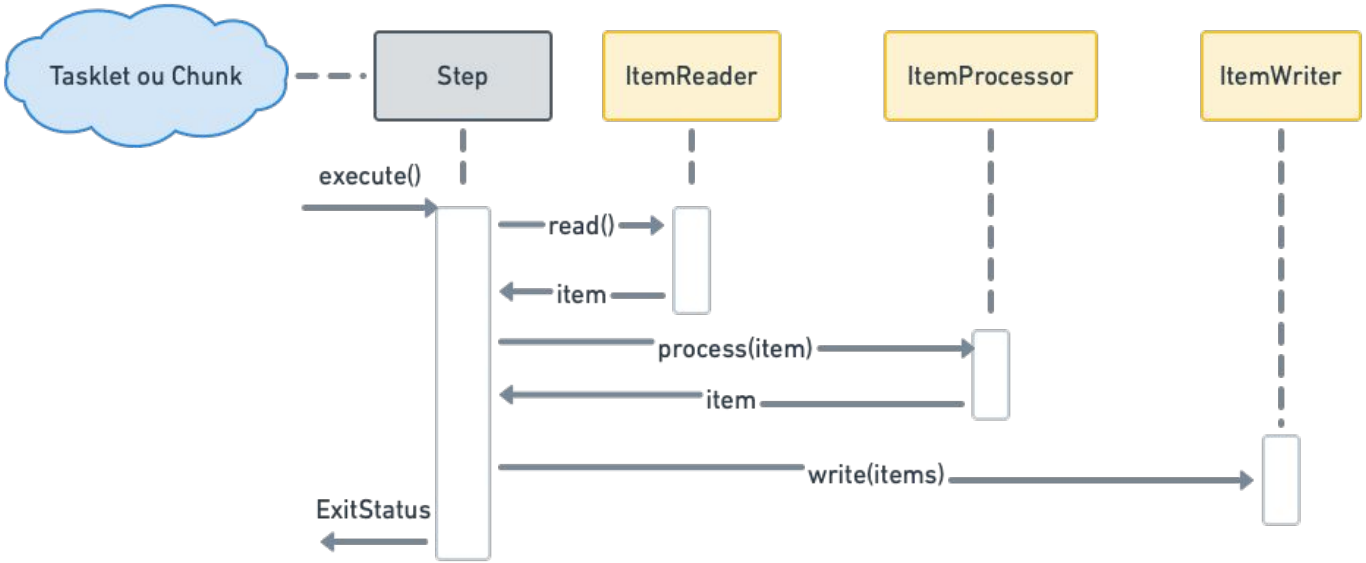
# O que é Spring Batch?




# Arquitetura do Spring Batch




# Arquitetura do Spring Batch




# Padrões e boas práticas

Recomendações baseadas em experiência


# 1. Separação de Responsabilidades

```java
@Configuration
@EnableBatchProcessing
public class PrimeiroJobSpringBatchConfig {
    @Autowired
    private JobBuilderFactory jobBuilderFactory;

    @Bean
    public Job imprimeOlaJob(Step imprimeOlaStep) {
        return jobBuilderFactory
                .get("imprimeOlaJob")
                .start(imprimeOlaStep)
                .build();
    }
}
```

```java
@Configuration
public class ImprimeOlaStepConfig {
    @Autowired
    private StepBuilderFactory stepBuilderFactory;

    @Bean
    public Step imprimeOlaStep(Tasklet imprimeOlaTasklet) {
        return stepBuilderFactory
                .get("imprimeOlaStep")
                .tasklet(imprimeOlaTasklet)
                .build();
    }
}
```

# 2. Encapsulamento de lógicas em fluxos

```java
@Bean
public Job job() {
    return this.jobBuilderFactory.get("faturaCartaoCredito")
                .start(consolidarTransacoes())
                .next(gerarFaturaCartaoCredito())
                .end()
                .build();
}
```

```java
@Bean
public Flow consolidarTransacoes() {
    return new FlowBuilder<SimpleFlow>("consolidarTransacoes")
            .start(processarTransacoes())
            .next(gerarTransacoesFatura())
            .next(tratarErrosTransacoes())
            .build();
}
```

# 3. Salvar metadados no banco de dados

```
spring.datasource.jdbcUrl=jdbc:mysql://localhost:3306/spring_batch
spring.datasource.username=user
spring.datasource.password=123456
app.datasource.jdbcUrl=jdbc:mysql://localhost:3306/migracao_dados
app.datasource.username=user
app.datasource.password=123456
spring.batch.initialize-schema=always
```

```
@Configuration
public class DataSourceConfig {
  @Primary
  @Bean
  @ConfigurationProperties(prefix="spring.datasource")
  public DataSource springDataSource() {
    return DataSourceBuilder.create().build();
  }

  @Bean
  @ConfigurationProperties(prefix="app.datasource")
  public DataSource appDataSource() {
    return DataSourceBuilder.create().build();
  }
}
```

# 4.1. Minimizar uso de I/O

```java
public class SavingItemWriter implements ItemWriter<Object> {
  private StepExecution stepExecution;

  public void write(List<? extends Object> items) throws Exception {
    // ...

    ExecutionContext stepContext = this.stepExecution.getExecutionContext();
    stepContext.put("someKey", someObject);
  }

  @BeforeStep
  public void saveStepExecution(StepExecution stepExecution) {
    this.stepExecution = stepExecution;
  }
}
```

```java
public class RetrievingItemWriter implements ItemWriter<Object> {
  private Object someObject;

  public void write(List<? extends Object> items) throws Exception {
    // ...
  }

  @BeforeStep
  public void retrieveInterstepData(StepExecution stepExecution) {
    JobExecution jobExecution = stepExecution.getJobExecution();
    ExecutionContext jobContext = jobExecution.getExecutionContext();
    this.someObject = jobContext.get("someKey");
  }
}
```

# 4.2. Minimizar uso de I/O

```java
@Bean
public JdbcCursorItemReader<CustomerCredit> itemReader() {
  return new JdbcCursorItemReaderBuilder<CustomerCredit>()
    .dataSource(this.dataSource)
    .name("creditReader")
    .sql("select ID, NAME, CREDIT from CUSTOMER")
    .rowMapper(new CustomerCreditRowMapper())
    .build();

}
```


| ID | NAME | CREDIT |
|---|---|---|
| 1 | name1 | credit1 |
| 2 | name2 | credit2 |
| 3 | name3 | credit3 |
| 4 | name4 | credit4 |

# 5. Ordenação de consultas

```
@Bean
public JdbcCursorItemReader<CustomerCredit> itemReader() {
  return new JdbcCursorItemReaderBuilder<CustomerCredit>()
    .dataSource(this.dataSource)
    .name("creditReader")
    .sql("select ID, NAME, CREDIT from CUSTOMER ORDER BY ID")
    .rowMapper(new CustomerCreditRowMapper())
    .build();

}
```

# 6. Uso correto de tasklets

```java
@Override
public RepeatStatus execute(StepContribution contribution, ChunkContext
chunkContext) throws Exception {
  List<DataDTO> list = dbService.getData();
  List<Data> chunkData = new ArrayList<>();

  for(int i = 0; i < list.size(); i++) {
    if (i % 1000 == 0) {
      chunkData.add(new Data(list.get(i)));
      jdbcTemplate.batchUpdate(
        "UPDATE data SET column = ? WHERE id = ?",
        chunkData,
        1000,
        new ParameterizedPreparedStatementSetter<Data>() {
          // Omitido por simplicidade
        }
      )
    } else {
      List<Data> chunkData = new ArrayList<>();
      chunkData.add(new Data(list.get(i)));
    }
  }
}
```

# 7. Uso de logs

```java
public class ItemFailureLoggerListener extends ItemListenerSupport {

    private static Log logger = LogFactory.getLog("item.error");

    public void onReadError(Exception ex) {
        logger.error("Encountered error on read", e);
    }

    public void onWriteError(Exception ex, List<? extends Object> items) {
        logger.error("Encountered error on write", ex);
    }
}
```

# 8. Coleta de métricas

```
@Bean
public LoggingMeterRegistry loggingMeterRegistry() {
  return new LoggingMeterRegistry(new LoggingRegistryConfig() {
    @Override
    public Duration step() {
        return Duration.ofMinutes(20L);
    }
    @Override
    public String get(String key) {
        return null;
    }
  }, Clock.SYSTEM);
}
```

| *Metric Name* | *Type* |
|---|---|
| `spring.batch.job` | `TIMER` |
| `spring.batch.job.active` | `LONG_TASK_TIMER` |
| `spring.batch.step` | `TIMER` |
| `spring.batch.item.read` | `TIMER` |
| `spring.batch.item.process` | `TIMER` |
| `spring.batch.chunk.write` | `TIMER` |

# 9. Tolerância à falhas

```java
@Bean
public Step step1() {
    return this.stepBuilderFactory.get("step1")
                .<String, String>chunk(10)
                .reader(itemReader())
                .writer(itemWriter())
                .startLimit(1)
                .allowStartIfComplete(true)
                .faultTolerant()
                .skipLimit(10)
                .skip(FlatFileParseException.class)
                .retryLimit(3)
                .retry(DeadlockLoserDataAccessException.class)
                .noRollback(ValidationException.class)
                .build();
}
```

# 10. Leitores compostos

```
@Configuration
public class CustomReader() {
   @Autowired
   private PageProcessor1 pageProcessor1;

   @Autowired
   private PageProcessor2 pageProcessor2;

   @Bean
   public JdbcPagingItemReader<Data> reader() {
      CompositeJdbcPagingItemReader<Data> jdbcReader =
        new CompositeJdbcPagingItemReader<>();
      // Omitido por simplicidade
      jdbcReader.addPageProcessor(pageProcessor1);
      jdbcReader.addPageProcessor(pageProcessor2);
      return jdbcReader;
}
```

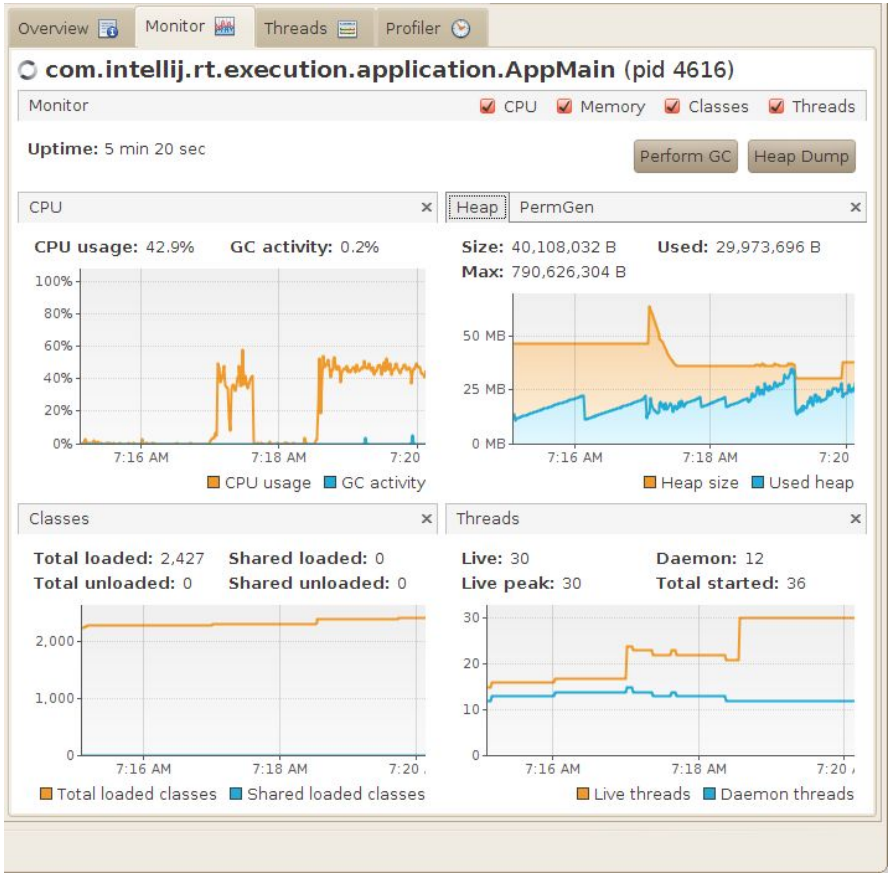
```
public class PageProcessor1 implements PageProcessor<Data> {
  private NamedParameterJdbcTemplate jdbcTemplate;

  public void setDataSource(DataSource dataSource) {
      jdbcTemplate = new NamedParameterJdbcTemplate(dataSource);
  }

  @Override
  public void process(List<Data> page) {
      Map<Integer, List<String>> list = new HashMap<>();
      MapSqlParameterSource parameters = new MapSqlParameterSource();
      parameters.addValue("ids", page.stream().map(o ->
o.getId()).collect(Collectors.toList()));
      jdbcTemplate.query("select * from data_info where id in (:ids) order
by id",
                parameters,
                dataExtractor());
      page.stream().forEach(o -> o.setDataInfo(

 list.get(o.getId()).stream().collect(Collectors.joining","))));
  }
}
```

# 11. Profiling



# Referências

- Documentação Oficial
- The Definitive Guide to Spring Batch - Michael Minella
- Cursos da Udemy
  - Curso para desenvolvimento de jobs com Spring Batch
  - Otimização de desempenho para jobs Spring Batch

# Obrigada!


Giuliana Silva Bezerra
Arquiteta de Software
Instrutora Online

Email: giu.drawer@gmail.com

LinkedIn: giulianabezerra

Medium: @giuliana-bezerra

Github: giuliana-bezerra