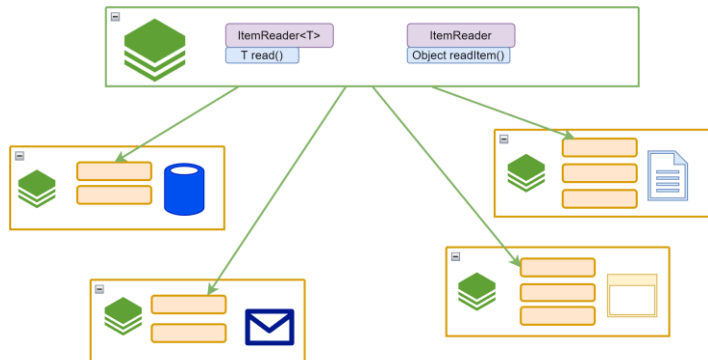


Leitores do Spring Batch

Todos os **leitores** do **Spring Batch** herdam da **Interface `ItemReader<T>`** (que pode ou não ser parametrizada), essa interface possui as assinaturas de método necessárias para as **implementações concretas de leitores**:



Arquivos flat

Um **arquivo flat** é um arquivo que possui **dados não estruturados**. E “**dados não estruturados**” simplesmente quer dizer que quando olhamos o conteúdo não dá pra saber o formato ou significado do que está representado **ali**.

Arquivos baseados em **largura fixa** (cada propriedade é representada por um determinado número de **colunas/caracteres**), **delimitadores** (cada propriedade é separada por algum delimitador), e de **múltiplos formatos** (contém mais de um objeto de negócio) são considerados **arquivos flat**:

Nome	Chaveiro	734191 Avenue	Mobile	AL35928	Field	Length
James	102 Street	8812 In St.	Saint Paul	PN07121	Item Name	11
Hogan	Wheeler	4812 Nec. Ave.	Gulfport	PN23193	Mobile Number	1
Sydney	Mobilecon	884 Omeare Ave	Olathe	KS23680	Last Name	10
				Address Number	4	
				Street	20	
				City	10	
				State	2	
				Zip Code	5	

Aimee, G, Hoover, 734191 Avenue, Mobile, AL, 35928

Jonas, U, Gilbert, 8852, In St., Saint Paul, MN, 57321

Regan, M, Baxter, 4851, Nec Av., Gulfport, MS, 33193

CUST, Marten, O, Darrow, 8272 4th Street, New York, IL, 76091

TRANS, 1165965, 2011-05-22 00:13:29, 51.43

CUST, Ann, V, Gates, 9247 Infinite Loop Drive, Hollywood, NE, 37612

CUST, Erica, I, Jobs, 8875 Farman Street, Aurora, IL, 36314

TRANS, 8116369, 2011-01-21 20:40:52, -14.48

TRANS, 8116369, 2011-01-21 15:50:17, -45.45

TRANS, 8116369, 2011-01-21 16:52:46, -74.6

TRANS, 8116369, 2011-01-22 13:51:09, 48.55

TRANS, 8116369, 2011-01-21 16:51:59, 98.53

Lendo um arquivo flat delimitado

Já que o **Spring Batch** tem uma implementação para leituras de arquivo flat, vamos usa-la (**`FlatFileItemReader<T>`**) para ler um arquivo do tipo “**delimitado**” (cada atributo separado por um delimitador).

A configuração desse leitor pode ser feita através de um builder, onde eu digo o **resource** (nesse caso o **arquivo**) fonte da leitura, dizemos que **ele é do tipo “tamanho fixo”**, ainda podemos dizer qual o **critério de delimitação** (embora seja opcional?):

```
@Configuration
public class LeituraArquivoDelimitadoReaderConfig {

    @StepScope
    @Bean
    public FlatFileItemReader<Cliente> leituraArquivoDelimitadoReader(
        @Value("#{jobParameters['arquivoClientes']}") Resource arquivoClientes) {

        return new FlatFileItemReaderBuilder<Cliente>()
            .name("leituraArquivoDelimitadoReader")
            .resource(arquivoClientes)
            .delimited()
            .delimiter(",")
            .names("nome", "sobrenome", "idade", "email")
            .targetType(Cliente.class)
            .build();
    }
}
```

clientes.txt

João, Silva, 34, joao@test.com

Maria, Silva, 36, maria@test.com

Jose, Silva, 29, jose@test.com

```
@Configuration
public class LeituraArquivoDelimitadoReaderConfig {

    @StepScope
    @Bean
    public FlatFileItemReader<Cliente> leituraArquivoDelimitadoReader(
        @Value("#{jobParameters['arquivoClientes']}") Resource arquivoClientes) {

        return new FlatFileItemReaderBuilder<Cliente>()
            .name("leituraArquivoDelimitadoReader")
            .resource(arquivoClientes)
            .delimited()
            .delimiter(",")
            .names("nome","sobrenome","idade","email")
            .targetType(Cliente.class)
            .build();
    }
}
```

```
public class Cliente {
    private String nome;
    private String sobrenome;
    private String idade;
    private String email;

    public String getNome() { return nome; }

    public void setNome(String nome) { this.nome = nome; }
}
```

The screenshot displays two parts of the development environment:

- Left Panel (Code Editor):** Shows the configuration class `@Configuration public class LeituraArquivoLarguraFixaReaderConfig {`. Inside, there's a `@StepScope @Bean` annotated method:


```
public FlatFileItemReader<Cliente> leituraArquivoLarguraFixaReader(
    @Value("#{jobParameters['arquivoClientes']}") Resource arquivoCliente){}
```

 The value expression `"#{jobParameters['arquivoClientes']}"` is highlighted with a red box.
- Right Panel (IDE Interface):** Shows the "Build and run" tab. At the top, it indicates "java 18 SDK of 'ArquivoLarguraFixa'" and "com.springbatch.arq". Below, the command `arquivoClientes -file:files/clientes.txt` is entered in a field, with `-file:files/clientes.txt` highlighted by a red box. To the right, a file explorer shows a folder named `files` containing a file named `clientes.txt`, both also highlighted with red boxes.

O tamanho do chunk influencia em como o job vai ser reexecutado em caso de falha, isso porque o chunk é uma unidade de contagem do job. Se eu tenho **100 registros** divididos em **10 chunks** de tamanho 10 (**10 registros são commitados por chunk**), então eu vou ter uma **contagem de 0 a 9** para esses chunks.

The screenshot shows the SQL query editor and the result grid. The query is:

```
SELECT * FROM batch_process BATCH_STEP_EXECUTION_CONTEXT;
```

The result grid shows the following data:

DN_ID	SHORT_CONTEXT
1	{batch.taskletType:"org.springframework.batch.core.step.item.ChunkOrientedTasklet","clientReader.read.coun...
2	{batch.taskletType:"org.springframework.batch.core.step.item.ChunkOrientedTasklet","clientReader.read.coun...
3	{batch.taskletType:"org.springframework.batch.core.step.item.ChunkOrientedTasklet","clientReader.read.coun...
4	{batch.taskletType:"org.springframework.batch.core.step.item.ChunkOrientedTasklet","clientReader.read.coun...

```
{ "batch.taskletType": "org.springframework.batch.core.step.item.ChunkOrientedTasklet", "clienteReader.read.count": 4, "batch.stepType": "org.springframework.batch.core.step.item.ChunkOrientedTasklet" }
```

```
{ "batch.taskletType": "org.springframework.batch.core.step.item.ChunkOrientedTasklet", "clienteReader.read.count": 0, "batch.stepType": "org.springframework...
```