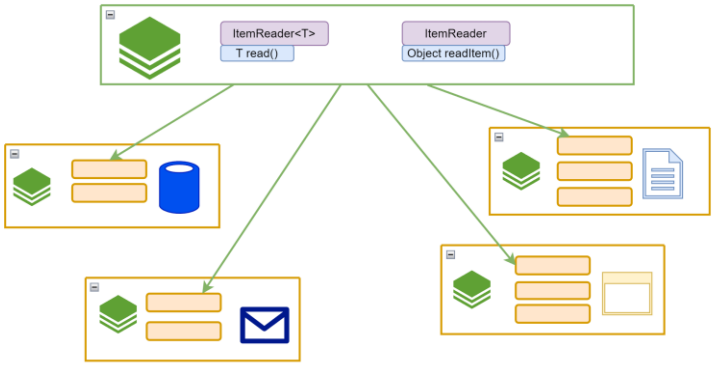
Leitores do Spring Batch

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



Arquivos flat

Um **arquivo flat** é um arquivo que possuí ***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**:

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?)**:

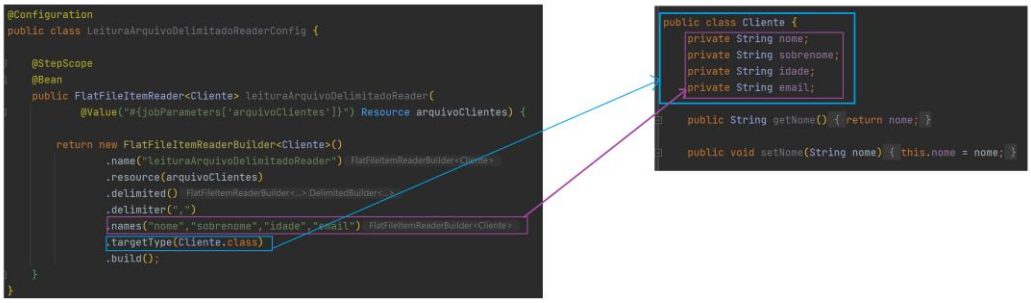
```java
@Configuration
public class LeituraArquivoDelimitadoReaderConfig {

    @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,32,joao@test.com
Maria,Silva,30,maria@test.com
José,Silva,20,jose@test.com
```

Em seguida **podemos nomear esses atributos** para que sejam convertidos em uma **classe de domínio**:

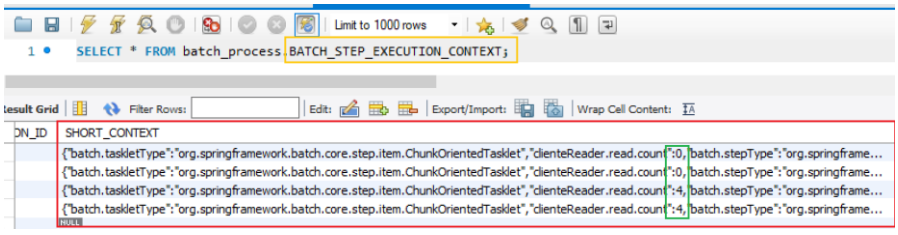


Outro ponto interessante aqui é a anotação **@StepScope** no reader, isso acontece porque estou **injetando um valor nesse Reader através de um argumento**, mas para que esse valor seja lido eu preciso dizer que o Reder está no escopo do Step, já que o batch chama o job que chama o step:

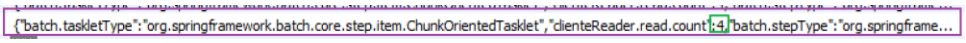
Restart na leitura

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.

O que representa a **quantidade de chunks executados no banco de dados** é o atributo **SHORT_CONTEXT** da tabela **BATCH_STEP_EXECUTION_CONTEXT**, nele podemos ver o **count** que representa qual o próximo chunk daquele step a ser executado:



nesse **exemplo aqui** eu li um arquivo com 3 registros onde cada **chunk tinha tamanho igual a 1**, depois de executar os **3 chunks com sucesso**, o próximo **chunk é o 4:**



Aqui **nenhum chunk foi feito com sucesso**, o que indica que o **próximo chunk desse step é o 0:**