## BL-07 ファイルアクセス機能

### 概要

### 機能概要

- ファイルアクセス機能は、TERASOLUNA Batch Framework for Java ver 2.x で使用 していたファイルアクセス機能と同一のものを利用して、ファイルアクセスを行
- 本項目では、TERASOLUNA Batch Framework for Java ver 2.x との違いのみを説明 するものとし、ファイルアクセス機能の詳細な説明は TERASOLUNA Batch Framework for Java 機能説明書(引用資料)の「BC-01 ファイルアクセス機能」を参 照すること。

### 概念図の違いについて

- TERASOLUNA Batch Framework for Java 機能説明書(引用資料)の「BC-01 ファイ ルアクセス機能」では、概念図「ファイル入力処理の概念図(Collector から利用さ れる場合)」において Collector から利用される場合の概念図を掲載しているが、 TERASOLUNA Batch Framework for Java ver 3.x では、このような使い方はしない ため、読み飛ばすこと。
- TERASOLUNA Batch Framework for Java ver 3.x でファイルアクセス機能を利用す るときはビジネスロジックから利用するので、「ファイル入力処理の概念図(ビジ ネスロジックから利用される場合)」の概念図を参考にすること。

### コーディングポイント

- 本説明書でのコーディングポイントは、別資料の「BC-01 ファイルアクセス機 能」のコーディングポイントと異なる以下の項目についてのみ説明を行う。
  - ファイル入力チェックについて
  - 例外処理
  - ファイル入力の実装例
  - ・ ファイル出力の実装例

その他の項目については、別資料の「BC-01 ファイルアクセス機能」を参照する こと。

次ページからのコーディングポイントの中で、TERASOLUNA Batch Framework for Java ver 3.x で本機能を使用する際に大事なポイントについては、二重線の吹き出 しを使用して強調している。



ファイル入力チェックについて

TERASOLUNA Batch Framework for Java ver 3.x ではファイル、DB から取得したデ ータの入力チェックを行うための「AL-043 入力チェック機能」を提供している。 使用方法についての詳細は「AL-043 入力チェック機能」の機能説明書を参照する こと。

例外処理について

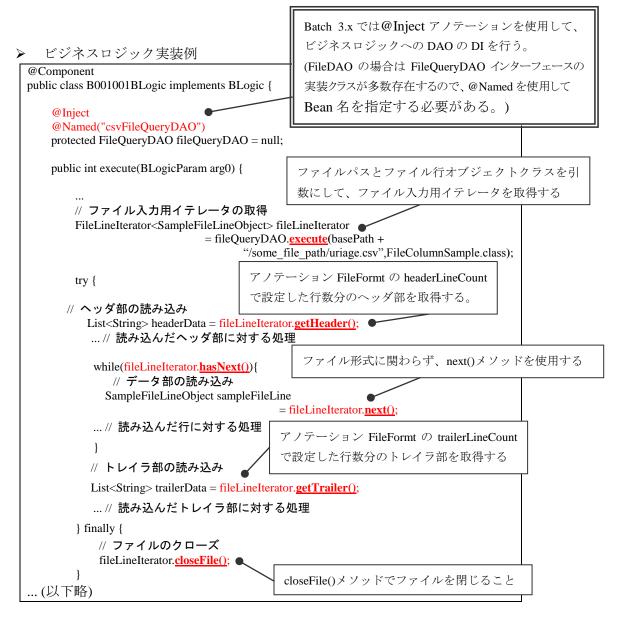
TERASOLUNA Batch Framework for Java ver 3.x では例外処理のための「BL-04 例 外ハンドリング機能」を提供している。

使用方法についての詳細は「BL-04 例外ハンドリング機能」の機能説明書を参照 すること。

- ファイル入力の実装例
  - ▶ ファイル入力処理の実装
  - (1) ファイル行オブジェクトを実装する。

上記の例ではパッケージ[jp.terasoluna.batch.sample.sample00001]内に存在する @Component アノテーションが付与されたクラスを自動的に検出する。

(3) ファイル入力処理を行うクラスでは、FileQueryDAOの execute()メソッドでファイル入力用イテレータを取得する。ファイル入力用イテレータ取得時に、ファイルオープンが行われる。ファイル入力用イテレータの next メソッドで、ファイル行オブジェクトを取得する。



#### ▶ ファイルの入力順序

トレイラ部の入力は、データ部の入力が全て終わった後に行う必要がある点に 留意すること。

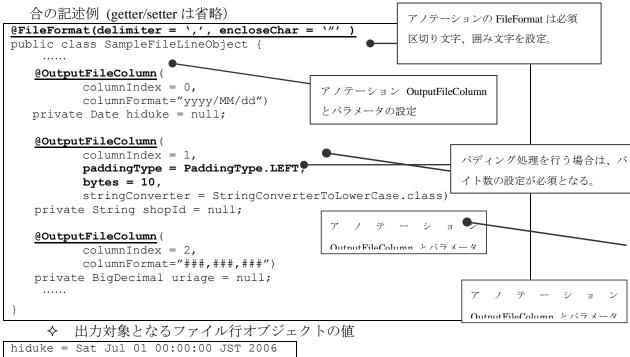
▶ スキップ処理

ファイル入力機能では入力を開始する行を指定する事ができる。 スキップ処理は中断していたファイルの読み込みを再開するような場合に使用 する事が出来る。



#### ● ファイル出力の実装例

▶ ファイルの設定として、囲み文字と区切り文字を設定し、データの一部をデフォルトのパディング文字でパディング処理したデータをファイルに出力する場



```
hiduke = Sat Jul 01 00:00:00 JST 2006
shopId = SHOP01
uriage = 1000000
```

◆ 上記のファイル行オブジェクトを出力すると以下の値となる。

```
"2006/07/01"," shop01","1,000,000"
```

データの一部を個別のパディング文字でパディング処理したデータをファイル に出力する場合の記述例 (getter/setter は省略)

```
@FileFormat(delimiter = ',', encloseChar = '"' )
public class SampleFileLineObject {
   \underline{\texttt{@OutputFileColumn}}\ (
          columnIndex = 0,
          columnFormat="yyyy/MM/dd")
   private Date hiduke = null;
   @OutputFileColumn (
          columnIndex = 1,
                                                   右側のパディング処理を行い、パディング文
          paddingType = PaddingType.RIGHT,
          paddingChar = '0',
                                                   字として'0'を設定する。
          bytes = 10,
          stringConverter = StringConverterToLowerCase.class)
   private String shopId = null;
   @OutputFileColumn (
          columnIndex = 2,
          columnFormat="###,###,###")
   private BigDecimal uriage = null;
                                                    Output File Column レパラメータ
```

出力対象となるファイル行オブジェクトの値

```
hiduke = Sat Jul 01 00:00:00 JST 2006
shopId = SHOP01
uriage = 1000000
```

上記のファイル行オブジェクトを出力すると以下の値となる。

"2006/07/01","shop010000","1,000,000"

ファイル出力処理の実装(1メソッドでファイルをオープン・クローズする場 合)

- (1) ファイル行オブジェクトを実装する。
- (2) ファイル出力処理を行うクラスの Bean 定義を行う。 【ジョブ Bean 定義ファイルの設定例】

Batch 3.x では@Component アノテーショ ンを使用して、Bean を自動的に検出 し、DI コンテナで管理する。

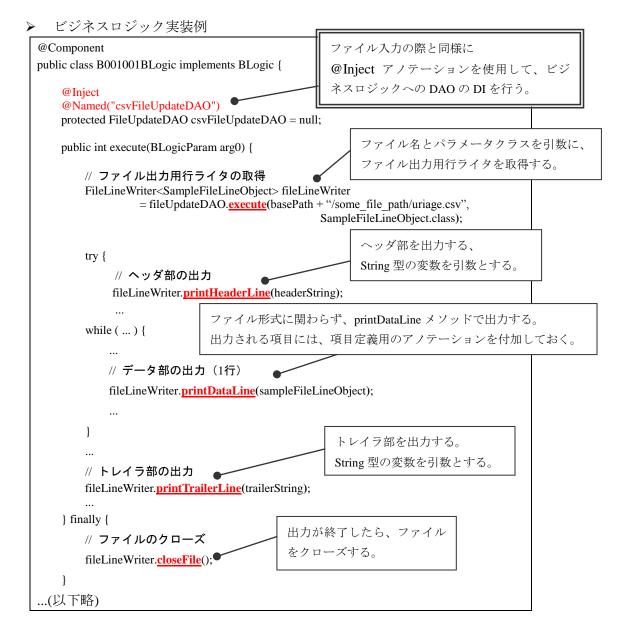
<!-- コンポーネントスキャン設定 -->

<context:component-scan base-package="jp.terasoluna.batch.sample.Sample00001" />

上記の例ではパッケージ[jp.terasoluna.batch.sample.Sample00001]内に存在する @Component アノテーションが付与されたクラスを自動的に検出する。

※ファイル入力処理の Bean 定義の際に上記の記述を行っていた場合は、 改めて記述する必要はない。

(3) ファイル出力処理を行うクラスでは、FileUpdateDAO の execute メソッドでフ ァイル出力用行ライタを取得する。ファイル出力用行ライタの取得時に、フ ァイルがオープンされる。



機能名 BL-07 ファイルアクセス機能 ページ BL-07\_09

# ◆ 拡張ポイント

• なし

# ■ 関連機能

- 『AL-043 入力チェック機能』
- 『BL-04 例外ハンドリング機能』

# ■ 使用例

- 機能網羅サンプル(terasoluna-batch-functionsample)
- チュートリアル(terasoluna-batch-tutorial)

# ■ 備考

• なし