# TERASOLUNA 3.3.1 移行ガイド(Batch 版)

## 変更履歴

バージョン	日付	改訂箇所	改訂内容
3.3.1	2015/02/23	_	新規作成

## ■ 概要

本ドキュメントは、TERASOLUNA Batch Framework for Java 3.2.0 で作成したアプ リケーションを、3.3.1へ移行する際の手順を示すドキュメントである。

## ◆ 3.3.1 の変更点概要

- 依存ライブラリのバージョンアップ
- Spring Bean 定義ファイルの XML スキーマ定義(XSD)からバージョン指定を削除
- バグの修正

## ■ 移行手順

3.2.0 から 3.3.1 への移行手順を説明する。

## 前提条件

TERASOLUNA Batch Framework for Java 3.2.0 を使用したアプリケーションが正 常に動作していること。

以後、本書ではこれをアプリケーションと呼称する。

## ① blank プロジェクトのダウンロード

TERASOLUNA Batch Framework for Java 3.3.1 の blank プロジェクト (terasoluna-batch4j-blank\_3.3.1.zip) を以下の URL よりダウンロードし、任 意のフォルダに展開する。

ダウンロードサイトの URL:

SourceForge. jp <a href="http://sourceforge.jp/projects/terasoluna/releases/">http://sourceforge.jp/projects/terasoluna/releases/</a>

## ② 依存ライブラリの変更

TERASOLUNA フレームワークや Spring の依存ライブラリの差し替えを行う。 差し替えるライブラリは以下の通りである。

3.2.0のライブラリ	3.3.1のライブラリ	種別
cglib-nodep-2.1_3.jar	-	削除
spring-asm-3.1.3.RELEASE.jar	_	削除
aspectjweaver-1.6.12.jar	aspectjweaver-1.7.4.jar	差し替え
commons-collections-3.2.jar	commons-collections-3.2.1.jar	差し替え
commons-digester-1.8.jar	commons-digester-2.0.jar	差し替え
commons-logging-1.1.1.jar	commons-logging-1.1.3.jar	差し替え
spring-aop-3.1.3.RELEASE.jar	spring-aop-3.2.13.RELEASE.jar	差し替え
spring-beans-3.1.3.RELEASE.jar	spring-beans-3.2.13.RELEASE.jar	差し替え
spring-context-3.1.3.RELEASE.jar	spring-context-3.2.13.RELEASE.jar	差し替え
spring-core-3.1.3.RELEASE.jar	spring-core-3.2.13.RELEASE.jar	差し替え
spring-expression-3.1.3.RELEASE.jar	spring-expression-3.2.13.RELEASE.jar	差し替え
spring-jdbc-3.1.3.RELEASE.jar	spring-jdbc-3.2.13.RELEASE.jar	差し替え
spring-orm-3.1.3.RELEASE.jar	spring-orm-3.2.13.RELEASE.jar	差し替え
spring-tx-3.1.3.RELEASE.jar	spring-tx-3.2.13.RELEASE.jar	差し替え
terasoluna-batch-3.2.0.jar	terasoluna-batch-3.3.1.jar	差し替え
terasoluna-batch-update-1.1.2.jar	terasoluna-batch-update-3.3.1.jar	差し替え
terasoluna-collector-1.1.0.jar	terasoluna-collector-3.3.1.jar	差し替え
terasoluna-commons-2.0.5.0.jar	terasoluna-commons-3.3.1.jar	差し替え
terasoluna-dao-2.0.5.0.jar	terasoluna-dao-3.3.1.jar	差し替え
terasoluna-filedao-2.0.3.2.jar	terasoluna-filedao-3.3.1.jar	差し替え
terasoluna-ibatis-2.0.5.0.jar	terasoluna-ibatis-3.3.1.jar	差し替え
terasoluna-logger-1.0.0.jar	terasoluna-logger-3.3.1.jar	差し替え
terasoluna-validator-2.0.5.0.jar	terasoluna-validator-3.3.1.jar	差し替え
jakarta-oro-2.0.8.jar	oro-2.0.8.jar ※名称変更のみ	差し替え

「種別」に従って下記の修正を行う。

※「3.3.1 のライブラリ」は terasoluna-batch4j-blank\_3.3.1.zip を展開したフ オルダの lib フォルダ直下に格納されている。

#### 削除:

アプリケーションの lib フォルダ直下から「3.2.0 のライブラリ」に記載した jar ファイルを削除する。

## 差し替え:

アプリケーションの lib フォルダ直下から「3.2.0 のライブラリ」に記載した jar ファイルを削除し、「3.3.1のライブラリ」に記載した jar ファイルを追加する。

## build.xml の修正

②依存ライブラリの変更に従って、build.xml のクラスパスの設定を修正する。 アプリケーション直下の/ant/build.xml を②依存ライブラリの変更で使用した依 存ライブラリ表を参照し、「種別」に従って下記の修正を行う。

#### 削除:

「3.2.0 のライブラリ」に記載した jar ファイルのクラスパス設定を削除する。

## 差し替え:

「3. 2. 0 のライブラリ」に記載した jar ファイルから「3. 3. 1 のライブラリ」に記 載した jar ファイルにクラスパス設定の jar ファイル名を変更する。

#### ●修正例●

#### [3. 2. 0 ∅ build. xml]

```
<!-- クラスパスの設定 -->
cproperty name="classpath.lib" value="
    ${lib.dir}/aspectjweaver-1.6.12.jar;
    ${lib.dir}/cglib-nodep-2.1_3.jar;
    ${lib.dir}/commons-beanutils-1.8.3.jar;
    ${lib.dir}/commons-collections-3.2.jar;
      * * * (中略) * * *
"/>
```

#### [3. 3. 1 Ø build. xml]

```
<!-- クラスパスの設定 -->
cproperty name="classpath.lib" value="
    ${lib.dir}/aspectjweaver-1.7.4.jar;
    ${lib.dir}/commons-beanutils-1.8.3.jar;
    ${lib.dir}/commons-collections-3.2.1.jar;
      * * * (中略) * * *
"/>
```

## ④ classpath.bat の修正

②依存ライブラリの変更に従って、classpath.bat の設定を修正する。 アプリケーション直下の/scripts/classpath.bat を②依存ライブラリの変更で使 用した依存ライブラリ表を参照し、「種別」に従って下記の修正を行う。

#### 削除:

「3.2.0 のライブラリ」に記載した jar ファイルのクラスパス設定を削除する。 差し替え:

「3. 2. 0 のライブラリ」に記載した jar ファイルから「3. 3. 1 のライブラリ」に記 載した jar ファイルにクラスパス設定の jar ファイル名を変更する。

#### ●修正例●

[3. 2. 0 O classpath. bat]

SET CLASSPATH=%OUTPUT\_FOLDER%;%LIB\_PATH%\terasoluna-batch-3.2.0.jar

SET CLASSPATH=%CLASSPATH%;%LIB\_PATH%¥aspectjweaver-1.6.12.jar SET CLASSPATH=%CLASSPATH%;%LIB\_PATH%\( \) cglib-nodep-2.1\_3.jar

SET CLASSPATH=%CLASSPATH%;%LIB\_PATH%¥commons-beanutils-1.8.3.jar

\* \* \* (中略) \* \* \*

#### (3.3.1 Ø classpath.bat)

SET CLASSPATH=%OUTPUT\_FOLDER%;%LIB\_PATH%\terasoluna-batch-3.3.1.jar

SET CLASSPATH=%CLASSPATH%;%LIB\_PATH%¥aspectjweaver-1.7.4.jar SET CLASSPATH=%CLASSPATH%;%LIB\_PATH%¥commons-beanutils-1.8.3.jar \* \* \* (中略) \* \* \*

### ⑤ Spring Bean 定義ファイルの XML スキーマ定義(XSD)からバージョン指定を削除

Spring プロファイル機能など、Spring3.1 またはそれ以降から登場した新しい Bean 定義の記述を使用する場合は、XML スキーマ定義(XSD)のバージョン指定を削 除する。Spring 3.1 またはそれ以降に登場する Bean 定義の記述を使用しない場 合は、XSDのバージョン指定を削除する必要はない。

バージョン指定を削除すると、プロパティファイルによるプレースホルダ置換機 能(<context:property-placeholder>)の動作が変わり、OS の環境変数と同名の プロパティが定義されていると、プロパティの値が OS の環境変数の値で上書きさ れるようになる。そのため、OS の環境変数と同名のプロパティのキーがある場合 は、プロパティのキー(プロパティファイル、Bean 定義ファイルの両方)を変更 する必要がある。

blank プロジェクトで使用しているプロパティのなかでは、データソース定義フ ァイル (dataSource.xml、AdminDataSource.xml) で使用している\${username}の 変更が必要になる。Windows 環境では環境変数の%USERNAME%(ログオンユーザ 名)で上書きされるため、データソース関連のプロパティにすべて接頭辞" jdbc."を追加し、環境変数と衝突しないように変更する。

## ●修正例●

## 【3.2.0 ⊘ AdminDataSource.xml】

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"</pre>
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:util="http://www.springframework.org/schema/util"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/util
    http://www.springframework.org/schema/util/spring-util-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-2.5.xsd">
    <!-- DBCP のデータソースを設定する。
    <context:property-placeholder location="SqlMapAdminConfig/jdbc.properties" />
    <bean id="adminDataSource" destroy-method="close"</pre>
      class="org.apache.commons.dbcp.BasicDataSource">
         cproperty name="driverClassName" value="${driver}" />
         cproperty name="url"
                                          value="${url}" />
         property name="username"
                                           value="${username}" />
                                           value="${password}" />
         property name="password"
                                           value="5" />
         cproperty name="maxActive"
         cproperty name="maxIdle"
                                           value="1" />
                                          value="5000" />
         cproperty name="maxWait"
    </bean>
     * * * (中略) * * *
```

## [3.2.0 ∅ jdbc.properties]

```
driver=org.postgresql.Driver
url=jdbc:postgresql://127.0.0.1:5432/postgres
username=postgres
password=postgres
```

### 【3.3.1 ⊘ AdminDataSource.xml】

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"</p>
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:util="http://www.springframework.org/schema/util"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/util
    http://www.springframework.org/schema/util/spring-util.xsd
    http://www.springframework.org/schema/aop
   http://www.springframework.org/schema/aop/spring-aop.xsd
   http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">
    <!-- DBCP のデータソースを設定する。
    <context:property-placeholder location="SqlMapAdminConfig/jdbc.properties" />
    <bean id="adminDataSource" destroy-method="close"</pre>
      class="org.apache.commons.dbcp.BasicDataSource">
         cproperty name="driverClassName" value="${jdbc.driver}" />
         property name="url"
                                          value="${jdbc.url}" />
         cproperty name="username"
                                           value="${jdbc.username}" />
         property name="password"
                                           value="${jdbc.password}"/>
         cproperty name="maxActive"
                                           value="5" />
                                           value="1" />
         cproperty name="maxIdle"
         cproperty name="maxWait"
                                          value="5000" />
    </bean>
     * * *(中略) * * *
```

#### [3.3.1 Ø jdbc.properties]

```
jdbc.driver=org.postgresql.Driver
jdbc.url=jdbc:postgresql://127.0.0.1:5432/postgres
jdbc.username=postgres
jdbc.password=postgres
```

## バグの修正に対する影響調査

3.3.1で修正されたバグに対して影響調査を実施する。

本書では機能説明書、バグ一覧に記載していた事項について説明する。プロジェ クトで独自の対処を行っている場合は、対処方針を別途検討する必要がある。

機能説明書「BL-06 データベースアクセス機能」の「備考」にて、複数デー タソースの利用時は transactionManager の Bean 定義で transactionSynchronization の値を 2 に設定すると説明していた。本設定は フレームワークの誤った実装が原因で必要だったが、3.3.1 でフレームワー クの実装を修正したため、本設定は不要になる。

ただし、既に複数データソースを利用していて transactionSynchronization の値を2に設定している場合は、本設定を削除する必要はない。

## ●修正例●

#### (3.2.0の dataSource.xml)

```
<!-- トランザクションマネージャの定義 -->
```

<br/><bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager"> cproperty name="dataSource" ref="dataSource" />

- <!-- 複数 DB 接続を行うためにトランザクション同期を行わない設定 -->
- <!-- 複数の DataSource 定義が使用される場合、下のコメントアウトを削除してください。 -->
- <!-- <pre><!-- <pre>cproperty name="transactionSynchronization" value="2"/> -->

</bean>

## [3.3.1 Ø dataSource.xml]

#### <!-- トランザクションマネージャの定義 -->

<br/><bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager"> cproperty name="dataSource" ref="dataSource" />

</bean>

3.2.0 では入力データ取得機能(コレクタ)を使用し、入力チェックエラー が発生した際にスキップするように拡張入力チェックエラーハンドラクラス を実装していた場合、または、データ取得時に例外が発生した際にスキップ するように拡張例外ハンドラクラスを実装していた場合、デッドロックを起 こしてしまっていた。本事象の対処方法として BLogic にパッチを当ててい る場合は、3.3.1への移行に伴いBLogicのパッチを削除する必要がある。 具体的には、入力データ取得機能(コレクタ)を使用しているすべての BLogic に対して、修正例に太字で記載したコードが追加されているか確認し、 BLogic にコードが追加されていた場合は、削除する。

## ●修正例●

## (3. 2. 0 Ø BLogic)

```
Collector<Data> collector = new FileValidateCollector<Data>(...) {
    protected BlockingQueue<DataValueObject> createQueue() {
         super.createQueue();
         return new ArrayBlockingQueueEx<DataValueObject>(this.queueSize) {
              @Override
              public DataValueObject poll() {
                  queueLock.lock();
                  try {
                       DataValueObject elm = super.poll();
                       if (elm != null) {
                           notFull.signal();
                       return elm;
                  } finally {
                       queueLock.unlock();
              }
         };
};
```

#### (3. 3. 1 Ø BLogic)

```
Collector<Data> collector = new FileValidateCollector<Data>(...);
```

非同期バッチ処理において、ジョブの実行結果をデータベースに書き込む際 のエラーメッセージが誤っていた。本事象の対処方法として、terasolunabatch-3.2.0. jar よりも先に通っているクラスパス上に META-INF/logmessages-AL025. properties を配置してメッセージを修正していた場合、 3.3.1 では本メッセージの修正に加え、その他メッセージの追加や修正があ るため、terasoluna-batch-3.3.1. jar に含まれる META-INF/log-messages-AL025. properties との差分をマージする必要がある。

なお、プロジェクトで配置している log-messages-ALO25.properties と terasoluna-batch-3.2.0.jar に含まれる META-INF/log-messages-AL025. properties を比較し、修正箇所がメッセージ ID「EAL025025」のみで あった場合は、差分をマージせずにファイルを削除するだけで良い。

▶ 入力データ取得機能(コレクタ)の Collector インタフェースで公開してい る getCurrent メソッド、getPrevious メソッド、getNext メソッドの仕様が 修正された。そのため、コントロールブレイク機能を用いている処理、およ び、各公開メソッドを使用している処理において、異常データ(DTO に変換 できないデータや、入力チェックエラーとなるデータ)を入力した際の挙動 に問題がないか十分に確認する。

コントロールブレイク機能の ControlBreakChecker クラスは Collector イン タフェースで公開している各メソッドを以下のように使用しているため、影 響を受けている。

- ✓ isPreBreak メソッドを使用した前ブレイク判定では、Collector インタ フェースの getCurrent メソッドを使用して取得する「現在取得している 入力データ」と、Collector インタフェースの getPrevious メソッドを 使用して取得する「1つ前に取得した入力データ」を用いて判定する。
- ✓ isBreak メソッドを使用した後ブレイク判定では、Collector インタフェ ースの getCurrent メソッドを使用して取得する「現在取得している入力 データ」と、Collector インタフェースの getNext メソッドを使用して 取得する「次に取得できる入力データ」を用いて判定する。

この前後のデータを取得する際に例外が発生している場合や、入力チェック 時にエラーが発生した場合の動作が 3.2.0 では未定義であったが、3.3.1 で は以下の通りに定義された。

(機能説明書「AL-042 コントロールブレイク機能」の「備考」より抜粋)

- 例外発生データ検出時の振る舞い
  - ▶ データ入力に成功後、入力チェックエラーにより、Collector#next 実行時に入力チェック例外が発生するケースでは、入力されたデー タを参照してコントロールブレイク判定を行う。
  - データ入力が正常にできておらず、Collector#next 実行時に例外 (データ入力時に発生した例外。FileLineException 等) が発生する ケースでは、コントロールブレイク判定時にも例外(データ入力時 に発生した例外) をスローする。

入力チェックエラー発生時の例を用いて 3.2.0 から 3.3.1 への移行で生じる 前ブレイク判定/後ブレイク判定の動作の違いを図で説明する。また、入力 チェックエラー発生時のハンドリング方法については機能説明書「AL-043 入 力チェック機能」を参照。

●入力チェックエラー発生時に発生した例外をそのままスローする場合

3.2.0 入力チェックエラー発 生データ\*を getPrevious メソッド で取得すると、1 つ前の正常デ ータを取得するため、ブレイクキ ーの値(田中と佐藤)の誤った。 比較を行う。その結果、コントロ ールブレイクが発生する。(不正 な動作)

3.3.1 . 入力チェックエラー発 生データ\*を getPrevious メソッド で取得し、ブレイクキーの値(田 中と田中)を比較する。その結 果、コントロールブレイクが発生 しない。(正常な動作)

データパターン	担当者 (ブレイクキー)	商品
 正常データ	佐藤	ボールペン
入力チェックエラー <sup>一</sup> 発生データ <sup>*</sup>	田中	xxx
現在取得している 入力データ	田中	鉛筆 — /
入力チェックエラー 発生データ <sup>**</sup>	田中	xxx
正常データ	田中	消しゴム

入力チェックエラー発 生データ\*\*を getNext メソッドで 取得すると、null が返却される ため、ブレイクキーの値を比較 できない。その結果、コントロー ルブレイクが発生する。(不正な 動作)

3.3.1 入力チェックエラー発 生データ\*\*を getNext メソッドで 取得し、ブレイクキーの値(田中 と田中)を比較する。その結果、 コントロールブレイクが発生しな い。(正常な動作)

●入力チェックエラー発生時に当該データの取得をスキップする場合

3.2.0 入力チェックエラー発 生データ\*を getPrevious メソッド で 1 つ前の正常データを取得 し、ブレイクキーの値(田中と佐 藤)を比較する。その結果、コン トロールブレイクが発生する。 (正常な動作)

3.3.1 3.2.0 と同じ動作とな る。(変更なし)

	データパターン	担当者 (ブレイクキー)	商品
	正常デ <b>ー</b> タ ー	佐藤	ボールペン
	入力チェックエラー 発生データ <sup>*</sup>	田中	xxx
	現在取得している 入力データ	田中	<b>鉛筆</b> /
	入力チェックエラー 発生データ <sup>**</sup>	田中	xxx
	正常データ	田中	消しゴム

3.2.0 入力チェックエラー発 生データ\*\*を getNext メソッドで 取得すると、null が返却される ため、ブレイクキーの値を比較 できない。その結果、コントロー ルブレイクが発生する。(不正な 動作)

入力チェックエラー発 生データ\*\*を getNext メソッドで 取得すると、次に取得できる正 常データが返却されるため、ブ レイクキーの値(田中と田中)を 比較する。その結果、コントロー ルブレイクが発生しない。(正常 な動作)

3.3.1

3.2.0

### ●入力チェックエラー発生時に入力データの取得を終了する場合

3.2.0 入力チェックエラー発 生データ\*を getPrevious メソッド で取得すると、1 つ前の正常デ ータを取得するため、ブレイクキ 一の値(田中と佐藤)の誤った 比較を行う。その結果、コントロ ールブレイクが発生する。(不正 な動作)

3.3.1 入力チェックエラー発 生データ\*を getPrevious メソッド で取得し、ブレイクキーの値(田 中と田中)を比較する。その結 果、コントロールブレイクが発生 しない。(正常な動作)

データパターン	担当者 (ブレイクキー)	商品
正常データ	佐藤	ボールペン
_入力チェックエラー 発生データ <sup>*</sup>	田中	XXX
現在取得している 入力データ	田中	鉛筆
入力チェックエラー 発生データ <sup>**</sup>	田中	xxx
正常データ	田中	消しゴム

入力チェックエラー発 生データ\*\*を getNext メソッドで 取得すると、null が返却される ため、ブレイクキーの値を比較 できない。その結果、コントロー ルブレイクが発生する。(正常な 動作)

3.3.1 3.2.0 と同じ動作とな る。(変更なし)

以下に例外(データ入力時に発生する FileLineException 等の例外)発生時 の例を用いて 3.2.0 から 3.3.1 への移行で生じる前ブレイク判定/後ブレイ ク判定の動作の違いを図で説明する。また、例外発生時のハンドリング方法 については機能説明書「AL-041 入力データ取得機能」を参照。

## ●例外発生時に発生した例外をそのままスローする場合

3.2.0 例外発生データ\*を getPrevious メソッドで取得する と、以前の正常データが取得さ れるため、ブレイクキーの値(田 中と佐藤)の誤った比較を行う。 その結果、コントロールブレイク が発生しない。(不正な動作)

例外発生データ\*を getPrevious メソッドで取得する 際、例外が発生する。その結 果、コントロールブレイクの判定 が行われない。(正常な動作)

3.3.1

データパターン	担当者 (ブレイクキー)	商品
正常データ	佐藤	ボールペン
例外 <sup>-</sup> 発生データ <sup>*</sup>	田中xxxDTO に変換できないため参照不能	
現在取得している 入力データ	田中	鉛筆
例外 発生データ <sup>**</sup>	田中 DTO に変換できな	xxx いため参照不能
正常データ	田中	消しゴム

3.2.0 例外発生データ\*\*を getNext メソッドで取得すると、 null が返却されるため、例外が 発生しない。また、null との比較 になるため、ブレイクキーの値を 比較できない。その結果、コント ロールブレイクが発生する。(不 正な動作)

<u>例</u>外発生データ\*\*を getNext メソッドで取得する際、 例外が発生する。その結果、コ ントロールブレイクの判定が行 われない。(正常な動作)

3.3.1

3.2.0

3.3.1

3.2.0

消しゴム

### ●例外発生時に当該データの取得をスキップする場合

3.2.0 例外発生データ\*を getPrevious メソッドで 1 つ前の 正常データを取得し、ブレイク キーの値(田中と佐藤)を比較す る。その結果、コントロールブレ イクが発生する。(正常な動作)

データパターン 担当者 商品 (ブレイクキー) 正常データ ボールペン 佐藤 例外 田中 発生データ\* DTO に変換できないため参照不能 現在取得している 田中 鉛筆

例外発生データ\*\*を getNext メソッドで取得すると、 null が返却されるため、例外が 発生しない。ブレイクキーの値 を比較できない。その結果、コ ントロールブレイクが発生する。 (不正な動作)

3.3.1 3.2.0 と同じ動作とな る。(変更なし)

入力データ 例外 田中 XXX発生データ\*\* DTO に変換できないため参照不能

田中

例外発生データ\*\*を getNext メソッドで取得すると、 次に取得できる正常データが返 却されるため、ブレイクキーの値 (田中と田中)を比較する。その 結果、コントロールブレイクが発

生しない。(正常な動作)

## ●例外発生時に入力データの取得を終了する場合

正常データ

3.2.0 例外発生データ\*を getPrevious メソッドで 1 つ前の 正常データを取得し、ブレイク キーの値(田中と佐藤)を比較す る。その結果、コントロールブレ イクが発生する。(不正な動作)

3.3.1 例外発生データ\*を getPrevious メソッドで取得する 際、例外が発生する。その結 果、コントロールブレイクの判定 が行われない。(正常な動作)

データパターン	担当者 (ブレイクキー)	商品
正常データ	佐藤	ボールペン
例外  発生データ <sup>*</sup>	田中 DTO に変換できな	xxx いため参照不能
現在取得している 入力データ	田中	鉛筆
例外 発生データ <sup>**</sup>	田中 DTO に変換できな	xxx いため参照不能
正常データ	田中	消しゴム

入力チェックエラー発 生データ\*\*を getNext メソッドで 取得すると、null が返却される ため、ブレイクキーの値を比較 できない。その結果、コントロー ルブレイクが発生する。(正常な 動作)

3.3.1 3.2.0 と同じ動作とな る。(変更なし)

#### ⑦ 動作確認

アプリケーションを起動させて、問題なく動作することを回帰試験等により確認 すること。