

# JEUS JBatchガイド

JEUS v8.0



Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

## Copyright Notice

Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

45, Jeongjail-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, 13613, South Korea

## Restricted Rights Legend

All TmaxSoft Software (JEUS®) and documents are protected by copyright laws and international convention. TmaxSoft software and documents are made available under the terms of the TmaxSoft License Agreement and this document may only be distributed or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TmaxSoft Co., Ltd.

Nothing in this software document and agreement constitutes a transfer of intellectual property rights regardless of whether or not such rights are registered) or any rights to TmaxSoft trademarks, logos, or any other brand features. This document is for information purposes only. The company assumes no direct or indirect responsibilities for the contents of this document, and does not guarantee that the information contained in this document satisfies certain legal or commercial conditions. The information contained in this document is subject to change without prior notice due to product upgrades or updates. The company assumes no liability for any errors in this document.

このソフトウェア(JEUS®)マニュアルの内容とプログラムは、日本国の著作権法および国際条約によって保護されています。マニュアルの内容とプログラムは、TmaxSoft Co., Ltd.との使用許諾契約書の下でのみ使用することができ、マニュアルは使用許諾契約で許可されている範囲を除いては、配布または複製することができません。TmaxSoftの書面による事前の承諾を得ることなく、このマニュアルの全部または一部を電子的または機械的な方法を問わず、転送、複製、配布したり、または二次的著作物を作成する等の行為を一切禁じます。

このソフトウェアのマニュアルとプログラムの使用許諾契約は、いかなる場合においても、マニュアル及びプログラムと関連する知的財産権(登録の有無を問わず)を譲渡するものと解釈されず、TmaxSoftのブランド、ロゴ、商標等の使用権限を与えるものではありません。マニュアルは、情報を提供する目的でのみ提供しており、これに伴う契約上の直接的ないしは間接的な責任を負わず、マニュアルの内容は法律上もしくは商業的な特定の条件が満たされることを保証しません。マニュアルの内容は、製品のアップグレード及び修正により、その内容が予告なく変更されることがあり、内容上の誤りがないことを保証しません。

## Trademarks

JEUS® is registered trademark of TmaxSoft Co., Ltd.

JEUS®は、TmaxSoft Co., Ltd.の登録商標です。

Java and Solaris are registered trademarks of Oracle Corporation and its subsidiaries and affiliates.

Java、Solarisは、Oracle Corporation及びその子会社、関連会社の登録商標です。

Microsoft, Windows, and Windows NT are registered trademarks or trademarks of Microsoft Corporation.

Microsoft、Windows、Windows NTは、Microsoft Corporationの登録商標または商標です。

HP-UX is a registered trademark of Hewlett Packard Enterprise Company.

HP-UXは、Hewlett Packard Enterprise Companyの登録商標です。

---

AIX is a registered trademark of International Business Machines Corporation.

AIXは、International Business Machines Corporationの登録商標です。

UNIX is a registered trademark of X/Open Company, Ltd.

UNIXは、X/Open Company, Ltd.の登録商標です。

Linux is a registered trademark of Linus Torvalds.

Linuxは、Linus Torvaldsの登録商標です。

Other products and company names are trademarks or registered trademarks of their respective owners.

その他、記載されている会社名、製品名などは、各社の商号、商標または登録商標です。

The names of companies, systems, and products mentioned in this manual may not necessarily be indicated with a trademark symbol (TM, ®).

本マニュアルに記載されている会社名、システム名、製品名などには必ずしも商標表示(TM、®)を付記しておりません。

### **Open Source Software Notice**

Some modules or files of this product are subject to the terms of the following licenses. : APACHE2.0, CDDL1.0, EDL1.0, OPEN SYMPHONY SOFTWARE1.1, TRILEAD-SSH2, Bouncy Castle, BSD, MIT, SIL OPEN FONT1.1

Detailed Information related to the license can be found in the following directory : \${INSTALL\_PATH}/lib/licenses

この製品の一部ファイルまたはモジュールは、APACHE2.0、CDDL1.0、EDL1.0、OPEN SYMPHONY SOFTWARE1.1、TRILEAD-SSH2、Bouncy Castle、BSD、MIT、SIL OPEN FONT1.1のライセンスに準拠します。

### **文書情報**

文書名: JEUS JBatchガイド

発行日: 2016年10月14日

ソフトウェアバージョン: JEUS v8.0

ガイドバージョン: v2.1.1

---



# 目次

このガイドについて .....	xi
<b>第1章 JBatchについて .....</b>	<b>1</b>
1.1. JBatchの基本概念 .....	1
1.2. JBatchの構成要素 .....	2
1.2.1. JobOperator .....	2
1.2.2. Job .....	2
1.2.3. Step .....	4
1.2.4. ItemReader .....	5
1.2.5. ItemProcessor .....	5
1.2.6. ItemWriter .....	6
1.2.7. チェックポイント .....	6
<b>第2章 JBatchのスレッドプールの環境設定 .....</b>	<b>7</b>
2.1. 概要 .....	7
2.2. DDの設定 .....	7
2.2.1. コンポーネントのDDに設定 .....	8
2.2.2. アプリケーションのDDに設定 .....	9
<b>第3章 JBatchの使用例 .....</b>	<b>11</b>
3.1. 概要 .....	11
3.2. jeus-web-dd.xmlの設定 .....	11
3.3. Jobの定義 .....	11
3.4. JBatchServletの例 .....	12
3.5. ItemReaderの例 .....	13
3.6. ItemProcessorの例 .....	15
3.7. ItemWriterの例 .....	16



## 図目次

[図 1.1]	JBatchの構成要素間の関係 .....	11
[図 1.2]	Job、JobInstance、JobParameters、JobExecutionの関係 .....	3
[図 1.3]	JobとStepの関係 .....	4
[図 1.4]	Step処理時のItemReader-ItemProcessor-ItemWriterの動作過程 .....	5
[図 3.1]	JBatchを使用するServletのWarファイルの構造 .....	11





# 例目次

[例 2.1]	コンポーネントのDDIに設定:<<jeus-web-dd.xml>> .....	8
[例 2.2]	コンポーネントのDDIに設定:<<jeus-ejb-dd.xml>> .....	8
[例 2.3]	アプリケーションのDDIに設定:<<jeus-application-dd.xml>> .....	9
[例 3.1]	Jobの定義:<<simple_file_batch.xml>> .....	12
[例 3.2]	JBatchServletの例 .....	12
[例 3.3]	ItemReaderの例 .....	13
[例 3.4]	ItemProcessorの例 .....	15
[例 3.5]	Personオブジェクト .....	15
[例 3.6]	ItemWriterの例 .....	16



# このガイドについて

## 対象読者

JBatchは、JEUS上でBatch Applications for the Java Platform(JSR-352)の仕様を実装した機能です。バッチ処理が必要なワークロード・パターンをみせ、順序制御や並列実行が必要なジョブを処理する際に有効に使用可能です。

本書は、JBatchを利用して、JEUSでバッチ・アプリケーションを開発しようとする開発者を対象にしています。

## 前提知識

JEUSの基本的な使用方法と製品を理解するには、以下のガイドについてあらかじめ熟知することをお勧めします。

- JEUS 紹介ガイド
- JEUS インストール & スタートガイド

本書のすべてのサンプルと環境構成は、UNIXスタイルに準拠します。Microsoft Windows™(以下、Windows)など他の環境で作業を行う場合は、次のような事項を考慮してください。

たとえば、Windowsプラットフォームでは、ディレクトリー区切り子をUNIXスタイルのスラッシュ(/)からWindowsスタイルのバックスラッシュ(\)に変えて使用してください。また、環境変数もWindowsスタイル(%%)に変更して使用してください。

本書で触れているJEUS\_HOMEは、JEUSがインストールされているディレクトリーです。

## 制限事項

本書の内容は、Java標準に準拠して作成されていますが、本書で触れているJava EEやJava仕様については詳しく取り上げていません。関連内容についてはJava関連ドキュメントを参照してください。

# 本書の構成

本書は計3章で構成されています。

- [「第1章 JBatchについて」](#)

JBatchの概念とJBatchの構成要素について説明します。

- [「第2章 JBatchのスレッドプールの環境設定」](#)

JBatchで定義したジョブをJEUSのスレッドプールで実行するために必要な環境設定方法について説明します。

- [「第3章 JBatchの使用例」](#)

JBatchを活用したアプリケーションの作成例について説明します。

## 表記上の規則

表記	意味
<<AaBbCc123>>	プログラム・ソースコードのファイル名
<Ctrl>+C	CtrlキーとCキーを同時に押す
[Button]	GUIのボタン、メニュー名
太字	強調
「」、『』（鍵カッコ）	関連文書、あるいはガイド内の他の章および節の表示
「入力項目」	画面UI上の入力項目
ハイパーリンク	メール・アカウント、Webサイト
>	メニューの実行順
+----	下位ディレクトリー/ファイル有り
----	下位ディレクトリー/ファイル無し
<b>参考</b>	参照/注意事項
<b>注</b>	注意事項
[図 1.1]	図の名前
[表 1.1]	表の名前
AaBbCc123	Javaコード、XMLドキュメント
[ <i>command argument</i> ]	オプション・パラメータ
< xyz >	「<」と「>」の間の内容は実際に使用される特定の名前または値で置き換えられる
	構文の中の相互に排他的な選択項目の選択肢を示す 例) A B: AとBのいずれかを選択
...	パラメータ、値、または他の情報が繰り返される
\${ }	環境変数

## システム要件

	要求事項
プラットフォーム	Solaris 9, 10, 11
	HP-UX 11.x, 11i, 11iV2
	IBM AIX 5L, 6L, AIX 7L
	MS Windows 2008, 2012, Vista, 7, 8
ハードウェア	最小2GB以上、推奨20GBのハードディスク容量
	推奨1GB以上のメモリー容量
JDK	JDK 7, JDK 8

## 関連文書

ガイド	説明
JEUS 紹介ガイド	JEUSサーバーについて全般的に紹介し、JEUSのアーキテクチャーを含む各構成要素について記述しています
JEUS インストール&スタートガイド	JEUSについて紹介し、JEUSのインストールおよび開始方法について記述しています
JEUS サーバガイド	JEUSシステムおよびサーバーの概要とシステムの管理方法について記述しています
JEUS WebAdminガイド	JEUSのWeb管理ツールであるWebAdminを利用したJEUSの設定および制御、モニタリング、クラスタリング、リソースの設定および管理について記述しています

## 参考資料

- JSR-352(Batch Applications for the Java Platform)の仕様書

## お問合せ先

### Korea

TmaxSoft Co., Ltd.  
45, Jeongjail-ro, Bundang-gu,  
Seongnam-si, Gyeonggi-do, 13613  
South Korea  
Tel: +82-31-8018-1000  
Fax: +82-31-8018-1115  
Email: [info@tmax.co.kr](mailto:info@tmax.co.kr)  
Web (Korean): <http://www.tmaxsoft.com>  
TechNet: <http://technet.tmaxsoft.com>

### USA

TmaxSoft Inc.  
101 North Wacker Drive, Suite 2014,  
Chicago, IL 60606  
U.S.A  
Tel: +1-312-525-8330  
Email: [info@tmaxsoft.com](mailto:info@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/us\\_en/home](http://www.tmaxsoft.com/us_en/home)

### Japan

TmaxSoft Japan Co., Ltd.  
5F Sanko Bldg, 3-12-16 Mita, Minato-Ku, Tokyo, 108-0073  
Japan  
Tel: +81-3-5765-2550  
Fax: +81-3-5765-2567  
Email: [info@tmaxsoft.co.jp](mailto:info@tmaxsoft.co.jp)  
Web (Japanese): <http://www.tmaxsoft.co.jp>

## China

Beijing TmaxSoft System Software Co., Ltd.  
Room103, No.2 Huizhong Building, Seven Street Shangdi,  
Haidian District, Beijing, 100085  
P.R.China  
Tel: +86-10-6298-8827  
Email: [info@tmaxsoft.com.cn](mailto:info@tmaxsoft.com.cn)  
Web (Chinese): [http://www.tmaxsoft.com/cn\\_en/home\\_cn\\_en](http://www.tmaxsoft.com/cn_en/home_cn_en)

## Brazil

Tmax Brasil Sistemas e Serviços Ltda.  
Av. Copacabana, 177, sala 32~35 Empresarial 18 do Fortel  
Alphaville Barueri, Sao Paulo, 06472-001  
Brazil  
Tel: +55-11-4191-3100  
Fax: +55(11) 4191-3705 (extension#112)  
Email: [info.bra@tmaxsoft.com](mailto:info.bra@tmaxsoft.com)  
Web (Portuguese): [http://www.tmaxsoft.com/br\\_en/home\\_br\\_en](http://www.tmaxsoft.com/br_en/home_br_en)

## Russia

Tmax Rus L.L.C.  
Leninsky prospekt, 113/1 (Park Place Moscow),  
Office 318e, Moscow, 117198  
Russia  
Tel: +7(495)970-01-35  
Email: [info.rus@tmaxsoft.com](mailto:info.rus@tmaxsoft.com)  
Web (Russian): [http://www.tmaxsoft.com/ru\\_ru/home\\_ru\\_ru](http://www.tmaxsoft.com/ru_ru/home_ru_ru)



## Singapore

Tmax Singapore Pte. Ltd.  
430 Lorong 6, Toa Payoh #10-02,  
OrangeTee Building, 319402  
Singapore  
Tel: +65-6259-7223  
Fax: +65-6258-7112  
Email: [info.sg@tmaxsoft.com](mailto:info.sg@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/sg\\_en/home\\_sg\\_en](http://www.tmaxsoft.com/sg_en/home_sg_en)

## United Kingdom

TmaxSoft UK Ltd.  
215 Knyvett House, Watermans Business Park,  
The Causeway, Staines TW18 3BAB  
United Kingdom  
Tel: +44-1784-895005  
Email: [info.uk@tmaxsoft.com](mailto:info.uk@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/gb\\_en/home\\_gb\\_en](http://www.tmaxsoft.com/gb_en/home_gb_en)

## Canada

TmaxSoft Canada, Inc.  
2425 Matheson Blvd East, 8th floor,  
Unit 824 Mississauga, ON, L4W 5K4  
Canada  
Tel: +1-905-361-2888  
Email: [info.canada@tmaxsoft.com](mailto:info.canada@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/ca\\_en/home\\_ca\\_en](http://www.tmaxsoft.com/ca_en/home_ca_en)

## Australia

TmaxSoft Proprietary Limited  
L32, 101 Miller Street, North Sydney 2060  
Australia  
Tel: +91-9845-330-704  
Email: [info.aus@tmaxsoft.com](mailto:info.aus@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/au\\_en/home\\_au\\_en](http://www.tmaxsoft.com/au_en/home_au_en)

## India

TmaxSoft Technologies Private Limited  
Sobha Alexander Plaza, 3rd Floor,  
16/2 Commissariat Road, Bangalore-560025  
India  
Tel: +91-9845-330-704  
Email: [info.india@tmaxsoft.com](mailto:info.india@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/in\\_en/home\\_in\\_en](http://www.tmaxsoft.com/in_en/home_in_en)

## Turkey

TmaxSoft Co., Ltd. Turkey Liaison Office  
Windowist Tower. Eski Buyukdere Cad. No:26,  
Maslak 34467 Istanbul  
Turkey  
Tel: +90-544-553-6045  
Email: [cslee@tmaxsoft.com](mailto:cslee@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/tr\\_en/home\\_tr\\_en](http://www.tmaxsoft.com/tr_en/home_tr_en)

# 第1章 JBatchについて

本章では、JBatchの基本概念とJBatchの構成要素について説明します。

## 1.1. JBatchの基本概念

バッチ処理は、広く使われるワークロード・パターンの一つです。特徴としては、バルクで処理し、相互互換には対応していないほか、バックグラウンドで実行される点が挙げられます。通常、実行時間が長く、計算量の多いジョブに適合しており、順序制御または並列実行が必要な、どちらのジョブでも使用可能です。バッチ処理は、アドホック、スケジュール、オンデマンドなど、多様な手法で開始することができます。

JBatchは、アプリケーションが必要とするロギング、チェックポイント、並列実行などの要件を満たしています。また、バッチのワークロードが必要とする運用制御機能のほか、バッチジョブのスタートや、ストップ、リスタートのような相互作業機能も提供しています。

JBatchはBatch Applications for the Java Platform(JSR-352仕様)のRIをベースに動作します。また、スレッドプールもユーザーが直接設定できるようにしています。

---

### 参考

Batch Applications for the Java Platformの詳細については、JSR-352の仕様をご参照ください。

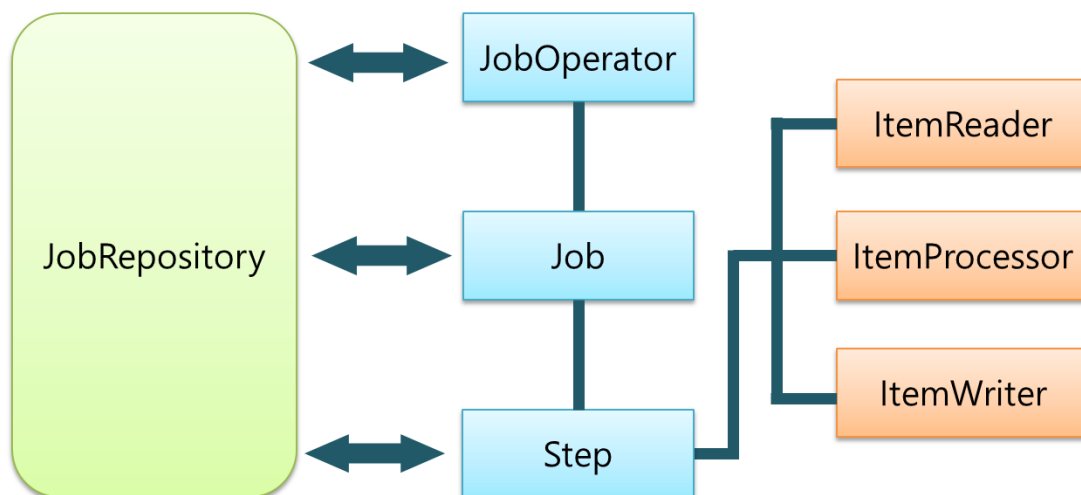
---

## 1.2. JBatchの構成要素

本節では、JBatchのバッチジョブの構成要素について説明します。

以下の図は、JBatchの構成要素間の関係を簡単に示しています。

[図 1.1] JBatchの構成要素間の関係



ユーザーアプリケーションはJobOperator経由でJobを制御(実行および中止)します。Jobは実行しようとするJobの明細、つまり、Jobの下位項目であるStepを保持しています。

1つのJobは1つ以上のStepで構成可能です。一方、StepはItemReader、ItemProcessor、ItemWriterの3つの処理モジュールで構成されています。

現在ランニング中のプロセスのメタデータは、JBatchの実装内のJobRepositoryに保存されます。

### 1.2.1. JobOperator

JobOperatorは、Job処理を照会および制御します。また、ユーザーアプリケーション内のバッチジョブを実行するインターフェースの役割もします。一例としては、バッチジョブのスタート、リスタート、ストップを制御したり、StepExecutionを呼び込むコマンドを提供したりします。

### 1.2.2. Job

Jobは、バッチプロセス全体をカプセル化したデータのまとまり(エンティティ)で、1つ以上のStepで構成されています。すべてのStepをグローバルに管理する環境設定をすることもできます。Jobの環境設定では、Jobの名前、Stepの定義と順序、Jobのリスタート機能の使用有無などを設定することができます。

- JobInstance

JobInstanceとは論理的なJobの実行を意味します。

毎日特定の時間に動作するJob(例: EndOfDayジョブ)を定義したと仮定します。そうすると、このJobは「毎日特定の時間に動作する(EndOfDayジョブ)」との論理的なJobInstanceを一つ持つことになります。もし、1月1日にこのJobを実行中に失敗が発生したら、このJobは失敗した1月1日のJobInstanceを翌日に再実行します。また、1月2日のJobInstanceを新しく作成し、そちらも実行します。したがって、JobInstanceはJobを複数回実行することができます。一方、実際に同じJobの実行を複数回トライする一つ一つの行為をJobExecutionといいます。JobExecutionは、JobInstanceが実行されるたびに新しく作成されます。

新しいJobInstanceを使用するというのは、Jobを最初から実行することを意味します。一方、既に定義済みのJobInstanceを使用するというのは、チェックポイントを使い、処理が必要な時点から処理を始めることを意味します。

JobInstanceそのものは、データについては知りません。データについての責任は、データをItem単位で読み込んでロードするItemReaderの実装にあります。

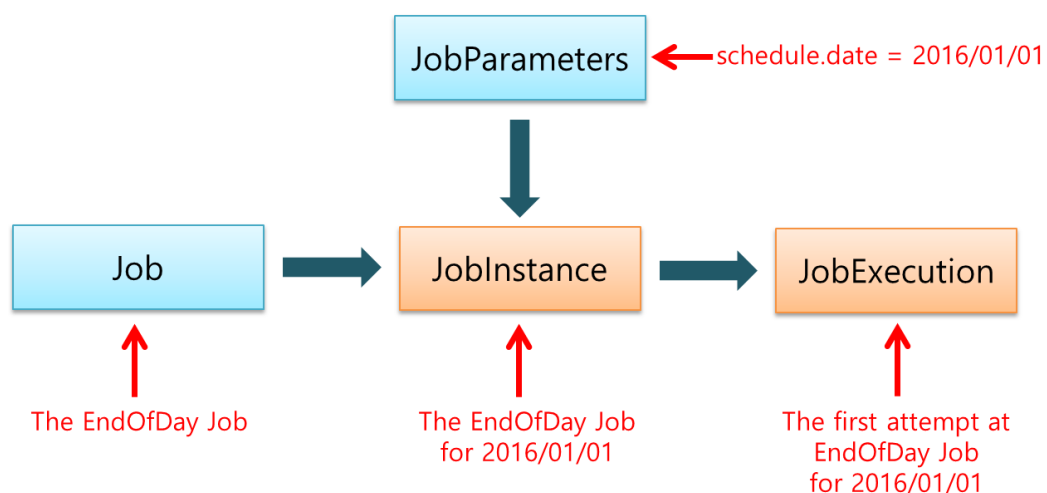
- JobParameters

あるJobInstanceを他のJobInstanceと区別するには、JobParametersが必要です。もちろん、複数のJobInstanceに同じJobParametersを使用することもできます。具体的に、JobParametersとは、Java SEより提供されるjava.util.Propertiesクラスのことです。

たとえば、1月1日のJobInstanceはJobParametersとして「schedule.date = 2016/01/01」を持っています。そして、1月2日のJobInstanceはJobParametersとして「schedule.date = 2016/01/02」を持っています。

つまり、Jobは複数のJobInstanceを持つことができ、それぞれのJobInstanceは各自のJobParametersで区別されます。JobExecutionの場合、一つのJobInstanceの下で複数回実行することができます。

[図 1.2] Job、JobInstance、JobParameters、JobExecutionの関係



- JobExecution

JobExecutionとは、Jobの実行を一回トライすることを意味します。トライの結果は、成功の場合も、そして失敗の場合もあります。JobInstanceは、JobExecutionのインスタンスが実行に成功した場合のみ成功と

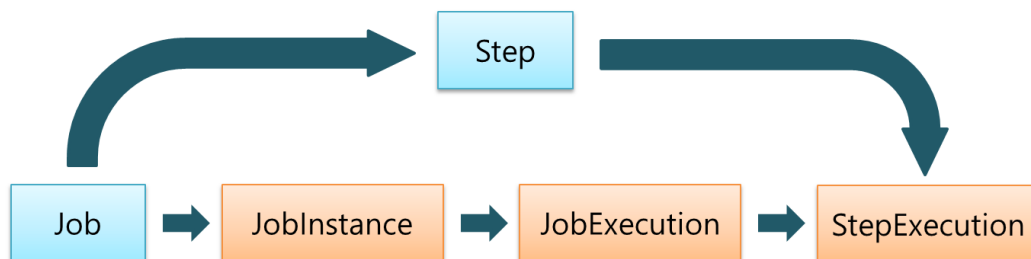
みなします。たとえば、「毎日特定の時間に動作するJob (EndOfDayジョブ)」を定義した場合、そのJobの実行をトライするインスタンスはJobExecutionです。もし、1月1日に実行したJobExecutionが失敗したら、翌日に実行をトライするのは、同じ1月1日に作成されたJobInstanceが再度定義した新しいJobExecutionが同じJobの実行をトライするのです。

### 1.2.3. Step

Stepは、バッチジョブの順次的かつ独立的な各段階をそれぞれカプセル化したドメイン・オブジェクトです。各Jobは1つ以上のStepで構成されています。Stepは、実際のバッチ処理を定義および制御するために必要なすべての情報を含めています。Stepの簡単な例としては、ファイルのデータをデータベースにロードすることが挙げられます。Jobと同様、StepもJobExecutionに相当するStepExecutionを保持します。

以下の図は、JobとStepの関係を示しています。JobとStepは1:Nの関係にあります。各Stepは状況によって1つ以上のStepExecutionを作成して実行することができます。JobExecutionとStepExecutionの関係もやはり1:Nであり、1つのJobが状況に合わせて複数のStepExecutionを作成して実行させることができます。

[図 1.3] JobとStepの関係



- StepExecution

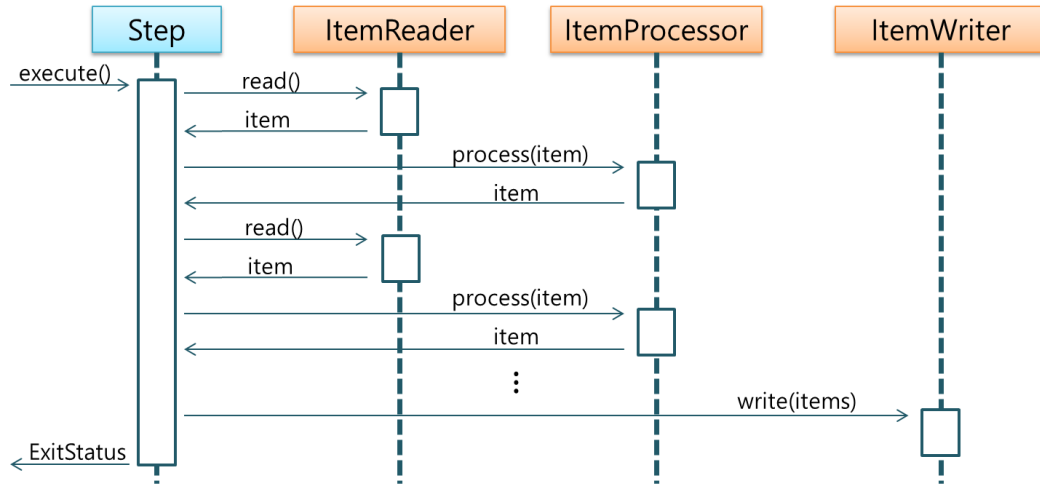
StepExecutionとは、Stepの実行を一回トライすることを意味します。Stepが実行されるたびにStepExecutionが作成されます。StepにはChunkとBatchletがありますが、相互排他的な関係にあり、同時に使用することはできません。

- Chunk

Chunk方式とは、Itemを基盤として処理をする方式のことです。つまり、複数のItemを1つずつ読み込んで一定の処理を行い、Chunkを作成します。作成されたChunkは処理周期内に出力されます。Chunk方式のStepは、ItemReader、ItemProcessor、ItemWriterをそれぞれ1つずつ持っています。Chunkを出力するItemWriterの段階が終わると、チェック・ポイント情報を書き出した後、そのトランザクションはコミットされます。その後、新しいStepが実行されます。

以下の図は、ItemのChunkをItemReader-ItemProcessor-ItemWriterのパターンで処理する過程を簡単に示しています。

**[図 1.4] Step処理時のItemReader-ItemProcessor-ItemWriterの動作過程**



Stepがスタートすると、ItemReaderが入力データをItem単位で読み込んでからItemProcessorに渡します。ItemProcessorは各Itemを処理します。この過程を繰り返しつつ、処理が終わった複数のItemをまとめてItemWriterに渡します。

- Batchlet

BatchletはStepレベルでシンプルに実装でき、ItemをベースとしないJobに有効に使用可能です。一例としては、他のサーバーへのファイルの転送や特定コマンドの実行などが挙げられます。

## 1.2.4. ItemReader

ItemReaderは、1つのStepを構成する複数のItemを読み込むインターフェースです。読み込んだItemはItemProcessorに1つずつ渡します。Itemの簡単な例として、1個のファイル内の1行や1個のデータベース内の1行などが挙げられます。つまり、Itemは1行で定義して渡せられます。

後ほど説明するチェックポイント機能は、ItemReaderが使用します。読み込みに成功した直近の位置をチェックポイントで記録しておけば、エラーが発生したり、リスタートが必要だった場合に、そのチェックポイント時点からリスタートすることができます。上記のファイル内のレコードの場合、最後に読み込んだ行数をチェックポイントで記録しておくことができます。

## 1.2.5. ItemProcessor

ItemProcessorは、Item処理を実際に行うインターフェースです。ItemProcessorは、ItemReaderが1つずつ読み込んだItemを処理します。この処理を複数回繰り返した後、ItemWriterを呼び出して処理結果を書き込みます。

## 1.2.6. ItemWriter

ItemWriterは、Itemの固まりであるChunkを出力するインターフェースです。ItemWriterは現在まで渡されたItemの情報だけを保持し、次の入力情報については知りません。ItemProcessor経由の複数のItemがリストの形態で渡されます。

## 1.2.7. チェックポイント

長時間の処理時間を必要とする大量のデータをバッチ処理するには、一般にチェックポイントとリスタート機能が必要です。

チェックポイントとは、StepExecutionの実行状態を周期的にブックマークしておき、リスタートが必要な際に直近のチェックポイント時点から実行できるようにする機能です。チェックポイントはロックをしてから行われるので、この機能を行過ぎて使用すると、全体システムの性能に悪影響を与えかねません。したがって、チェックポイントの周期を適切に設定するのは、性能チューニングの重要なポイントです。



# 第2章 JBatchのスレッドプールの環境設定

本章では、JBatchで定義したジョブをJEUSのスレッドプールで実行するために必要な環境設定方法について説明します。

## 2.1. 概要

スレッドプールは大きく2箇所に設定することができます。一つは、WebやEJBのDDに設定することであり、もう一つは、アプリケーションのDDに設定することです。

通常は、batch-thread-poolを、Webだけに(またはEJBだけに)設定するか、アプリケーションだけに設定して、優先順位に関係なく、batch-thread-poolに記述されている情報をベースに動作します。

---

**参考**

JEUSは、WebのDDとアプリケーションのDDの両方にスレッドプールの設定が存在する場合(または、EJBのDDとアプリケーションのDDの両方が存在する場合)、WebのDD(またはEJBのDD)のスレッドプールの設定をベースにして動作します。

---

## 2.2. DDの設定

DDに別途でbatch-thread-poolに対する記述をしていない場合は、デフォルト値で設定されているスレッドプールよりジョブが実行されます。

以下は、各設定項目についての説明です。

項目	説明
min	スレッドプールで管理するスレッド数の最小値を指定します  (デフォルト値:0)
max	スレッドプールで管理するスレッド数の最大値を指定します  (デフォルト値:0)
keep-alive-time	スレッドプールで管理するスレッド数がMax以下でMin以上である場合、設定時間内に何の作業も行っていないスレッドがあれば、そのスレッドは自動でスレッドプールから削除されます。

項目	説明
	ただし、スレッドの数が0であれば削除されません(デフォルト値:60000、単位:ミリ秒、ms)
queue-size	スレッドプールで定義したジョブが保存されるキューのサイズを指定します (デフォルト値:4096)

DDで別途でスレッドプールのチューニングを行う場合は、DD間の優先順位にしたがってスレッドプールが設定されます。本節では、各DDの設定方法について説明します。

## 2.2.1. コンポーネントのDDに設定

コンポーネントとはServletやEJBなどのことです。スレッドプールの設定間に競合が発生すれば、最も優先順位の高いのがこのコンポーネントです。コンポーネントはjeus-web-dd.xmlとjeus-ejb-dd.xmlに設定します。

以下は、/WEB-INF/jeus-web-dd.xmlにbatch-thread-poolを設定した例です。

### [例 2.1] コンポーネントのDDに設定:<<jeus-web-dd.xml>>

```
<jeus-web-dd xmlns="http://www.tmaxsoft.com/xml/ns/jeus" version="8.0">
  <batch-thread-pool>
    <min>10</min>
    <max>20</max>
    <keep-alive-time>20</keep-alive-time>
    <queue-size>4096</queue-size>
  </batch-thread-pool>
</jeus-web-dd>
```

以下は、/META-INF/jeus-ejb-dd.xmlにbatch-thread-poolを設定した例です。

### [例 2.2] コンポーネントのDDに設定:<<jeus-ejb-dd.xml>>

```
<jeus-ejb-dd xmlns="http://www.tmaxsoft.com/xml/ns/jeus" version="8.0">
  <batch-thread-pool>
    <min>10</min>
    <max>20</max>
    <keep-alive-time>20</keep-alive-time>
    <queue-size>4096</queue-size>
  </batch-thread-pool>
</jeus-ejb-dd>
```

## 2.2.2. アプリケーションのDDに設定

以下は、/META-INF/jeus-application-dd.xmlにbatch-thread-poolを設定した例です。

### [例 2.3] アプリケーションのDDに設定:<<jeus-application-dd.xml>>

```
<application xmlns="http://www.tmaxsoft.com/xml/ns/jeus" version="8.0">
  <batch-thread-pool>
    <min>10</min>
    <max>20</max>
    <keep-alive-time>20</keep-alive-time>
    <queue-size>4096</queue-size>
  </batch-thread-pool>
</application>
```



## 第3章 JBatchの使用例

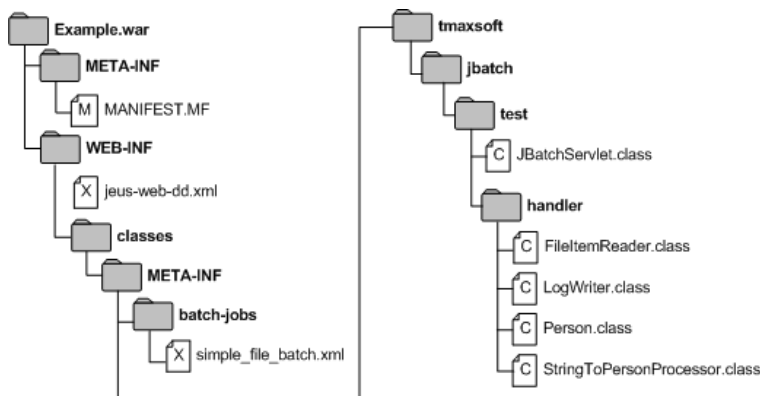
本章では、JBatchを活用したアプリケーションの作成例について説明します。

### 3.1. 概要

ここでは、JEUS上でServletの要求に対するバッチジョブを実行し、その実行結果をクライアントに回答する簡単なアプリケーションを紹介します。

以下は、JBatchを使用するServletのWarファイルの構造です。

[図 3.1] JBatchを使用するServletのWarファイルの構造



### 3.2. jeus-web-dd.xmlの設定

Jobを実行させるスレッドプールを設定します。jeus-web-dd.xmlにbatch-thread-poolの記述がない場合、デフォルト値でスレッドプールが作成されます。詳細については、「[2.2.1. コンポーネントのDDに設定](#)」を参照してください。

### 3.3. Jobの定義

XMLファイルに、Job、Step、ItemReader、ItemWriter、ItemProcessorのJob情報を設定します。このXMLファイルは、META-INF/batch-jobs/ディレクトリー配下に置きます。

---

#### 注

WarファイルのMETA-INFは、構造上の理由から、/WEB-INF/classesディレクトリー配下に置かれないと動作しません。

---

以下は、/WEB-INF/classes/META-INF/batch-jobs/simple\_file\_batch.xmlファイルの設定例です。

**[例 3.1] Jobの定義:<<simple\_file\_batch.xml>>**

```
<job id="demo" xmlns="http://xmlns.jcp.org/xml/ns/javaee" version="1.0">
  <step id="step1">
    <chunk>
      <reader ref="tmaxsoft.bhkim.test.filebatch.FileItemReader">
        <properties>
          <property name="filePath" value="#{jobParameters['filePath']}" />
        </properties>
      </reader>
      <processor ref="tmaxsoft.bhkim.test.filebatch.StringToPersonProcessor" />
      <writer ref="tmaxsoft.bhkim.test.filebatch.LogWriter" />
    </chunk>
  </step>
</job>
```

## 3.4. JBatchServletの例

Jobを設定したXMLをJobOperatorに渡します。Jobの実行時に必要なパラメーターを定義し、それを一緒にJobOperatorに渡すこともできます。

以下はJBatchServletの例です。filePathをキーにするプロパティに、処理しようとするデータを保持しているpersonList.txtファイルのパスを渡す例です。

**[例 3.2] JBatchServletの例**

```
@WebServlet("/JBatchServlet")
public class JBatchServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private final String FILE_PATH = "META-INF/task/personList.txt";

    public JBatchServlet() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        Properties jobParams = new Properties();
        URL personListURL =
            Thread.currentThread().getContextClassLoader().getResource(FILE_PATH);
        jobParams.setProperty("filePath", personListURL.getPath());

        JobOperator operator = BatchRuntime.getJobOperator();
        // xml file name without format
        long id = operator.start("simple_file_batch", jobParams);

        waitForEnd(operator, id);
    }
}
```

```

        response.getWriter().println("File Batch is successfully precessed.");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);
    }

    public static void waitForEnd(final JobOperator jobOperator, final long id) {
        final Collection<BatchStatus> endStatuses = Arrays.asList(BatchStatus.COMPLETED,
BatchStatus.FAILED);
        do {
            try {
                Thread.sleep(100);
            } catch (final InterruptedException e) {
                return;
            }
        } while
(!endStatuses.contains(jobOperator.getJobExecution(id).getBatchStatus()));
    }
}

```

FILE\_PATHで定義しているファイルの内容は以下のとおりです。

```

Name1,27,Computer Science
Name2,22,English Education
Name3,23,Electronic Engineering

```

## 3.5. ItemReaderの例

ItemReaderを実装するためには、4つの関数のopen、close、readItem、checkpointInfoの実装が必要です。ItemReaderでは、Item単位でファイルにアクセスして処理します。この例では、Item単位がファイルの1行に該当します。

open関数は、媒介変数としてSerializableを渡します。これがチェックポイントを確認する機能です。異常終了してリスタートする場合に、最後に記録されたチェックポイントから実行できるようにします。

以下はItemReaderのFileItemReaderの例です。readItemをするたびにrecordNumberを記録しておき、Jobが異常終了してリスタートされる際に、そのrecordNumberからリスタートできるようにする作業をopen関数が実行する例です。

### [例 3.3] ItemReaderの例

```

public class FileItemReader implements ItemReader {

    @Inject
    @BatchProperty

```

```

private String filePath; // JobProperties in xml setting is injected to the field

private BufferedReader reader = null;

private int recordNumber;

@Override
public void open(Serializable checkpoint) throws Exception {
    if (filePath == null)
        throw new RuntimeException("Can't find any input");

    final File file = new File(filePath);
    if (!file.exists())
        throw new RuntimeException("A file '" + filePath + "' doesn't exist");

    reader = new BufferedReader(new FileReader(file));

    if (checkpoint != null) {
        assert (checkpoint instanceof Integer);
        recordNumber = (Integer) checkpoint;

        // Pass over latest checkpoint record
        for (int i = 1; i < recordNumber; i++)
            reader.readLine();
    }
}

@Override
public void close() throws Exception {
    if (reader != null)
        reader.close();
}

@Override
public Object readItem() throws Exception {
    Object item = reader.readLine();

    // checkpoint line update
    recordNumber++;
    return item;
}

@Override
public Serializable checkpointInfo() throws Exception {
    return recordNumber;
}

```



```
}
```

## 3.6. ItemProcessorの例

ItemProcessorはItemReaderから読み込んだデータを処理します。

以下はItemProcessorのStringToPersonProcessorの例です。読み込んだファイルの行の文字列がItemパラメータとして渡された後、その文字列を処理してPersonオブジェクトとして保存します。その後、特定周期ごとに処理済みの情報をItemWriterに渡します。

### [例 3.4] ItemProcessorの例

```
public class StringToPersonProcessor implements ItemProcessor {

    @Override
    public Object processItem(Object item) throws Exception {
        final String[] line = String.class.cast(item).split(",");

        if (line == null || line.length != 3)
            return null;

        // name, age, department
        return new Person(line[0], Integer.parseInt(line[1]), line[2]);
    }
}
```

Processor内で使用するPersonオブジェクトは、以下のように簡単に定義できます。

### [例 3.5] Personオブジェクト

```
public class Person {
    private String name;
    private int age;
    private String department;

    public Person(String name, int age, String department) {
        this.name = name;
        this.age = age;
        this.department = department;
    }

    @Override
    public String toString() {
        return name + "," + age + "," + department;
    }
}
```

## 3.7. ItemWriterの例

ItemWriterを実装するためには、4つの関数のopen、close、writeItems、checkpointInfoの実装が必要です。writeItemで処理する必要のあるItemがバルク(リスト形態)で渡されます。

以下はItemWriterのLogWriterの例です。渡されたItemリストに対し、内部で定義したloggerに出力する例です。

### [例 3.6] ItemWriterの例

```
public class LogWriter implements ItemWriter {
    private static final Logger logger =
        Logger.getLogger(LogWriter.class.getSimpleName());
    int writeRecordNumber; //

    @Override
    public void open(Serializable checkpoint) throws Exception {
    }

    @Override
    public void close() throws Exception {
    }

    @Override
    public void writeItems(List<Object> items) throws Exception {
        // simply print passed item to logger
        for (Object o: items) {
            logger.info("writeItems > " + o.toString());
        }
    }

    @Override
    public Serializable checkpointInfo() throws Exception {
        return null;
    }
}
```