

JEUS スケジューラガイド

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 スケジューラガイド

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
第1章 はじめに	1
1.1. 概要	1
1.2. スケジューラー・コンポーネント	1
1.3. スケジューラー・サーバー	2
第2章 スケジューラーのプログラミング	5
2.1. 概要	5
2.2. JEUS Schedulerクラス	6
2.3. ジョブの定義	7
2.3.1. ScheduleListenerインターフェースの実装	8
2.3.2. Scheduleクラスの継承	8
2.3.3. RemoteScheduleクラスの継承	9
2.4. Schedulerオブジェクトの取得	10
2.4.1. ローカル環境でSchedulerオブジェクトを取得	11
2.4.2. リモート環境でSchedulerオブジェクトを取得	11
2.5. ジョブの登録	12
2.5.1. 一回限りのジョブの登録	12
2.5.2. 繰り返されるジョブの登録	12
2.5.3. Scheduleオブジェクトの登録	14
2.6. ジョブの制御	15
2.7. スケジューラーの使用	15
2.7.1. スタンドアロン環境での使用	15
2.7.2. JEUSサーバーでの使用	17
2.7.3. Java EEコンポーネントでの使用	19
2.8. Job-listの使用	21
第3章 スケジューラーの設定	23
3.1. 概要	23
3.2. Job-listの設定	23
3.3. サーバーのスケジューラー設定	25
3.4. スレッドプールの設定	32
3.4.1. 共用スレッドプール	32
3.4.2. 専用スレッドプール	35
3.5. クライアント・コンテナの設定	39
索引	41

図目次

[図 1.1]	JEUSでのスケジューラー・コンポーネントの構成	2
[図 1.2]	スケジューラー・サーバーのタイプ別実行方式	2
[図 2.1]	スケジューラーAPIクラス	6
[図 3.1]	WebAdminサーバー・リスト画面	25
[図 3.2]	WebAdminのスケジューラー設定画面	26
[図 3.3]	WebAdminのスケジューラー設定 - ロック設定	27
[図 3.4]	WebAdminのスケジューラー設定 - スケジューラー使用の設定	28
[図 3.5]	WebAdminのスケジューラー設定 - ジョブの追加	29
[図 3.6]	WebAdminのスケジューラー設定 - ジョブの保存	29
[図 3.7]	WebAdminのスケジューラー設定 - ジョブ追加実行の確認	30
[図 3.8]	WebAdminのスケジューラー設定 - 追加されたジョブの反映	31
[図 3.9]	スレッドプールの設定 - 共用スレッドプールの設定(1)	33
[図 3.10]	スレッドプールの設定 - 共用スレッドプールの設定(2)	34
[図 3.11]	スレッドプールの設定 - 専用スレッドプールの設定(1)	36
[図 3.12]	スレッドプールの設定 - 専用スレッドプールの設定(2)	37

例目次

[例 2.1]	Taskオブジェクトの例	8
[例 2.2]	Scheduleオブジェクトの例	9
[例 2.3]	RemoteScheduleオブジェクトの例	10
[例 2.4]	繰り返しジョブの登録	14
[例 2.5]	Scheduleオブジェクトの登録	14
[例 2.6]	ScheduleController.cancelメソッドの使用	15
[例 2.7]	スタンドアロン・クライアントでのスケジューラーの使用	16
[例 2.8]	リモート・クライアントでのスケジューラーの使用	18
[例 2.9]	EJBでのJEUSスケジューラーの使用	20
[例 3.1]	jeusadminによるスレッドプール設定	35
[例 3.2]	jeusadminによるスケジューラー・サービスの共用スレッドプールの設定	38
[例 3.3]	クライアント・コンテナの設定 : <<jeus-client-dd.xml>>	39
[例 3.4]	Job-list設定 : <<jeus-client-dd.xml>>	40

このガイドについて

対象読者

本書は、JEUS[®](以下、JEUS)システムを全般的に理解し、JEUSシステムが提供する各種の機能および特性を習得するための基本ガイドです。本書は、JEUSスケジューラー機能を使用する管理者および開発者を対象としています。

前提知識

本書を理解するためには、以下の事項をあらかじめ熟知している必要があります。

- RMIおよびJNDIについての知識
- JEUSの基本概念についての知識
- Javaについての基本知識

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。

実務での具体的な使用方法や管理および運用に関する事項は、各製品ガイド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 はじめに](#)」

JEUSスケジューラーの機能と使用可能な環境、実行方法について説明します。

- 「[第2章 スケジューラーのプログラミング](#)」

JEUSスケジューラーのプログラミングに必要な基本知識とJEUSスケジューラーの使用方法について説明します。

- 「[第3章 スケジューラーの設定](#)」

JEUSスケジューラーの設定方法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
<div>参考</div>	参照/注意事項
<div>注</div>	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 アプリケーションクライアントガイド	Java EEクライアントとJEUS間の相互運用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

参考文献

- Javadoc JEUS API

JEUS_HOME/docs/api/jeus-api/index.html

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
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
Web (English): 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
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
Web (Chinese): 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
Web (Portuguese): 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
Web (Russian): 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
Web (English): 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
Web (English): 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
Web (English): 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
Web (English): 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
Web (English): 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
Web (English): http://www.tmaxsoft.com/tr_en/home_tr_en

第1章 はじめに

本章では、JEUSスケジューラーの機能と使用可能な環境、実行方法について説明します。

1.1. 概要

JEUSスケジューラーは、決められた時間に実行される作業や、繰り返し実行される作業をスケジューリングする機能を提供します。

スケジューラーはJEUSの拡張機能であり、決められた時間、あるいは周期的に作業を実行する必要がある場合に使用できます。EJBではタイマー・サービス(Timer Service)を提供しますが、JEUSスケジューラーはEJB環境でなくても似た目的で 사용할 ことができます。たとえば、周期的に一時ファイルを削除したり、データベース・コネクションをチェックしたりするなどのシステム管理に使用できます。

Java EE環境でタイマー・サービスを使用する場合、Java SEタイマー(`java.util.Timer`)を直接使うことができないため、EJBタイマー・サービスを使うか、JEUSが提供するJEUSスケジューラーを使用しなければなりません。EJBタイマー・サービスがEJB環境でしか使えないのに対し、JEUSスケジューラーはJava EE環境でも使うことができると共に、一般のJava SEアプリケーションでも使用することができます。

JEUSスケジューラーはJava SEタイマー(`java.util.Timer`)に似ているため、Java SEタイマーに使い慣れていれば、JEUSスケジューラーを簡単に使用できます。一方、JEUSスケジューラーはJava SEタイマーにはない、実行終了時点(end time)と最大実行回数(max count)を設定する機能も提供しています。

JEUSスケジューラーは、以下のような多様な環境で使用することができます。

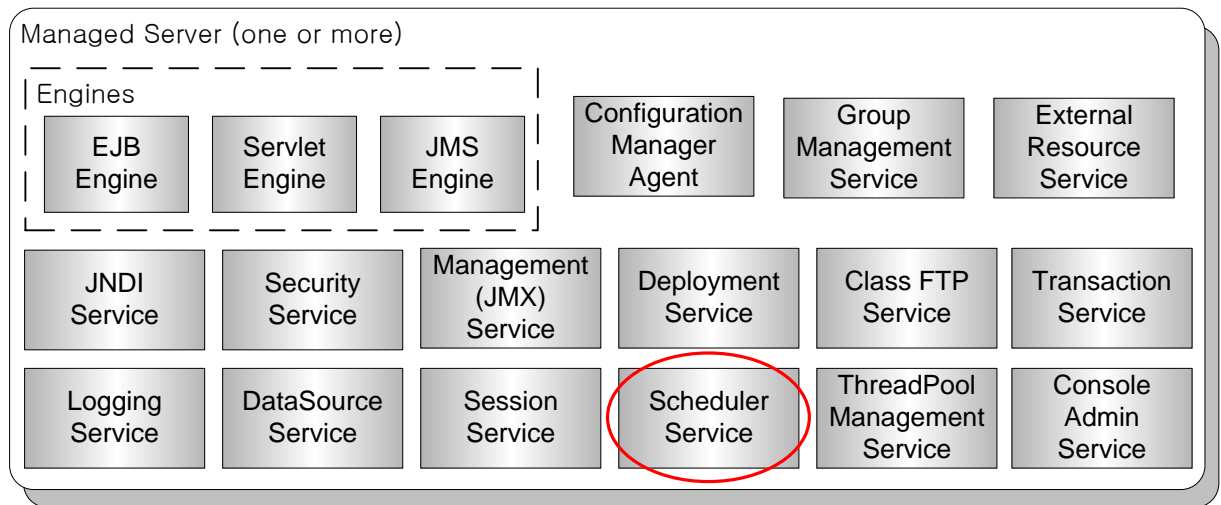
- Java SEアプリケーションでスタンドアロン・スケジューラーを使う
- Java EEアプリケーション・クライアントでスタンドアロン・スケジューラーを使う
- JEUSサーバーのスケジューラー・サービスをリモートでアクセスして使う
- JEUSサーバーの設定ファイルにジョブを登録して使う
- JEUSサーバーのスケジューラー・サービスをJava EEコンポーネントで使う

1.2. スケジューラー・コンポーネント

JEUSスケジューラーは、ユーザー・アプリケーション内で使用することも、JEUSサーバーで使用することもできます。JEUSサーバーは、リモートで接続できるスケジューラー・サービスを起動することができ、また設定ファイルによりスケジューリングされるジョブを登録することもできます。

以下は、JEUSにおけるスケジューラー・コンポーネントのあり方です。

[図 1.1] JEUSでのスケジューラー・コンポーネントの構成

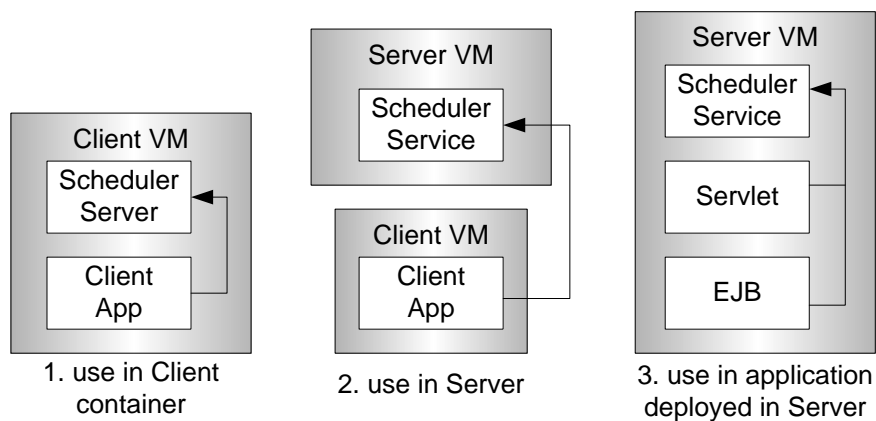


1.3. スケジューラー・サーバー

スケジューラー・サーバーのタイプによって、スケジューラーの実行方式が異なり、サービスが区分されます。

下の図は、スケジューラー・サーバーのタイプ別実行方式を示したものです。

[図 1.2] スケジューラー・サーバーのタイプ別実行方式



以下は、スケジューラー・サーバーのタイプ別のスケジューラー・サービスについての説明です。

- サーバーのスケジューラー・サービス

サーバーで周期的に実行される必要のあるジョブをJob-list設定に予め登録して、サーバーの起動時に実行させることができます。

Java EEコンポーネント(Servlet、JSP、EJBなど)が周期的に実行するジョブをスケジューリングするときに主に使用します。

JNDIリポジトリに登録され、リモート・クライアントで 사용할 こともできます。

- クライアントのスケジューラー・サービス

アプリケーション・クライアントでスタンドアロン方式で起動されるJEUSスケジューラー・サービスは、アプリケーション・クライアント内部であるジョブを周期的に実行するときに主に使 用します。アプリケーション・クライアントが終了すると、それ以上実行されないジョブに対して使 います。

第2章 スケジューラーのプログラミング

本章では、JEUSスケジューラーのプログラミングに必要な基本知識とJEUSスケジューラーの使用方法について説明します。

2.1. 概要

JEUS 5バージョンから次のような機能が追加または変更されました。JEUS 8スケジューラーは旧バージョンとの互換性を維持しているので、既存のプログラムでも修正せずに運用することができます。

- SchedulerListenerインターフェースの追加

ジョブを定義するためのインターフェースとして、ScheduleListenerインターフェースが追加されました。

ScheduleListenerインターフェースは最上位のジョブ・インターフェースで、すべてのジョブはこのインターフェースを実装する必要があります。既存のScheduleジョブ・クラスもScheduleListenerで実装しています。

そのため、Scheduleジョブ・クラスを継承してジョブを定義しなくても、ScheduleListenerを実装してジョブを定義することができます。

- SchedulerFactoryクラスの追加

以前使われていたクライアントとサーバー用のSchedulerManagerは使用されなくなり(deprecated)、SchedulerFactoryクラスが新たに追加されました。SchedulerFactoryクラスは、Schedulerオブジェクトを取得するために使われます。Schedulerオブジェクトを使用することで、サーバー、クライアント、リモート・クライアントなど、環境の区別なくジョブを登録することができます。

- 多様なメソッドの追加

Schedulerインターフェースにはジョブを登録する様々なメソッドが追加されました。たとえば、ジョブを登録する際に、実行開始時間、実行周期、実行終了時間、最大実行回数を指定できます。

- スレッドプールを利用したマルチ・スレッド方式への変更

既存のJEUSスケジューラーは基本的にシングル・スレッド方式でジョブを実行したのに対し、新しいJEUSスケジューラーはスレッドプールを利用してそれぞれのジョブを別々のスレッドで動作させます。そのため、あるジョブが実行中にブロックされたとしても、他のジョブの実行には影響を与えません。

- Job-list設定の追加

プログラミングによりジョブを登録するのではなく、JEUSサーバー設定(Job-list)によりジョブを登録できる機能が追加されました。

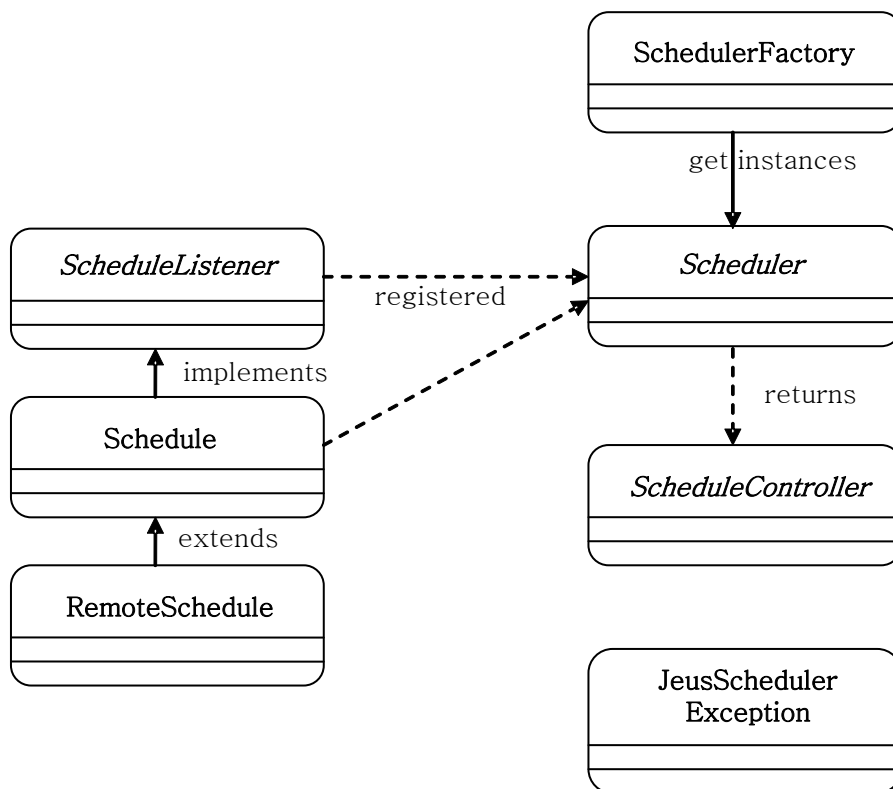
JEUSスケジューラーのプログラミングに関する理解を助けるために、まずJEUSスケジューラーを構成しているクラスを紹介し、続いてスケジューリング作業を定義する方法、ジョブの登録方法、登録されたジョブを制御する方法について説明します。その後、JEUSスケジューラーを使用できるケースを例示します。本章で扱うすべての例題ファイルは、JEUS_HOME/samples/schedulerディレクトリーで見つけることができます。

2.2. JEUS Schedulerクラス

基本的にJEUSスケジューラーはJ2SEタイマーと似た概念およびインターフェースを持っています。タスクを記述するjava.util.TimerTaskクラスはJEUSスケジューラーのjeus.schedule.ScheduleListenerインターフェースに対応し、タスクを登録するjava.util.TimerクラスはJEUSスケジューラーのjeus.schedule.Schedulerインターフェースに対応します。こうした類似性を考慮してJEUSスケジューラーを使用すると、すぐに使い慣れることができます。

以下では、Scheduler APIクラスについて説明します。

[図 2.1] スケジューラーAPIクラス



APIクラス	説明
jeus.scheduleパッケージ	JEUSスケジューラーのすべてのクラスとインターフェースは、jeus.scheduleパッケージとその下位パッケージに属しています
ScheduleListenerインターフェース	決められた時間に実行されるジョブは、ScheduleListenerインターフェースを実装したクラスで定義します。ScheduleListenerはコールバック・メソッドの1つであるonTime()を保持し、決められた時間になると、このメソッドが呼び出されます
Schedule抽象クラス	ScheduleListenerを実装した抽象クラスで、JEUS 5以前バージョンで使用していたクラスです。このクラスはonTime()の他にnextTime()コールバック・メソッドも保持し、ジョブを登録する際に呼び出し時間を予約するのではなく、ジョブを登録した後動的に次の呼び出し時間を決めることができます
RemoteSchedule抽象クラス	特殊なScheduleクラスで、initialize()コールバック・メソッドを保持しているため、オブジェクトを作成する際に初期化パラメータ値を取得することができます
SchedulerFactoryクラス	実際の(concrete)Schedulerオブジェクトを取得するために使います
Schedulerインターフェース	JEUSスケジューラーにジョブを登録するためのコア・インターフェースであり、多様なregisterSchedule()メソッドを定義しています
ScheduleControllerインターフェース	スケジューラーにジョブを登録すると、ScheduleControllerインターフェースを実装したオブジェクトが返されます。このハンドル・オブジェクトは、ジョブ情報を取得したり、ジョブをキャンセルしたりするときに使います
JeusSchedulerException例外	JEUSスケジューラーでジョブを登録またはキャンセルするときに、内部に問題が発生すると、JeusSchedulerExceptionが発生する可能性があります

参考

それぞれのインターフェースとクラスに関するより詳しい内容は、JEUSスケジューラーのJavadoc APIを参照してください。

2.3. ジョブの定義

ジョブを実行するためには、ジョブ・クラスを定義し、ジョブの実行時間や周期に合わせて適切な方法でジョブを定義する必要があります。

2.3.1. ScheduleListenerインターフェースの実装

決められた時間に実行されるジョブは、ScheduleListenerインターフェースを実装したクラスで定義します。

ScheduleListenerはコールバック・メソッドの1つであるonTime()を保持し、決められた時間になると、このメソッドが呼び出されます。ジョブ・クラスを定義するには、onTime()メソッドを実装し、メソッド内でジョブが実行されるようにプログラムを作成します。

[例 2.1] Taskオブジェクトの例

```
public class SimpleTask implements ScheduleListener {
    private String name;
    private int count;

    // no-arg constructor is required if classname is used for task registration
    public SimpleTask() {
    }

    public SimpleTask(String name) {
        this.name = name;
    }

    public void onTime() {
        count++;
        echo("##### " + name + " is waked on " + new Date());
    }

    ...
}
```

2.3.2. Scheduleクラスの継承

ScheduleListenerを直接実装せずに、ScheduleクラスやRemoteScheduleクラスを継承してジョブ・クラスを定義することもできます。

ScheduleクラスとRemoteScheduleクラスは、JEUS 5以前バージョンのスケジューラーで使われていたジョブ・クラスです。JEUS 7では一般のジョブを定義するときにScheduleListenerを実装するように勧めていますが、下位互換性を保持するために既存のジョブ・クラスもそのまま提供しています。

Schedule抽象クラスにはonTime()のほかにもnextTime()コールバック・メソッドがあり、ジョブを登録する際に呼び出し時間を予約せず、ジョブ・クラス内で次の呼び出し時間を決めるようにしています。そのため、固定周期のジョブよりは可変的な周期のジョブに適しています。

JEUSスケジューラーはScheduleジョブ・オブジェクトの最初の実行時間を決めるために、ジョブ・オブジェクトを登録した後、まずnextTime()を呼び出して最初の実行時間を決めます。そして、指定した時間になるとonTime()を呼び出してジョブを実行し、onTime()が終了すれば再びnextTime()を呼び出して次の実行時

間を決めます。nextTime()は次回ジョブが実行される時間をミリセカンド(ms)の値で渡す必要があります。0を返す場合、ジョブはそれ以上実行されません。

Scheduleジョブ・オブジェクトはonTime()が実行された後にnextTime()を呼び出すため、onTime()でジョブを実行した時間の分だけnextTime()の呼び出しが遅延されます。そのため、正確な時間間隔でジョブを呼び出すようにプログラミングすることは簡単ではありません。したがって、ジョブ内でnextTime()により繰り返し周期を実装するよりは、ジョブを登録する際に繰り返し周期を決めることが望ましいです。

[例 2.2] Scheduleオブジェクトの例

```
public class SimpleSchedule extends Schedule {
    private String name;
    private int count;
    private long period = 2000; // 2 seconds

    // no-arg constructor is required if classname is used for task registration
    public SimpleSchedule() {
    }

    public SimpleSchedule(String name) {
        this.name = name;
    }

    public void onTime() {
        count++;
        echo("##### " + name + " is waked on " + new Date());
    }

    public long nextTime(long currentTime) {
        return currentTime + period;
    }

    ...
}
```

2.3.3. RemoteScheduleクラスの継承

RemoteScheduleクラスは、リモートでジョブを登録するときに初期化変数を指定できるScheduleオブジェクトです。主にリモートでクラス名を使ってジョブ・オブジェクトを登録する際に使われます。

このクラスはinitialize()コールバック・メソッドを保持し、ジョブを登録した後に初期化パラメータで一度呼び出されます。したがって、リモートでクラス名を使ってジョブを登録するときに初期化値を設定する場合に使われます。

initialize()コールバックは、Scheduler.registerSchedule(classname, hashtable, daemon_flag)メソッドを使って、RemoteScheduleジョブ・オブジェクトを登録する場合にのみ呼び出されます。

[例 2.3] RemoteScheduleオブジェクトの例

```
public class SimpleRemoteSchedule extends RemoteSchedule {
    private String name;
    private int count;
    private long period;

    // no-arg constructor is required if classname is used for task registration
    public SimpleRemoteSchedule() {
    }

    // this is called by scheduler after creation
    public void initialize(Hashtable parameters) {
        name = (String) parameters.get("name");
        Long interval = (Long) parameters.get("interval");
        if (interval != null)
            period = interval.longValue();
        else
            period = 2000;
    }

    public void onTime() {
        count++;
        echo("##### " + name + " is waked on " + new Date());
    }

    public long nextTime(long currentTime) {
        return currentTime + period;
    }

    ...
}
```

参考

Job-listを使って、またはクラス名を使用するAPIを使ってジョブを登録する場合には、ジョブ・クラスにno-argコンストラクタ(デフォルト)が必ず必要です。これは、コンテナが当該クラスを初期化するためです。

2.4. Schedulerオブジェクトの取得

本節では、Schedulerオブジェクトを取得する方法について説明します。JEUSスケジューラーはローカル環境とリモート環境のどちらでも起動可能です。

2.4.1. ローカル環境でSchedulerオブジェクトを取得

ローカル環境で起動するとは、プログラムが起動されているローカルJVMでスケジューラー・インスタンスが生成され、登録されているすべてのジョブが同じJVMで起動されることを意味します。

現在JEUSスケジューラーは、JVM内で1つのインスタンスのみ生成されます。このインスタンスをデフォルト・スケジューラーといいます。現在はJVM内ですべてのクライアントはデフォルト・スケジューラーを共有することになります。ローカル環境のJEUSスケジューラーは、一般のJ2SEアプリケーションやJava EEアプリケーション・クライアント、Java EEコンポーネントなどで使われます。ローカル環境のJEUSスケジューラーを使用するには、SchedulerFactoryを利用します。

以下のように、簡単にデフォルト・スケジューラー・インスタンスを取得することができます。

```
// Get the default scheduler
Scheduler scheduler = SchedulerFactory.getDefaultScheduler();
```

2.4.2. リモート環境でSchedulerオブジェクトを取得

リモート環境で起動するとは、プログラムが起動されているJVMでない他のリモートのJVMでSchedulerインスタンスが生成され、登録されているすべてのジョブがリモートJVMで起動されることを意味します。

リモート・スケジューラーはRMIオブジェクトとして働くため、クライアントはRMIコールを通じてJEUSスケジューラーを使うことになります。JEUS環境ではJEUSサーバーでリモート・スケジューラー・サービスが起動します。リモート環境のJEUSスケジューラーは、リモート・クライアントがJEUSサーバーにジョブを登録するときに使われます。

JEUSサーバーのリモートJEUS スケジューラーを使うには、JNDIルックアップを利用します。

以下のように、JEUSサーバー・スケジューラーのインスタンス(Stub)を取得することができます。

```
// Get the remote scheduler
InitialContext ic = new InitialContext();
Scheduler scheduler = (Scheduler)ic.lookup(
    Scheduler.SERVER_SCHEDULER_NAME);
```

参考

JEUS 5以前バージョンで使われていたjeus.schedule.server.SchedulerManagerとjeus.schedule.client.SchedulerManagerは使用されなくなりました(deprecated)。その代替として、SchedulerFactoryを使ってSchedulerオブジェクトを取得し使用することを推奨します。ただし、下位互換性を保持するために、これらのクラスもそのまま提供しています。

2.5. ジョブの登録

本節では、Schedulerインターフェースを使ってジョブを登録する方法について説明します。

ローカル環境であれリモート環境であれ、スケジューラー・インスタンスを取得していれば、ジョブを登録する方法は同じです。ただ、リモートJEUSスケジューラーはジョブ・オブジェクトがリモートで転送(serialization)され、リモートで起動するという違いがあります。

2.5.1. 一回限りのジョブの登録

特定時間に1回だけ実行されるジョブは、1つの実行時間のみを設定してジョブを登録することができます。このとき、実行時間はjava.util.Dateオブジェクトで絶対時間を設定することもでき、ミリセカンド(ms)の値で現在時間からどれほどの時間が経過した後実行するかを設定することもできます。

以下のメソッドを利用してジョブを登録します。

```
registerSchedule(ScheduleListener task, Date time, boolean isDaemon)
registerSchedule(ScheduleListener task, long delay, boolean isDaemon)
```

以下は、メソッドの使用例です。

```
SimpleTask task1 = new SimpleTask("task1");
Date firstTime1 = new Date(System.currentTimeMillis() + 2000);
ScheduleController handle1 = scheduler.registerSchedule(task1, firstTime1, false);
```

参考

メソッドのパラメータについては、[「2.5.2. 繰り返されるジョブの登録」](#)を参照してください。

2.5.2. 繰り返されるジョブの登録

繰り返されるジョブは、実行開始時間、実行周期、実行終了時間、最大実行回数などを設定して登録することができます。ジョブの特性によって繰り返し周期をFixed-delay方式にするか、Fixed-rate方式にするかを決めます。

- Fixed-delay方式

ジョブの実行周期が一定に維持されます。ジョブの次回の実行時間は、前回の実行時間と周期によって決まります。もしジョブの実行が遅れ(ジョブの実行時間が長い場合や、ガーベジ・コレクションなどの外部的な理由によって遅れる場合)、次のジョブが実行される時間が経過した場合は、以降実行されるジョブはその分だけ実行が遅れてしまいます。そのため、長期的にはジョブの実行時間が少しずつ遅れることもあり得ます。

- Fixed-rate方式

ジョブの実行比率が一定に維持されます。ジョブの次回の実行時間はジョブの開始時間と周期によって決まります。ジョブの実行が遅れても、次のジョブは先行ジョブに続いて実行され、時間当たりの実行比率を維持します。長期的にジョブの実行時間は初期指定した周期で維持されます。

参考

Fixed-rate方式でジョブを登録すれば、JEUSスケジューラーは比較的正確な呼び出し時間を保証するために、ジョブの実行が遅れても設定した時間になると他のスレッドによってジョブを呼び出します。そのため、ジョブの実行が遅れる場合は、ジョブが同時に実行されることになり、ジョブ・オブジェクトがスレッドセーフであるかを考慮する必要があります。

以下のメソッドを利用してジョブを登録します。

```
registerSchedule(ScheduleListener task, Date firstTime, long period, Date endTime,
                long maxcount, boolean isDaemon)

registerSchedule(ScheduleListener task, long delay, long period, Date endTime,
                long maxcount, boolean isDaemon)

registerScheduleAtFixedRate(ScheduleListener task, Date firstTime, long period,
                            Date endTime, long maxcount, boolean isDaemon)

registerScheduleAtFixedRate(ScheduleListener task, long delay, long period,
                            Date endTime, long maxcount, boolean isDaemon)
```

以下は、ジョブを登録するときに使用するパラメータです。

パラメータ	説明
Date firstTime	開始時間です。最初に実行される時間を指定します
long delay	開始時間です。現在以降、初めて実行される時間を指定します(単位 : ms)
long period	繰り返し実行周期を指定します(単位 : ms)
Date endTime	終了時間です。終了時間以降、ジョブは実行されません。nullの場合は、終了時間の制約がありません
long maxcount	最大実行回数です。Scheduler.UNLIMITEDの場合は、制限がありません
boolean isDaemon	リモートでスケジュールを登録する場合にのみ意味があります。trueに設定する場合、クライアントとの接続が終了されるときに、ジョブが終了されます。 現在はRMIランタイムの分散ガベージ・コレクション(DGC)ポリシーによりクライアントの接続の終了を判断するため、実際にクライアントの接続が終了してから15分程度経過してから終了したことが検知され、スケジューリングが取り消されます

パラメータ	説明
boolean isThreaded	使用されなくなりました(deprecated)

以下は、繰り返されるジョブを登録する例です。

[例 2.4] 繰り返しジョブの登録

```
SimpleTask task2 = new SimpleTask("task2");
ScheduleController handle2 = scheduler.registerSchedule(task2, 2000, 2000, null,

    Scheduler.UNLIMITED, false);

SimpleTask task3 = new SimpleTask("task3");
Date firstTime3 = new Date(System.currentTimeMillis() + 2000);
Date endTime3 = new Date(System.currentTimeMillis() + 10 * 1000);
ScheduleController handle3 = scheduler.registerScheduleAtFixedRate(task3,
    firstTime3, 2000, endTime3, 10, false);
```

2.5.3. Scheduleオブジェクトの登録

下位互換性を維持するために、ScheduleオブジェクトとRemoteScheduleオブジェクトの登録機能を提供しています。

ジョブの実行開始時間と以降の繰り返し実行時間は、ScheduleオブジェクトのnextTime()メソッドを利用するため、登録の際に別途のパラメータを設定する必要がありません。

以下のメソッドを利用してジョブを登録します。

```
registerSchedule(Schedule task, boolean isDaemon)
registerSchedule(String classname, Hashtable params, boolean isDaemon)
```

以下はジョブを登録する例です。

[例 2.5] Scheduleオブジェクトの登録

```
Hashtable params = new Hashtable();
params.put("name", "task3");
params.put("interval", new Long(3000));
ScheduleController handle3 = scheduler.registerSchedule(
    "samples.scheduler.SimpleRemoteSchedule", params, true);

SimpleSchedule task4 = new SimpleSchedule("task4");
ScheduleController handle4 = scheduler.registerSchedule(task4, true);
```

注

スケジューラーはManaged Server(MS)が提供するサービスです。設定に間違いがありスケジューラーを生成できなかった場合、MSはエラーメッセージを表示し起動に失敗するので、設定の際には注意が必要です。

2.6. ジョブの制御

JEUSスケジューラーにジョブを登録すると、ハンドル・オブジェクトのScheduleControllerを返します。このオブジェクトは登録されるジョブごとに作成され、ジョブの情報を取得したり、ジョブをキャンセルしたりするなど、登録されたジョブの制御に使われます。

以下は、ScheduleController.cancel()メソッドを呼び出してジョブをキャンセルする例です。

[例 2.6] ScheduleController.cancelメソッドの使用

```
SimpleTask task2 = new SimpleTask("task2");
ScheduleController handle2 = scheduler.registerSchedule(task2, 2000, 2000, null,
                                                         Scheduler.UNLIMITED, false);

Thread.sleep(10 * 1000);
handle2.cancel();
```

2.7. スケジューラーの使用

ジョブの定義、Schedulerオブジェクトの取得、ジョブの登録、ジョブの制御まで完了すれば、JEUSスケジューラーを使用する準備がすべて完了したことになります。

本節では、環境別のスケジューラーの使用方法について説明します。

2.7.1. スタンドアロン環境での使用

一般のJ2SEアプリケーションやJava EEアプリケーション・クライアントで、JEUSスケジューラーを使うことができます。この場合は、JEUSサーバーとは別に、JEUSスケジューラーをJ2SEタイマーと同様にライブラリーのように使えます。Schedulerオブジェクトを取得するには、SchedulerFactoryクラスを使用します。また、ローカル環境でジョブを登録する際にはデーモン・フラグは使われないため、どの値を入れても関係ありません。

以下はスタンドアロン・クライアントの例です。

[例 2.7] スタンドアロン・クライアントでのスケジューラーの使用

```
public class StandAloneClient {
    public static void main(String args[]) {
        try {
            // Get the default scheduler
            Scheduler scheduler = SchedulerFactory.getDefaultScheduler();

            // Register SimpleTask which runs just one time
            echo("Register task1 which runs just one time...");
            SimpleTask task1 = new SimpleTask("task1");
            Date firstTime1 = new Date(System.currentTimeMillis() + 2000);
            ScheduleController handle1 = scheduler.registerSchedule(task1,
                firstTime1, false);

            Thread.sleep(5 * 1000);
            echo("");

            // Register SimpleTask which is repeated
            // with fixed-delay
            echo("Register task2 which is repeated " + "until it is canceled...");

            SimpleTask task2 = new SimpleTask("task2");
            ScheduleController handle2 = scheduler.registerSchedule(task2, 2000,
                2000, null, Scheduler.UNLIMITED, false);

            Thread.sleep(10 * 1000);
            handle2.cancel();
            echo("");

            // Register SimpleTask which is repeated
            // with fixed-rate
            echo("Register task3 which is repeated " + "for 10 seconds...");
            SimpleTask task3 = new SimpleTask("task3");
            Date firstTime3 = new Date(System.currentTimeMillis() + 2000);
            Date endTime3 = new Date(System.currentTimeMillis() + 10 * 1000);
            ScheduleController handle3 = scheduler.registerScheduleAtFixedRate(
                task3, firstTime3, 2000, endTime3, 10, false);

            Thread.sleep(12 * 1000);
            echo("");

            // Register SimpleSchedule which is repeated
            // every 2 seconds
            echo("Register task4 which is repeated " + "every 2 seconds...");
            SimpleSchedule task4 = new SimpleSchedule("task4");
```

```

        ScheduleController handle4 = scheduler.registerSchedule(task4, false);

        Thread.sleep(10 * 1000);
        echo("");

        // Cancel all tasks
        echo("Cancel all tasks registered on the scheduler...");
        scheduler.cancel();
        Thread.sleep(5 * 1000);

        System.out.println("Program terminated.");
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private static void echo(String s) {
    System.out.println(s);
}
}

```

Java EEアプリケーション・クライアントでJEUSスケジューラーを使う場合は、DD(jeus-client-dd.xml)にJEUSスケジューラーのスレッドプールを設定できます。

参考

1. スケジューラーのスレッドプール関連設定については、『JEUS リファレンスガイド』を参照してください。
 2. スケジューラーを使用したコードをコンパイルするか起動するには、JEUS関連クラス(jeus.jarなど)がクラスパスに指定されている必要があります。
-

2.7.2. JEUSサーバーでの使用

JEUSサーバーでスケジューラー・サービスが起動されていれば、リモート・クライアントはサービスを使用することができます。そのためには、JEUSサーバーでスケジューラー・サービスが起動されるように予め設定されている必要があります。

JEUSサーバーのスケジューラー・サービスはRMI SchedulerオブジェクトをJNDIに登録します。したがって、クライアントはJNDIルックアップによりリモートのSchedulerオブジェクト(実際にはStubオブジェクト)を取得できます。このオブジェクトのJNDI名は「jeus_service/Scheduler」で、Scheduler.SERVER_SCHEDULER_NAME定数を使用します。

Schedulerオブジェクトを取得した後、ジョブを登録する方法はスタンドアロン環境と同じです。ただ、登録されたジョブ・オブジェクトは転送(Serialization)され、リモートのスケジューラーで動作します。つまり、JEUSサーバーで実行されます。

この場合、ジョブを登録する際にデーモン・フラグは意味があります。デーモン・フラグをtrueに設定すると、リモートのクライアントが終了するときにリモートのジョブも終了されます。

参考

JEUSサーバーでスケジューラー・サービスが起動されるように設定する方法については、『JEUS リファレンスガイド』を参照してください。

以下は、リモート・クライアントの例です。

[例 2.8] リモート・クライアントでのスケジューラーの使用

```
public class RemoteClient {
    public static void main(String args[]) {
        try {
            // Get the remote scheduler
            InitialContext ic = new InitialContext();
            Scheduler scheduler = (Scheduler)ic.lookup(
                Scheduler.SERVER_SCHEDULER_NAME);

            // Register SimpleTask which runs just one time
            echo("Register task1 which runs just one time...");
            SimpleTask task1 = new SimpleTask("task1");
            Date firstTime1 = new Date(System.currentTimeMillis() + 2000);
            ScheduleController handle1 = scheduler.registerSchedule(task1,
                firstTime1, true);

            Thread.sleep(5 * 1000);
            echo("");

            // Register SimpleTask which is repeated
            // with fixed-delay
            echo("Register task2 which is repeated " + "until it is canceled...");

            SimpleTask task2 = new SimpleTask("task2");
            ScheduleController handle2 = scheduler.registerSchedule(task2,
                2000, 2000, null, Scheduler.UNLIMITED, true);

            Thread.sleep(10 * 1000);
            handle2.cancel();
            echo("");
        }
    }
}
```

```

        // Register SimpleRemoteSchedule which is repeated
        // every 3 seconds
        echo("Register task3 which is repeated " + "every 3 seconds...");
        Hashtable params = new Hashtable();
        params.put("name", "task3");
        params.put("interval", new Long(3000));
        ScheduleController handle3 = scheduler.registerSchedule(
            "samples.scheduler.SimpleRemoteSchedule", params, true);

        Thread.sleep(10 * 1000);
        echo("");

        // Cancel all tasks
        echo("Cancel all tasks registered on the scheduler...");
        scheduler.cancel();
        Thread.sleep(5 * 1000);

        System.out.println("Program terminated.");
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

private static void echo(String s) {
    System.out.println(s);
}
}

```

注

上記の例題を実行するには、該当するジョブのクラス・ファイルをJARファイルでまとめてDOMAIN_HOME/lib/application、またはSERVER_HOME/lib/applicationにコピーして移す必要があります。JEUSがすでに起動されている場合は再起動する必要があります。JEUSサーバーがジョブを実行するには、該当するクラスをロードする必要があるためです。

2.7.3. Java EEコンポーネントでの使用

EJBやサーブレットなどのJava EEコンポーネントでJEUSスケジューラーを使うことができます。この場合、JEUSスケジューラーはJEUSサーバーで起動されます。

EJB 2.1標準ではEJBタイマー・サービスを明記しており、EJBタイマー・サービスを提供しています。したがって、EJBコンポーネントの場合は、Java EE標準を遵守するには、JEUSスケジューラーよりはEJBタイマー・サービスの使用を推奨します。しかし、EJB以外のJava EEコンポーネントではEJBタイマー・サービスを使用できないため、JEUSスケジューラーを使用する必要があります。

Java EEコンポーネントでのJEUSスケジューラーの使用方法は、スタンドアロン環境でJEUSスケジューラーを使用する方法と同じです。SchedulerFactoryクラスを利用し、サーバーで起動されるSchedulerオブジェクトを取得した後、必要な登録メソッドを呼び出して、ジョブを登録します。

参考

JEUSサーバーで動作するJEUSスケジューラーに対し、スレッドプールを設定できます。設定方法については、「[3.4. スレッドプールの設定](#)」を参照してください。

以下は、EJBでのJEUSスケジューラーの使用例です。

[例 2.9] EJBでのJEUSスケジューラーの使用

```
public class HelloEJB implements SessionBean {
    private SimpleTask task;
    private ScheduleController taskHandler;
    private boolean isStarted;

    public HelloEJB() {
    }

    public void ejbCreate() {
        task = new SimpleTask("HelloTask");
        isStarted = false;
    }

    public void trigger() throws RemoteException {
        if (!isStarted) {
            Scheduler scheduler =
                SchedulerFactory.getDefaultScheduler();
            taskHandler = scheduler.registerSchedule(
                task, 2000, 2000, null, Scheduler.UNLIMITED, false);
            isStarted = true;
        }
    }

    public void ejbRemove() throws RemoteException {
        if (isStarted) {
            taskHandler.cancel();
            isStarted = false;
        }
    }

    public void setSessionContext(SessionContext sc) {
    }

    public void ejbActivate() {
    }
}
```

```

    }

    public void ejbPassivate() {
    }
}

public class HelloClient {
    public static void main(String args[]) {
        try {
            InitialContext ctx = new InitialContext();

            HelloHome home = (HelloHome) ctx.lookup("helloApp");
            Hello hello = (Hello) home.create();
            hello.trigger();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

2.8. Job-listの使用

JEUSサーバーにプログラミング方式でジョブを登録する代わりに、設定ファイルを使ってジョブを登録することができます。Job-listはJEUSサーバーのスケジューラーに登録することができます。Job-list設定でジョブを登録すると、ジョブはFixed-rate方式で繰り返し実行されます。

参考

JEUSサーバーで動作するJEUSスケジューラーにJob-listを設定できます。Job-listの設定方法については、[「3.2. Job-listの設定」](#)を参照してください。

第3章 スケジューラーの設定

本章では、JEUSの設定ファイルやデプロイメント記述子(DD)にJEUSスケジューラーを設定する方法について説明します。

3.1. 概要

JEUSスケジューラー・サービスを使用するには、サービス別に必要な内容を設定する必要があります。

● JEUSサーバーにスケジューラー・サービスを設定

リモートでJEUSサーバーのスケジューラー・サービスにアクセスする場合や、Job-list設定を用いてJEUSサーバーで周期的な作業を行うためには、サーバーでスケジューラー・サービスをアクティブ化する必要があります。WebAdminでスケジューラー・サービスを設定すると、スケジューラー・サービスがアクティブ化し、サーバーの起動時にスケジューラー・サービスが開始されます。

- スケジューラー・サービスを使用するように設定します。
- スケジューラー・サービスで使用するスレッドプールを設定します。
- サーバーでスケジューリングおよび実行されるジョブ・リストを設定します。

● クライアント・コンテナにスケジューラー・サービスを設定

- デプロイメント記述子(DD)で設定します。

スケジューラーを使用するには、サービスをアクティブ化する設定、スレッドプールの設定、および実際に作業を行うジョブ・リストの設定が必要です。

3.2. Job-listの設定

本節では、実際に作業を行うジョブを登録する方法について説明します。

JEUSサーバーを起動する際に自動でジョブがスケジューリングされるようにするには、プログラム・コードでジョブを登録する方法のほかに、JEUS設定ファイルにジョブを登録する方法があります。ジョブはスケジューリングされる1つの作業単位です。ジョブのクラスは必ず`jeus.schedule.ScheduleListener`を実装している必要があり、当該クラスと関連クラスをJARファイルにまとめて以下のパスに配置します。

下のディレクトリーは、Domain Administration Server(以下、DAS)とManaged Server(以下、MS)間で同期化されないため、ユーザーが手動で同期化する必要があります。

```
DOMAIN_HOME/lib/application
```

下のディレクトリーは、JEUSをインストールするときに自動で生成されないので、ユーザーが直接作成する必要があります。

```
SERVER_HOME/lib/application
```

ジョブをサーバーに設定すると、JEUSスケジューラー・サービスでジョブが実行されます。

以下は、ジョブを設定するときに注意が必要な項目です。

項目	説明
Begin Time	ジョブの実行開始時間を指定します。設定しない場合、JEUSサーバーの起動時に実行されます。 – タイプ: XML dateTime type – 形式: yyyy-mm-ddThh:mm:ss.sss 登録したジョブのBegin Timeが過去である場合は、周期によって現在時間以降、最初実行時間に初めて実行されるように調整されます
End Time	ジョブの実行終了時間を指定します。設定しない場合は、終了されません。 – タイプ: XML dateTime type – 形式: yyyy-mm-ddThh:mm:ss.sss End Timeが過去である場合は、ジョブは一度も実行されません
Count	ジョブの最大実行回数を指定します。 設定しないか、-1の場合は、最大実行回数に制限がありません

参考

Job-listで登録されたジョブは、Fixed-rate方式で繰り返されます。したがって、比較的正確な時間に呼び出されますが、ジョブの実行に時間がかかる場合にはジョブが並行して実行されることもあるので、ジョブがスレッド・セーフであるように用心する必要があります。

3.3. サーバーのスケジューラー設定

本節では、WebAdminを使って実際にサーバーにスケジューラーを設定する方法について説明します。

リモートでJEUSサーバーのスケジューラー・サービスにアクセスするか、Job-list設定を用いてJEUSサーバーで周期的な作業を行う場合には、サーバーでスケジューラー・サービスをアクティブ化する必要があります。以下で説明する方法でWebAdminでその設定を行いますと、サーバーが起動するときにスケジューラー・サービスが開始されます。

以下は、「[2.3.2. Scheduleクラスの継承](#)」で作成した例をサーバーにジョブとして追加する手順です。

1. WebAdminの左側のメニューで**[Servers]**を選択すると、サーバー・リスト照会画面が表示されます。サーバー・リストからスケジューラー設定を変更するサーバーを選択します。

[図 3.1] WebAdminサーバー・リスト画面

domain1

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

2 Running

0 Shutdown

0 Suspended

0 Other

Runtime Info

LOCK & EDIT

指定したドメインの項目を探索、追加、削除する機能です。

運用マニュアル

Domain

Server もっと見る

Servers

HISTORY

ドメイン内でJEUSサーバを構成するとき、各サーバの設定を行います。

ヘルプ

Servers

No Group node Group Cluster

	Name	Status	Pid	Need To Restart	Command	Delete	Add
<input type="checkbox"/>	adminServer (*)	RUNNING(03:01:27)	11728	false	Start Stop	Delete Duplicate	
<input type="checkbox"/>	server1	RUNNING(02:39:08)	2796	false	Start Stop	Delete Duplicate	

Server Templates

Name

Add

該当する内容が存在しません。

2. **Servers**設定画面で、**[Resource] > [Scheduler]**メニューを選択すると、**Scheduler**設定画面に移動します。

[図 3.2] WebAdminのスケジューラー設定画面

The screenshot displays the JEUS Scheduler configuration interface. On the left is a sidebar for 'domain1' with navigation links: Domain, Session, Clusters, Servers (selected), Applications, Security, Resources, Monitoring, and Console. Below these are system status indicators (0 Failed, 0 Standby, 2 Running, 0 Shutdown, 0 Suspended, 0 Other) and buttons for 'Runtime Info', 'LOCK & EDIT', and a '運用マニュアル' (Operation Manual) section with links for 'Domain' and 'Server'. The main content area is titled 'Scheduler' and includes a 'HISTORY' dropdown. A message states 'JEUSスケジューラに関する設定です。' (Settings for JEUS Scheduler). Below this are tabs for 'Basic', 'Resource' (selected), and 'Engine'. A breadcrumb trail shows 'Listener | Jms Resource | Jmx Manager | Scheduler | Lifecycle Invocation | External Resource'. A tip indicates that settings can be locked or edited via the left menu. The '動的設定' (Dynamic Settings) section shows 'Enabled' with a checkbox and a default value of 'true'. The '詳細設定' (Detailed Settings) section includes 'Pooling' (Shared/Dedicated), 'Reserved Thread Num', 'Min/Max' thread counts, 'Keep Alive Time' (ms), 'Queue Size', 'Stuck Thread Handling' (Max Stuck Thread Time, Action On Stuck Thread, Stuck Thread Check Period), and a 'Lock & Edit' button.

3. WebAdminの左側のメニューの下部にある[LOCK & EDIT]ボタンをクリックし、スケジューラー設定を変更するためのロックを取得します。ロックの設定変更に関する詳細は、『JEUS WebAdminガイド』の「2.4.4. ロック機能」を参照してください。

ロックを設定すると、スケジューラー設定を変更することができます。

[図 3.3] WebAdminのスケジューラー設定 - ロック設定

The screenshot displays the JEUS WebAdmin interface for configuring the Scheduler. The left-hand navigation pane is expanded, showing the 'domain1' menu with options like Domain, Session, Clusters, Servers, Applications, Security, Resources, Monitoring, and Console. The 'Schedulers' link is highlighted. The main content area is titled 'Scheduler' and contains a tabbed interface with 'Basic', 'Resource', and 'Engine' tabs. The 'Resource' tab is active, showing a list of resources with columns for Name, Class Name, Begin Time, End Time, Interval, and Count. The 'Enabled' checkbox is checked, and the 'Pooling' checkbox is unchecked. Below the table, a message states '該当する内容が存在しません。' (No matching content exists). The bottom of the page shows the 'Job' section with a table header and an 'Add' button.

4. 「Enabled」項目にチェックを入れ、詳細設定領域でスケジューラーで使用するスレッドプール情報を設定します。スケジューラーのスレッドプールに関する内容は「3.4. スレッドプールの設定」を参照してください。

設定が完了すると、**[確認]**ボタンをクリックします。設定が保存されると、画面上部のメッセージボックスに設定変更が保存されたという結果メッセージが表示されます。「Enabled」項目にチェックが入っている場合には、この手順を省略します。

[図 3.4] WebAdminのスケジューラー設定 - スケジューラー使用の設定

Scheduler

HISTORY

JEUSスケジューラに関する設定です。

ヘルプ

Basic Resource Engine

Listener | Jms Resource | Jmx Manager | Scheduler | Lifecycle Invocation | External Resource

動的設定 必須項目

確認 再設定 削除

Enabled ☒ [デフォルト: true] スケジューラサービスの実行有無を指定します。

詳細設定

すべてを開く

Pooling ☒

Shared ☒ Dedicated ☐

Reserved Thread Num

Min

Max

Keep Alive Time ms

Queue Size

Stuck Thread Handling

Max Stuck Thread Time ms

Action On Stuck Thread

Stuck Thread Check Period ms

確認 再設定 削除

5. ジョブを追加するには、画面下部のJobリストで[Add]ボタンをクリックし、ジョブを追加します。

[図 3.5] WebAdminのスケジューラー設定 - ジョブの追加

Job

Name	Class Name	Begin Time	End Time	Interval	Count	Add
該当する内容が存在しません。						

6. Job設定画面でジョブの名前とジョブを実行するクラス名、ジョブの実行周期を入力し、[確認]ボタンをクリックします。

[図 3.6] WebAdminのスケジューラー設定 - ジョブの保存

Job HISTORY

スケジューラに登録する1つのジョブを指定します。 ヘルプ

Basic **Resource** **Engine**

Listener | Jms Resource | Jmx Manager | **Scheduler** | Lifecycle Invocation | External Resource

☐ 動的設定 ☒ 必須項目 確認 再設定

Name *	<input type="text" value="job1"/> <small>ジョブの名前を指定します。(IDに対する検証(Validation)が必要)</small>
Class Name *	<input type="text" value="test.scheduler.TestScheduler"/> <small>ジョブを実行するクラスの完全修飾名を設定します。</small>
Description	<input type="text"/> <small>ジョブの説明が記述できます。</small>
Count	<input type="text"/> <small>[デフォルト:-1] ジョブの実行回数を設定します。</small>
Begin Time	<input type="text"/> <small>ジョブの開始時間を設定します。設定していない場合、即時に開始されます。</small> EX 2012-03-13T16:53:42,333
End Time	<input type="text"/> <small>ジョブの終了時間を設定します。設定していない場合、終了しません。</small> EX 2012-03-13T16:53:42,333

Interval
ジョブが実行される周期を設定します。

<input type="radio"/> Millisecond	<input type="text"/> <small>周期をミリ秒単位で設定します。</small> ms
<input checked="" type="radio"/> Minutely	<input type="text" value="30"/> <small>周期を分単位で指定します。</small> m
<input type="radio"/> Hourly	<input type="text"/> <small>周期を時間単位で指定します。</small> h
<input type="radio"/> Daily	<input type="text"/> <small>周期を日数単位で指定します。</small> d

確認 再設定

7. 以下は、**[確認]**ボタンをクリックして変更された内容が一時的に保存された後の結果画面です。画面上部では一時的保存の結果メッセージを、**Job**リストでは追加したジョブを確認できます。

【図 3.7】 WebAdminのスケジューラー設定 - ジョブ追加実行の確認

Scheduler

HISTORY

JEUSスケジューラに関する設定です。

追加されました。

ヘルプ

Basic Resource Engine

Listener | Jms Resource | Jmx Manager | Scheduler | Lifecycle Invocation | External Resource

動的設定 必須項目

確認 再設定 削除

Enabled ☒ [デフォルト: true] スケジューラサービスの実行有無を指定します。

詳細設定 [すべてを開く](#)

Pooling ☐

確認 再設定 削除

Job

Name	Class Name	Begin Time	End Time	Interval	Count	Add
job1	test.scheduler.TestScheduler			30 m		Delete

8. WebAdminの左側のメニューの下部にある**[Activate Changes]**ボタンをクリックして、スケジューラーの追加内容を反映します。

参考

運用中のサーバーにはスケジューラーの追加・変更作業が動的に反映されません。設定のみ保存した後、サーバーが再起動されるときに反映されます。

【図 3.8】 WebAdminのスケジューラー設定 - 追加されたジョブの反映

domain1

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

2 Running

0 Shutdown

0 Suspended

0 Other

Runtime Info

LOCK & EDIT

指定したドメインの項目を登録、追加、削除する機能です。

運用マニュアル

Domain

Server もっと見る

Scheduler

HISTORY

JEUSスケジューラに関する設定です。

domain.xmlの設定を変更しました。

domain.xml : PENDING

servers.server.[? name == 'server1'].scheduler : PENDING

変更内容を適用するには、サーバを再起動してください。

ヘルプ

Basic Resource Engine

Listener | Jms Resource | Jmx Manager | Scheduler | Lifecycle Invocation | External Resource

動的設定 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。

Enabled ☒ [デフォルト: true] スケジューラサービスの実行有無を指定します。

詳細設定

すべてを開く

Pooling ☐

Shared

Reserved Thread Num

Dedicated

Min

Max

Keep Alive Time ms

Queue Size

Stuck Thread Handling

Max Stuck Thread Time ms

Action On Stuck Thread

Stuck Thread Check Period ms

このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。

Job

Name	Class Name	Begin Time	End Time	Interval	Count	
job1	test.scheduler.TestScheduler			30 m	-1	Add Delete

9. サーバーを再起動すると、追加されたジョブが実行され、サーバー・ログでロギング結果を照会できます。

```
##### waked on Tue Apr 23 14:50:19 KST 2013
##### waked on Tue Apr 01 15:20:19 KST 2013
```

3.4. スレッドプールの設定

本節では、WebAdminとコンソール・ツールを使ってスレッドプールを設定する方法について説明します。設定項目に関する詳細は、『JEUS サーバガイド』の「2.3.3. スレッド・プールの設定」を参照してください。スケジューラー・サービスはスレッドプールの設定により、スケジューラー・サービスを実行するために必要なスレッドの数を調整します。スレッドプールは、システム・スレッドプールを共有して使用する**共用スレッドプール**と、別のスレッドプールを設定する**専用スレッドプール**に分けられます。

3.4.1. 共用スレッドプール

スレッドの数を予め割り当てただけで、スケジューラー・サービスで共用スレッドプールを使用できます。

WebAdminの使用

以下は、WebAdminを使ってスレッドプールを設定する手順です。スレッドプールを設定する前にロック設定を変更する必要があります。

1. WebAdminの左側のメニューで**[Servers]**を選択すると、**サーバー・リスト照会画面**([\[図 3.1\]](#))が表示されます。

サーバー・リストから設定するサーバー(server1)を選択し、**[Resource] > [Scheduler]**メニューを選択すると、**Scheduler画面**([\[図 3.2\]](#))に移動します。
2. WebAdminの左側のメニューの下部にある**[LOCK & EDIT]**ボタンをクリックして設定変更モードに切り替えた後、詳細設定領域でスレッドプールの設定を変更できます。ロック設定変更モードについての詳細は『JEUS WebAdminガイド』の「2.4.4. ロック機能」を参照してください。

3. 詳細設定のPooling設定で「Shared」項目を選択し、「Reserved Thread Num」の数を10に設定します。[確認]ボタンをクリックすると、変更した設定が一時保存され、画面上部のメッセージボックスに結果メッセージが表示されます。

[図 3.9] スレッドプールの設定 - 共用スレッドプールの設定(1)

The screenshot displays the JBoss Scheduler configuration page for 'domain1'. The left sidebar contains navigation links for Domain, Session, Clusters, Servers, Applications, Security, Resources, Monitoring, and Console. The 'Servers' link is selected. The main content area is titled 'Scheduler' and includes a 'HISTORY' dropdown. Below the title, there's a breadcrumb trail: Listener | Jms Resource | Jmx Manager | Scheduler | Lifecycle Invocation | External Resource. The 'Scheduler' tab is active. The 'Enabled' checkbox is checked, with a default value of 'true'. The 'Pooling' section is expanded, showing the 'Shared' radio button selected. The 'Reserved Thread Num' is set to 10. Other settings like 'Min', 'Max', 'Keep Alive Time', 'Queue Size', 'Stuck Thread Handling', 'Max Stuck Thread Time', 'Action On Stuck Thread', and 'Stuck Thread Check Period' are also visible. The bottom of the page shows a 'Runtime Info' section with 'Activate Changes' and 'Undo All Changes' buttons, and a '運用マニュアル' (Operation Manual) link.

4. WebAdminの左側のメニューの下部にある[**Activate Changes**]ボタンをクリックしてサーバーに反映します。サーバーへの反映が完了すると、設定したスケジューラーのスレッドプール設定がサーバーに反映され、WebAdmin画面上部にその結果メッセージが表示されます。

[図 3.10] スレッドプールの設定 - 共用スレッドプールの設定(2)

domain1

Domain

Session

Clusters

Servers

Applications

Security

Resources

Monitoring

Console

システム状態

0 Failed

0 Standby

2 Running

0 Shutdown

0 Suspended

0 Other

Runtime Info

LOCK & EDIT

指定したドメインの項目を探索、追加、削除する機能です。

運用マニュアル

Domain

Server

もっと見る

Scheduler

HISTORY

JEUSスケジューラに関する設定です。

domain.xmlの設定を変更しました。

domain.xml : PENDING

servers.server.[? name == 'server1'].scheduler : PENDING

変更内容を適用するには、サーバを再起動してください。

ヘルプ

Basic Resource Engine

Listener | Jms Resource | Jmx Manager | Scheduler | Lifecycle Invocation | External Resource

動的設定 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。TIP

Enabled ☒ [デフォルト: true] スケジューラサービスの実行有無を指定します。

詳細設定

すべてを開く

Pooling ☒

Shared

Reserved Thread Num 10

Dedicated

Min

Max

Keep Alive Time ms

Queue Size

Stuck Thread Handling

Max Stuck Thread Time ms

Action On Stuck Thread

Stuck Thread Check Period ms

このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。TIP

コンソール・ツールの使用

以下は、コンソールツール(jeusadmin)を使ってスケジューラーのスレッドプールを設定する方法です。

[例 3.1] jeusadminによるスレッドプール設定

```
[DAS]domain1.adminServer>modify-system-thread-pool server1 -service scheduler -r
10
Successfully performed the MODIFY operation for The scheduler thread pool of the
server (server1)., but all changes were non-dynamic. They will be applied after
restarting.
Check the results using "show-system-thread-pool server1 -service scheduler or
modify-system-thread-pool server1 -service scheduler".

[DAS]domain1.adminServer>show-system-thread-pool server1 -service scheduler
Shows the current configuration.
the system thread pool of the server (server1)
=====
+-----+-----+
| Min                | 0                |
| Max                | 100              |
| Keep-Alive Time    | 300000           |
| Queue Size         | 4096             |
| Max Stuck Thread Time | 3600000         |
| Action On Stuck Thread | IGNORE_AND_REPLACE |
| Stuck Thread Check Period | 300000         |
| Reserved Threads for the Service transaction | 0                |
| Reserved Threads for the Service scheduler   | 10               |
| Reserved Threads for the Service namingserver | 0                |
+-----+-----+
=====
```

3.4.2. 専用スレッドプール

スケジューラー・サービスで専用スレッドプールを使用する場合も、WebAdminまたはコンソール・ツール(jeusadmin)を使って設定を行います。本節では、専用スレッドプールを設定する2つの方法について説明します。

WebAdminの使用

以下は、WebAdminを使ってスレッドプールを設定する手順です。スレッドプールを設定する前にロック設定を変更する必要があります。

1. WebAdminの左側のメニューで**[Servers]**を選択すると、サーバー・リスト照会画面([\[図 3.1\]](#))が表示されます。

サーバー・リストから設定するサーバーを選択し、**[Resource] > [Scheduler]**メニューを選択すると、**Scheduler画面**([\[図 3.2\]](#))に移動します。

2. WebAdminの左側のメニューの下部にある**[LOCK & EDIT]**ボタンをクリックして設定変更モードに切り替えた後、詳細設定領域でスレッドプールの設定を変更できます。ロック設定の変更モードについての詳細は『JEUS WebAdminガイド』の「2.4.4. ロック機能」を参照してください。

3. **詳細設定**で「**Pooling**」項目にチェックを入れ、「**Dedicated**」項目を選択します。「**Min**」に0、「**Max**」に20を設定し、**[確認]**ボタンをクリックすると、変更した設定が一時的に保存され、画面上部にその結果メッセージが表示されます。

[図 3.11] スレッドプールの設定 - 専用スレッドプールの設定(1)

The screenshot displays the JEUS Scheduler configuration interface. On the left, a sidebar menu lists various system components, with 'Servers' highlighted. The main content area is titled 'Scheduler' and includes a 'HISTORY' dropdown. Below the title, there's a section for 'JEUSスケジューラに関する設定です。' (Settings for the JEUS Scheduler). The 'Resource' tab is active, showing a list of resources with 'Scheduler' selected. The 'Pooling' section is expanded, showing 'Dedicated' selected with 'Min' set to 0 and 'Max' set to 20. The 'Stuck Thread Handling' section is also visible, showing 'Max Stuck Thread Time' set to 0 ms and 'Action On Stuck Thread' set to 'Kill'. The 'Queue Size' is set to 10. The 'Keep Alive Time' is set to 0 ms. The 'Stuck Thread Check Period' is set to 0 ms. The 'Pooling' section is checked, and the 'Dedicated' option is selected. The 'Shared' option is also visible. The 'Reserved Thread Num' is set to 10. The 'Min' and 'Max' values are set to 0 and 20 respectively. The 'Keep Alive Time' is set to 0 ms. The 'Queue Size' is set to 10. The 'Stuck Thread Handling' section is expanded, showing 'Max Stuck Thread Time' set to 0 ms and 'Action On Stuck Thread' set to 'Kill'. The 'Stuck Thread Check Period' is set to 0 ms. The 'Pooling' section is checked, and the 'Dedicated' option is selected. The 'Shared' option is also visible. The 'Reserved Thread Num' is set to 10. The 'Min' and 'Max' values are set to 0 and 20 respectively. The 'Keep Alive Time' is set to 0 ms. The 'Queue Size' is set to 10. The 'Stuck Thread Handling' section is expanded, showing 'Max Stuck Thread Time' set to 0 ms and 'Action On Stuck Thread' set to 'Kill'. The 'Stuck Thread Check Period' is set to 0 ms.

4. WebAdminの左側のメニューの下部にある**[Activate Changes]**ボタンをクリックし、サーバーに反映します。サーバーへの反映が完了すると、設定したスケジューラーのスレッドプール設定がサーバーに反映され、WebAdmin画面上部にその結果メッセージが表示されます。スレッドプールの使用や種類を変更するには、動的設定が適用されない設定であるため、再開始が必要です。しかしプールの数字のみを変更する設定は動的に変更が可能です。

[図 3.12] スレッドプールの設定 - 専用スレッドプールの設定(2)

The screenshot displays the JBoss Scheduler configuration interface. On the left, the 'domain1' menu is open, showing 'Servers' selected. The main content area is titled 'Scheduler' and shows the 'Resource' tab. The 'Pooling' section is expanded, showing 'Dedicated' thread pool settings. The 'Reserved Thread Num' is set to 0, 'Min' is 0, 'Max' is 20, 'Keep Alive Time' is 60000 ms, and 'Queue Size' is 4096. The 'Stuck Thread Handling' section shows 'Max Stuck Thread Time' as 3600000 ms, 'Action On Stuck Thread' as 'None', and 'Stuck Thread Check Period' as 300000 ms. A 'Lock & Edit' button is visible in the sidebar.

domain1

Scheduler

HISTORY

JEUSスケジューラに関する設定です。

domain.xmlの設定を変更しました。

domain.xml : PENDING

servers.server.{? name == 'server1' },scheduler : PENDING

変更内容を適用するには、サーバを再起動してください。

Basic Resource Engine

Listener | Jms Resource | Jmx Manager | Scheduler | Lifecycle Invocation | External Resource

動的設定 * 必須項目 このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 TIP

Enabled ☒ [デフォルト: true] スケジューラサービスの実行有無を指定します。

詳細設定 [すべてを開く](#)

Pooling ☒

Shared

Reserved Thread Num

Dedicated

Min 0

Max 20

Keep Alive Time 60000 ms

Queue Size 4096

Stuck Thread Handling

Max Stuck Thread Time 3600000 ms

Action On Stuck Thread None

Stuck Thread Check Period 300000 ms

このページの設定を変更したい場合は、左のメニューから[Lock & Edit]ボタンをクリックしてください。 TIP

コンソール・ツールの使用

以下は、コンソール・ツール(jeusadmin)を使ってスケジューラーのスレッドプールを設定する方法です。

[例 3.2] jeusadminによるスケジューラー・サービスの共用スレッドプールの設定

```
[DAS]domain1.adminServer>show-service-thread-pool server1 -service scheduler
Shows the current configuration.
=====
+-----+-----+
(No data available)
=====

[DAS]domain1.adminServer>modify-service-thread-pool server1 -service scheduler
-min 0 -max 20
Successfully performed the MODIFY operation for The scheduler thread pool of the
server (server1)., but all changes were non-dynamic. They will be applied after
restarting.
Check the results using "show-service-thread-pool server1 -service scheduler or
modify-service-thread-pool server1 -service scheduler".

[DAS]domain1.adminServer>show-service-thread-pool server1 -service scheduler
Shows the current configuration.
=====
+-----+-----+
| Min                                | 0      |
| Max                                | 20     |
| Keep-Alive Time                    | 60000  |
| Queue Size                         | 4096   |
| Max Stuck Thread Time              | 3600000|
| Action On Stuck Thread             | NONE   |
| Stuck Thread Check Period          | 300000 |
+-----+-----+
=====
```

参考

スケジューラー・サービスのスレッドプール設定は、設定の変更を動的に反映できるため、サーバーを再起動する必要がありません。ただし、共用スレッドプールから専用スレッドプールへとスレッドプールのタイプを変更した場合は動的に反映されないため、サーバーを再起動する必要があります。

3.5. クライアント・コンテナの設定

Java EEアプリケーションを使用する場合、クライアント・コンテナで起動されるJEUSスケジューラーを設定することができます。アプリケーション・クライアントのためのJEUS DDのjeus-client-dd.xmlファイルに、以下のように<scheduler>設定を追加します。

[例 3.3] クライアント・コンテナの設定 : <<jeus-client-dd.xml>>

```
<?xml version="1.0"?>
<jeus-client-dd>
  <module-info>
    ...
  </module-info>
  ...
  <scheduler>
    <enabled>true</enabled>
    <!-- Scheduler Thread-pool settings -->
    <pooling>
      <dedicated>
        <min>2</min>
        <max>10</max>
        <keep-alive-time>60000</keep-alive-time>
        <queue-size>4096</queue-size>
        <stuck-thread-handler>
          <max-stuck-thread-time>3600000</max-stuck-thread-time>
          <action-on-stuck-thread>None</action-on-stuck-thread>
        </stuck-thread-handler>
      </dedicated>
    </pooling>
  </scheduler>
</jeus-client-dd>
```

設定ファイルを修正した後、JEUSサーバーは再起動する必要はありませんが、修正した設定内容を反映するためにクライアント・モジュールとクライアント・コンテナは再起動する必要があります。

注

クライアント・コンテナでスケジューラー・サービスを使用する場合は、共用スレッドプールを使用できません。

クライアント・コンテナとアプリケーション・クライアントについては、『JEUS アプリケーションクライアントガイド』を参照してください。

以下は、クライアント・コンテナで実行するジョブの設定例です。

[例 3.4] Job-list設定 : <<jeus-client-dd.xml>>

```
<scheduler>
  ...
  <job-list>
    <job>
      <class-name>samples.ScheduleJob</class-name>
      <name>ScheduleJob</name>
      <description>This is a sample for scheduler service</description>
      <begin-time>2011-02-01T00:00:00</begin-time>
      <end-time>2011-03-01T00:00:00</end-time>
      <interval>
        <minutely>30</minutely>
      </interval>
      <count>-1</count>
    </job>
  </job-list>
</scheduler>
```

索引

A

abstract class RemoteSchedule抽象クラス, 7

C

cancel, 15

D

daemon, 9, 15, 18

E

EJBタイマー・サービス, 1, 19

F

Fixed-delay方式, 12

Fixed-rate方式, 12

I

initialize() Callback, 9

J

jeus.scheduleパッケージ,

JeusSchedulerException例外, 7

JEUSスケジューラー, 1

Job-list, 6, 21, 23, 25

L

Lookup, 11

N

nextTime, 7, 8

O

onTime, 7, 8, 9

R

registerSchedule, 13

registerScheduleAtFixedRate, 13

boolean isDaemon, 13

boolean isThreaded, 14

Date endTime, 13

Date firstTime, 13

long delay, 13

long maxcount, 13

long period, 13

RemoteSchedule, 8, 9

S

ScheduleControllerインターフェース, 7

ScheduleListenerインターフェース, 7

SchedulerFactory, 5, 11, 15, 20

SchedulerFactoryクラス, 7

Schedulerインターフェース, 7

Schedule抽象クラス, 7

SERVER_SCHEDULER_NAME, 11, 17

か

クライアント・コンテナ, 39

さ

スタンドアロン・スケジューラー, 20

