↑ Top

目次

改訂情報

変更年月日	変更内容
2012-10-01	初版
2015-04-01	第2版 下記を追加・変更しました ■ 「 *モニタリング* 」の「 *過去の実行結果* 」を修正
2015-12-01	第3版 下記を追加・変更しました ■ 「 *モニタリング* 」の「 *過去の実行結果* 」を修正

はじめに

本書の目的

本書ではジョブスケジューラの仕様について述べます。

対象読者

本書は、以下の条件を満たす人を対象としています。

- intra-mart Accel Platform を理解している
- ジョブスケジューラの管理者
- 以下のいずれかの条件を満たす、ジョブスケジューラ機能を利用する開発者
  Java を理解している開発者
  サーバサイドJavaScript を理解している開発者

本書の構成

本書は、以下のような内容で構成されています。

- *概要*

  ジョブスケジューラの概要について説明しています。
- *ジョブスケジューラの構成*

  ジョブスケジューラとジョブ、ジョブネット、トリガの構成について説明しています。
- *スケジュールの定義方法*

  スケジュールの定義方法について説明しています。
- *モニタリング*

  ジョブネットの実行状況の監視と操作、実行履歴について説明しています。
- *設定*

  ジョブスケジューラの各種設定について説明しています。
- *サンプル*

  サンプルによるジョブのプログラミング方法について説明しています。

概要

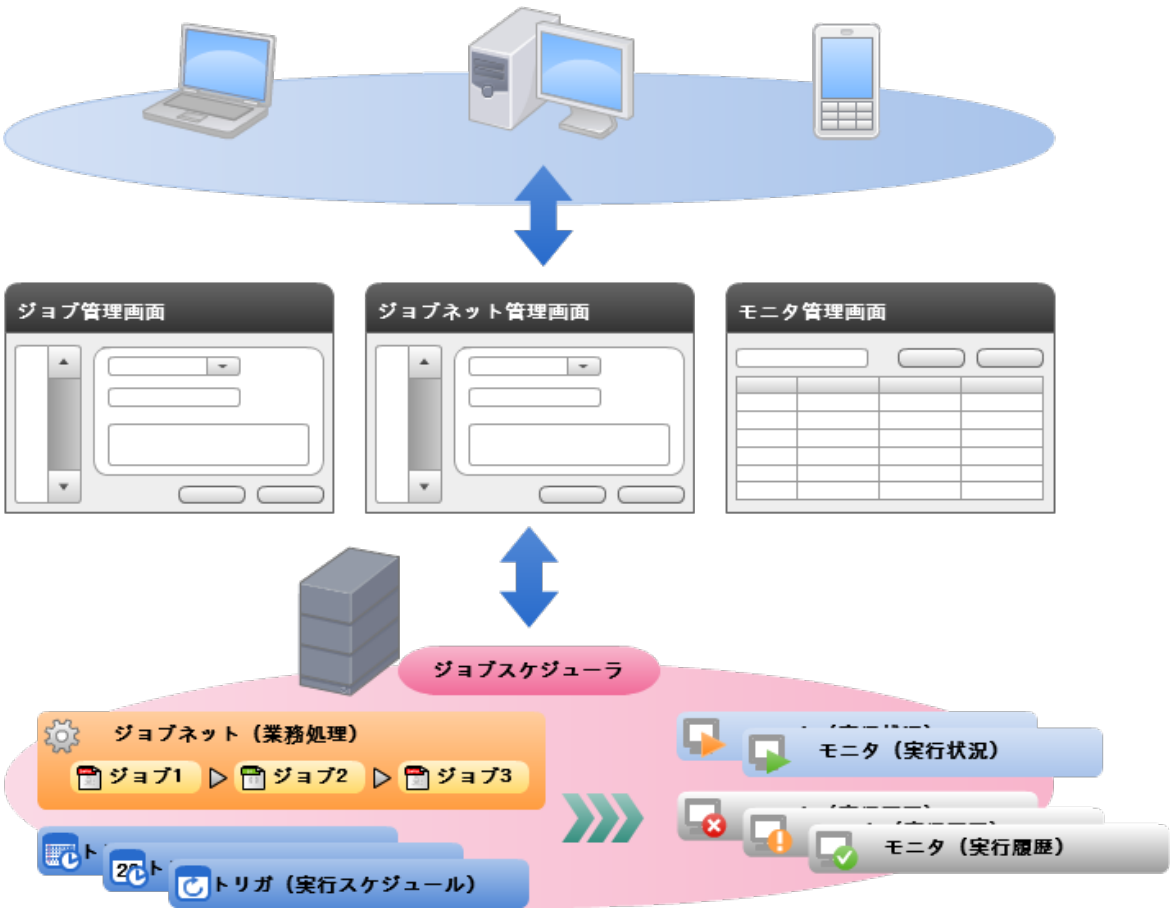
ジョブスケジューラとは

ジョブスケジューラとは、ジョブと呼ばれる処理単位に事前にいつ実行するかというスケジュールを定義しておくことで自動的に実行する機能です。1つの業務を定期的に複数回実行する場合や、大量データを扱うため時間かかる業務処理を夜間に実行する場合などに利用します。

intra-mart Accel Platform のジョブスケジューラは、このような要求を実現するためサーバ上の Java や サーバサイドJavaScript で構成された 任意の業務処理をスケジュールで定義されたタイミングで自動的に実行する機能や、実行状況の監視や実行結果の管理を行うための機能を提供します。

全体像

全体の構成イメージ



用語

ジョブスケジューラ

ジョブネットをトリガのスケジュールに従って自動的に実行する intra-mart Accel Platform のバックエンドサービスです。

ジョブ

実際に実行されるプログラムなどをジョブスケジューラで利用するために定義したものです。

ジョブネット

複数のジョブで構成されジョブスケジューラから実行される業務処理を定義したものです。

トリガ

ジョブネットを実行するスケジュールなどを定義したものです。
1つのジョブネットに対して複数のトリガを定義することができます。

モニタ

現在実行されているジョブネットの実行状況です。
ジョブネットの実行が完了すると実行結果が格納されて実行履歴として利用することができます。

ジョブスケジューラの流れ

1. **ジョブ実行プログラムの作成**
   実際に実行されるプログラムを Java 、または サーバサイドJavaScript で作成します。
   作成したプログラムは、intra-mart上に配置しておきます。

2. **ジョブの作成**
   実行プログラムにパラメータや名前などの管理情報を設定してジョブを作成します。
   作成したジョブは、ジョブネットを作成する際に利用できるようになります。
   ジョブの作成は、「ジョブ管理」→「ジョブ設定」画面または、ジョブスケジューラAPIを利用して行います。

3. **ジョブネットの作成**
   ジョブを組み合わせて業務処理の流れを定義したジョブネットを作成します。
   作成されたジョブネットは、ジョブスケジューラから実行できるようになります。
   ジョブネットの作成は、「ジョブ管理」→「ジョブネット設定」画面または、ジョブスケジューラAPIを利用して行います。

4. **トリガの作成**
   ジョブネットに対して実行するスケジュールを定義したトリガを作成します。
   トリガが作成されると、ジョブスケジューラによって対象のジョブネットが自動的に実行されるようになります。
   トリガの作成は、「ジョブ管理」→「ジョブネット設定」画面または、ジョブスケジューラAPIを利用して行います。

5. **ジョブネットの自動実行**
   ジョブスケジューラは、トリガに定義されたスケジュールにしたがって対象のジョブネットを実行します。
   ジョブネットが実行されると、実行状況を管理するためのモニタが自動的に作成されます。

6. **実行状況の確認**

   ジョブスケジューラで生成されたモニタを利用してジョブネットの実行状況を確認することができます。
   ジョブネットの実行監視を行ったり、実行中のジョブネットに対して停止や再開などの操作を行うことができます。
   モニタの確認や操作は、「ジョブ管理」→「ジョブネットモニタ」画面または、ジョブスケジューラAPIを利用して行います。

7. **実行履歴の確認**

   ジョブネットの実行が完了すると、モニタには終了時間やジョブネットの実行結果が格納され過去の実行履歴になります。
   モニタの確認や操作は、「ジョブ管理」→「ジョブネットモニタ」画面または、ジョブスケジューラAPIを利用して行います。

コラム

プログラムの作成方法については、「サンプルプログラム」を参照してください。
ジョブスケジューラの画面の操作方法については「 テナント管理者 操作ガイド 」を参照してください。

ジョブスケジューラの構成

ジョブスケジューラ

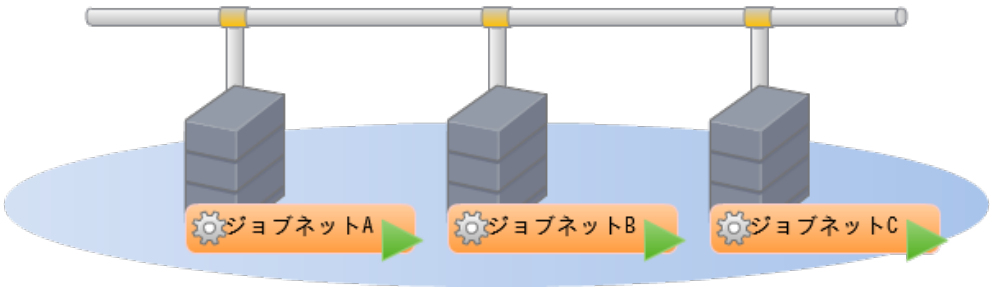
ジョブスケジューラは、任意の業務処理をスケジューリングされたタイミングで自動的に実行するための機能で、intra-mart Accel Platform のバックエンドサービスとして動作します。

intra-mart Accel Platform に設定されたジョブ、ジョブネット、トリガと呼ばれる業務処理と実行スケジュール定義にしたがって自動的にプログラムの実行と実行状況の管理を行います。

複数ノード構成

ジョブスケジューラは、複数のノードで実行させることできます。
複数ノードでジョブスケジューラを実行すると、同じタイミングで実行されるジョブネットが複数あった場合などに、各ノードで分散して効率的に処理を行うことができます。

複数ノード構成イメージ図



ジョブ

ジョブとは、ジョブスケジューラで実行する業務処理を定義する際に利用される最小の処理単位を定義したものです。
intra-mart Accel Platform 上に配置された、実際に実行されるJavaプログラム、またはサーバサイドスクリプトプログラムと実行時のパラメータにジョブを管理するための識別情報としてID、カテゴリ、名前、説明を付与します。

> **コラム**
> ジョブプログラムの作成方法の詳細については、「サンプルプログラム」を参照してください。

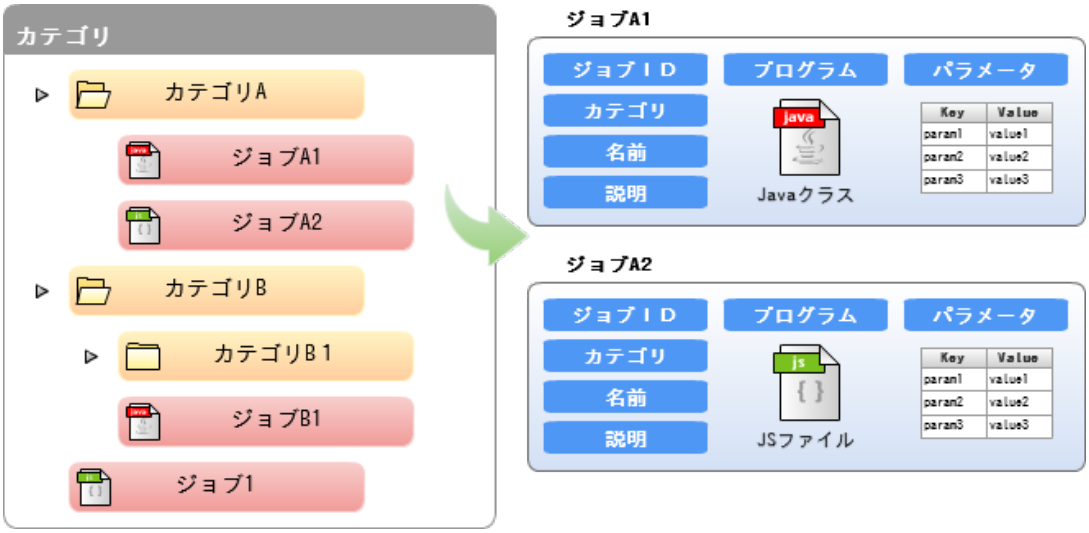
ジョブの定義情報

名前	説明
ジョブID	intra-mart Accel Platform 上でジョブを識別するためのユニークなIDです。
カテゴリ	ジョブを分類、管理するために付与する所属情報です。階層構造になっているため細かい分類定義することができます。

名前	説明
名前、説明	ジョブに対して名前と仕様や動作などを詳細な情報を付与するための説明を多言語で付与することができます。多言語で設定された情報は、画面上でユーザアカウントに設定された言語で表示することが可能です。
プログラム	intra-mart Accel Platform 上に存在する Java または、サーバサイドJavaScript です。
パラメータ	ジョブ定義に設定できる実行プログラムで利用するパラメータ情報です。設定されたパラメータは、ジョブスケジューラのコンテキストに格納されて実行プログラムに渡されます。

※ジョブのパラメータがどのように利用されるかは、プログラム実装によりますので動作仕様を理解した上で適切なパラメータを設定してください。

ジョブの構成図



ジョブネット

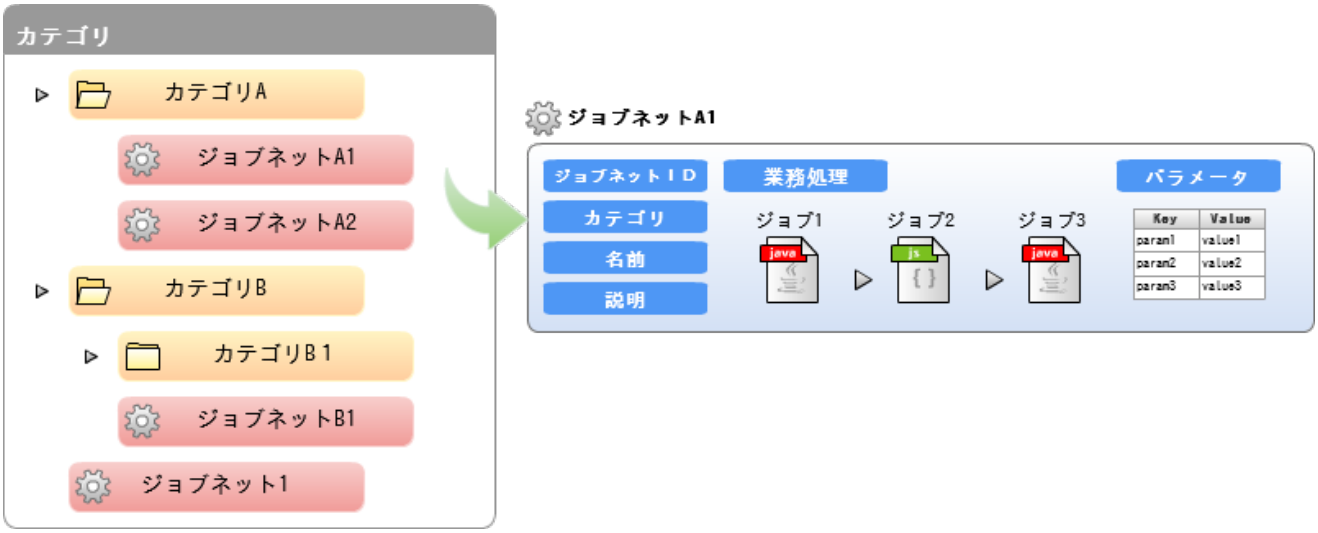
ジョブネットとは、ジョブスケジューラから実行される業務処理を定義したものです。
ジョブネットは、ジョブの集合で構成されておりジョブの実行順序やパラメータに、ジョブと同様にIDや名前など intra-mart Accel Platform 上でジョブネットを管理するための識別情報を付与します。

ジョブネットの定義情報

名前	説明
ジョブネットID	intra-mart Accel Platform 上でジョブネットを識別するためのユニークなIDです。
カテゴリ	ジョブネットを分類、管理するために付与する所属情報です。階層構造になっているため細かい分類定義することができます。
名前、説明	ジョブネットに対して名前と仕様や動作などを詳細な情報を付与するための説明を多言語で付与することができます。多言語で設定された情報は、画面上でユーザアカウントに設定された言語で表示することが可能です。
業務処理	ジョブネットが実行する業務処理の流れをジョブを利用して定義したものです。

名前	説明
パラメータ	ジョブの実行パラメータと同様で、ジョブネット定義に設定できるパラメータです

ジョブネットの構成図



トリガ

トリガとは、ジョブネットを実行するスケジュールを定義したもので、1つのジョブネットに対して複数設定することができます。
トリガは、スケジュール定義とパラメータに intra-mart Accel Platform 上でトリガを管理するための識別情報を付与します。

スケジュールには、繰り返し実行や日時を指定、intra-mart Accel Platform のカレンダで定義された営業日に実行させるなどの定義方法が利用できます。

トリガの定義情報

名前	説明
トリガID	上でトリガを識別するためのユニークなIDです。
スケジュール定義	ジョブネットの実行契機をさまざまな方法で定義したものです。日時を利用したスケジュール、繰り返し実行するスケジュール intra-mart Accel Platform のカレンダ定義の営業日に実行するスケジュール定義を行うことができます。
備考	トリガに対して簡単な備考を設定することができます。多言語には対応していません。

トリガの構成図


トリガ
ジョブネット1
繰り返しトリガ1
日時指定トリガ2
営業日トリガ3
ジョブネット2
繰り返しトリガ4
繰り返しトリガ1
トリガID
スケジュール
備考
10分間隔で20回繰り返す
日時指定トリガ2
トリガID
スケジュール
備考
毎月20日の18時00分00秒
営業日トリガ3
トリガID
スケジュール
備考
カレンダの営業日の9時00分00秒

スケジュールの定義方法

トリガに設定するスケジュールは、以下の方法で定義することが可能です。
スケジュール定義は、「ジョブ管理」→「ジョブネット設定」画面で行います。

日時指定

日時指定は、タイムゾーンと年月週日時分を利用して実行するスケジュールを定義する方法です。
指定した日時に1回だけ実行するようなスケジュールや日時の設定によって繰り返し実行するようなスケジュール定義が可能です。

日時指定
　年月日時分を指定して、指定した日時に1回だけ実行するスケジュール定義

毎年
　月日と時分を指定して、毎年同じ日の同じ時間に実行するスケジュール定義

毎月
　日と時分を指定して、毎月指定した日の同じ時間に実行するスケジュール定義

毎週
　曜日と時分を指定して、毎週指定した曜日の同じ時間に実行するスケジュール定義

毎日
　時分を指定して、毎日指定した時間に実行するスケジュール定義

毎時
　分を指定して、1時間毎に指定した分に実行するスケジュール定義

毎分
　毎分0秒に実行するスケジュール定義

コラム
上記の組み合わせによって毎月1日の毎時0分と30分に実行といった様々なスケジュール定義が可能です。

繰り返し

繰り返し指定は、繰り返し回数と繰り返し間隔（単位：秒）を指定してのスケジューリングが可能です。
この定義を利用することで即時実行するスケジュールや繰り返し間隔（単位：秒）で無限に実行するスケジュールなども定義可能です。

即時
　繰り返し回数を1回とすることで即時実行するスケジュール定義

無限
　繰り返し回数を指定しないことで、指定された繰り返し間隔（単位：秒）で無限に繰り返し実行するスケジュール定義

営業日

営業日指定は、intra-mart Accel Platform のカレンダーAPIを利用した営業日のみ実行するスケジューリングが可能です。
対象のカレンダーとタイムゾーンと時分を指定することで、カレンダーで定義された営業日の指定された時間にジョブネットを実行させます。

モニタリング

ジョブスケジューラからジョブネットが実行されるとジョブネットの実行状況を表すモニタが生成されます。
モニタは、現在実行中のジョブネットの監視や過去に実行されたジョブネットの履歴として利用します。
モニタの情報は「ジョブ管理」→「ジョブネットモニタ」画面で参照することが出来ます。

現在の実行状況

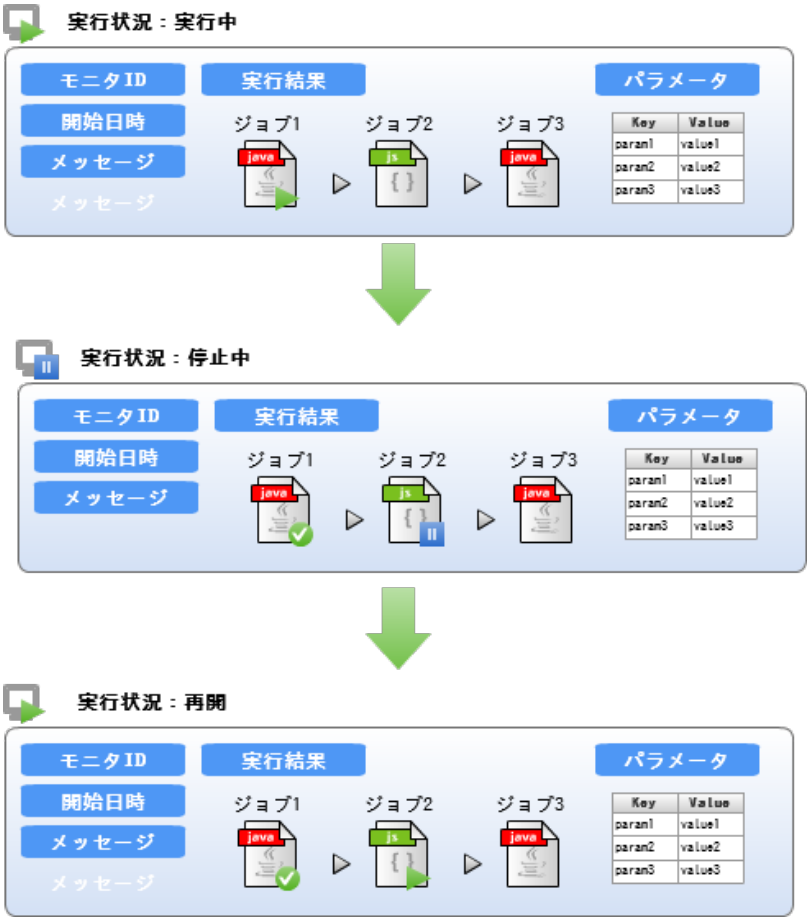
現在実行されているジョブネットのモニタには、以下のステータスが設定されています。

ステータス	説明
実行中	ジョブネットが実行中であることを表します。
停止中	ジョブネットが停止中であることを表します。
停止処理中	ジョブネットの実行中に停止要求を受け付けた状態であることを表します。
再開処理中	ジョブネットの停止中に再開要求を受け付けた状態であることを表します。
終了処理中	ジョブネットの実行中、または停止中に終了要求を受け付けた状態であることを表します。

ジョブネットの操作

モニタを利用することで、実行中のジョブネットに対して停止や再開、終了などを行う事ができます。
ジョブスケジューラは、ジョブネットに対する操作要求を受け付けると現在実行中のジョブが完了したタイミングで要求された操作を実行します。

ジョブネットとジョブの構成図に操作を受け付けた時の停止、再開イメージ



ジョブネットに対する操作は、ステータスによって実行可能な操作が制限されています。

各ステータスと可能な操作の一覧

ステータス	可能な操作	説明
実行中	停止 / 終了	停止処理中、終了処理中にステータスが変更され実行中のジョブ処理が完了すると停止、または終了する。
停止中	再開 / 終了	再開処理中、終了処理中にステータスが変更され停止されたジョブ処理から再開、または終了する。
停止処理中	再開 / 終了	停止処理をキャンセルして再開処理中、終了処理中にステータスが変更されジョブネットをそのまま続行、または終了する。
再開処理中	停止 / 終了	再開処理をキャンセルして停止処理中、終了処理中にステータスが変更され実行中のジョブ処理が完了すると停止、または終了する。
終了処理中	再開 / 停止	終了処理をキャンセルして再開処理中、停止処理中にステータスが変更されジョブネットをそのまま続行、または実行中のジョブ処理が完了すると停止する。

過去の実行結果

終了したジョブネットのモニタには、以下のステータスが設定されています。

ステータス	説明
成功	全てのジョブが正常に終了してジョブネットが完了したことを表します。
エラー	ジョブでエラーが発生したためにジョブネットが途中で終了したことを表します。
警告	1つ以上のジョブで警告（継続可能なエラー）が発生してジョブネットが完了したことを表します。
強制終了	ジョブネットの実行中に終了要求をされたために、ジョブネットが途中で終了したことを表します。

注意

モニタからジョブネットを再実行すると、現在ジョブネットに対して設定されている情報で即時実行されます。
終了したジョブと同じ設定で実行されるのではなく、現在の設定で実行される点に注意してください。
実行時の情報を変更する場合は、ジョブ情報画面およびジョブネット情報画面から操作を行ってください。

実行結果の通知

ジョブネットの実行が完了すると、実行されたジョブネットと実行結果などをプロパゲーションに対して通知します。
プロパゲーションの受信プログラムを作成して設定を行うことで、ジョブネットの完了をさまざまな手段で通知することが可能になります。

設定

ジョブスケジューラの設定

詳細は「設定ファイルリファレンス」-「ジョブスケジューラ」を参照してください。

サンプル

ジョブのサンプルプログラムを利用してジョブプログラムの作成方法について説明します。

サンプルプログラム

Java と サーバサイドJavaScript で作成されたジョブのサンプルプログラムです。
固定文字列”Hello.”に、”message”というキーに設定されたパラメータ値を連結して出力します。

- Java の場合

```java
package jp.co.intra_mart.example.job;

import jp.co.intra_mart.foundation.context.Contexts;
import jp.co.intra_mart.foundation.context.model.AccountContext;
import jp.co.intra_mart.foundation.job_scheduler.Job;
import jp.co.intra_mart.foundation.job_scheduler.JobResult;
import jp.co.intra_mart.foundation.job_scheduler.JobSchedulerContext;
import jp.co.intra_mart.foundation.job_scheduler.annotation.Parameter;
import jp.co.intra_mart.foundation.job_scheduler.annotation.Parameters;
import jp.co.intra_mart.foundation.job_scheduler.exception.JobExecuteException;

public class HelloJob implements Job {

   public HelloJob() {
   }

   @Override
   @Parameters(@Parameter(key = "message", value = "world!"))
   public JobResult execute() throws JobExecuteException {
      try {
         // アカウントコンテキスト
         final AccountContext accountContext = Contexts.get(AccountContext.class);
         System.out.println("Account context : " + accountContext.toString());

         // ジョブスケジューラコンテキスト
         final JobSchedulerContext jobSchedulerContext = Contexts.get(JobSchedulerContext.class);
         System.out.println("Job scheduler context : " + jobSchedulerContext.toString());

         // パラメータの取得
         final String message = jobSchedulerContext.getParameter("message");
         if (null == message) {
            // 処理結果:異常
            return JobResult.error("パラメータにメッセージが存在しません。");
         } else if (message.trim().isEmpty()) {
            // 処理結果:警告
            return JobResult.waring("メッセージが空です。");
         }
         // メッセージの表示
         System.err.println("Hello. " + message);
         // 処理結果:正常
         return JobResult.success("ジョブが正常に実行されました。");
      } catch (Exception e) {
         // 処理結果:異常(例外による処理結果の返却)
         throw new JobExecuteException("予期しないエラーが発生しました。", e);
      }
   }

}
```

Java のジョブプログラムは、jp.co.intra_mart.foundation.job_scheduler.Jobの実装クラスを作成します。
このインタフェースには、ジョブの実行処理を記述するためのexecuteメソッドが定義されています。
ジョブ開発者は、このexecuteメソッドにジョブ処理で実行したいプログラムを記述します。

- サーバサイドJavaScript の場合

```
/**
 * @parameter message world!
 */
function execute() {
  let accountContext = Contexts.getAccountContext();
  let jobSchedulerContext = Contexts.getJobSchedulerContext();
  let message = jobSchedulerContext.getParameter('message');
  if (null == message) {
    return {
      status : 'error' ,
      message : 'パラメータにメッセージが存在しません。'
    };
  } else if ('' == message) {
    return {
      status : 'warning' ,
      message : 'メッセージが空です。'
    };
  }
  Debug.console('Hello. ' + message);
  return {
    status : 'success' ,
    message : 'ジョブが正常に実行されました。'
  };
}
```

サーバサイドJavaScript のジョブプログラムは、任意のJSファイルにexecute関数を記述します。
ジョブ開発者は、このexecute関数にジョブ処理で実行したいプログラムを記述します。

ジョブ処理では、アカウントコンテキストとジョブスケジューラコンテキストが取得可能です。
この2つのコンテキストを利用して任意の業務処理を記述します。

アカウントコンテキスト

アカウントコンテキストには、ジョブスケジューラから実行されたことを表すアカウント情報が格納されています。

ジョブスケジューラコンテキスト

ジョブスケジューラコンテキストには、ジョブ、ジョブネット、トリガの定義情報と実行日時などの実行情報が格納されています。